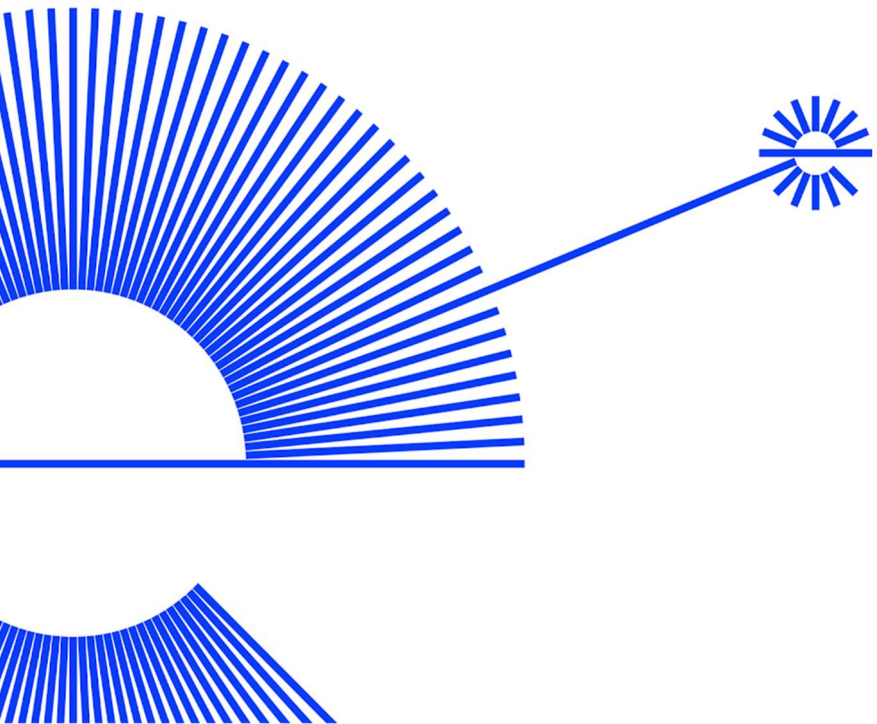
Preservation Metadata for Software

Describing software in archives

Version 1.0 | September 2021



dutch digital heritage network

Contents

Introduction

Software preservation is concerned with the long-term storage of and access to computer programs in order to keep software alive. As archival objects, software presents us with many challenges. The notion of software covers a huge variety of applications that have complex relationships with often specific hard- and software environments that are maybe considered quite standard at the time of release, but over time become obsolete. In order for archived software to remain accessible we need a lot of information about the original environments in which the software worked at the time of release so that it can be reconstructed, either by using original hard- and software or, more likely, by setting up an emulation of the original environment.

This report addresses the challenges of how to capture information about the original environment in (preservation) metadata. It will first provide an overview of the types of metadata that one needs to consider when describing software, but also look at standards that will facilitate standardization and interoperability with systems for accessing software collections.

It will then zoom in on system requirements: lists of properties that inform the user of software about some basic requirements that the environment must adhere to. The focus on this originally provided information at the time of publication is intentional. It is up to institutions to decide whether they want to go one step further and also describe the emulation environments that they want to use for providing access to software from their collections. However, the emulation environments will change over time, whereas the system requirements that were originally given will stay the same and should always help to set-up emulation environments at a certain point in time.

In the following two chapters we first evaluate PREMIS as a model for describing system requirements. We then explore Wikidata as a potential semantic database that can be used as a vocabulary. Both of these chapters are exploratory in nature and will critically reflect on the degree to which PREMIS and Wikidata can play a role in describing system requirements. The final chapter will bring together the conclusions from the preceding chapters and make recommendations.

Even though a lot of decisions will still need to be made based on the specific situation and goals of the archiving institution, we hope that this report will be a rich resource that can serve as a starting point, a check-list with considerations to make the right decisions and by doing so lowering the threshold for starting or advancing software archiving.

1. Overview of Software Metadata

In this chapter we will first provide an overview of aspects that need to be considered within metadata for the preservation of software. It will also address a few recommendations for the *form* that the metadata should take in order to facilitate efficient and high quality work flows.

1.1 Metadata for software preservation

Metadata can be characterised in different ways. Brown uses the terms “descriptive” and “technical” metadata[1]. Descriptive metadata documents each information object. Technical metadata describes each data object. Instead of the term “technical metadata” we will use the term “preservation metadata” as it describes its purpose better. Preservation metadata helps to maintain the chain of custody of a preserved object, to document its version and modifications, to document and automate preservation actions and to document how an object can be accessed.

The metadata listed in appendix 1 are relevant for software preservation in general. Which one of these metadatafields are being used depends on the content and purpose of the archive. The minimum preservation metadata is according to Brown (p.176) beside descriptive metadata:

- Name,
- file size,
- format (PRONOM PID, or a MIME type),
- last modified date,
- checksum value and type

This is a list that fits singular files that are not executable themselves. For executable software, we need to take into consideration that it needs a specific environment to run. Without information about this environment in the metadata, the archived software essentially becomes impossible to run. Besides consisting of executable files, software also most often consists of more than one file. For these two reasons, we recommend that the above list is extended with the following element:

- system requirements.

Chapter 3 will discuss what the system requirements encompass and which ones are the essential requirements that need to be described in the metadata.

We marked this list of minimum preservation metadata in appendix 1, as part of a more comprehensive list of metadata for software archiving. With these minimum metadata, it is not guaranteed that the software will run or the file can be opened in the future. But they increase the chance that it will be possible with additional effort to reconstruct the original environment or set up a

[1] Brown, Adrian (2013): Practical Digital Preservation : A How-to Guide for Organizations of Any Size, p.155

compatible emulation environment. In order to preserve the functionality of the software or a complex digital object, additional metadata such as the description of significant properties, behaviours, dependencies and interactions might be necessary. Again, a lot depends on the purpose of the archiving institution and the degree to which they want to facilitate their designated community.

In addition to appendix 1, appendix 2 lists resources and literature about metadata used in digital preservation in general, but also domain specific resources such as metadata for video games, legacy software etc. The list is not exhaustive. It includes literature about technical preservation metadata of disk images and environments, metadata standards, and linked metadata.

1.2 Consistency, automation and interoperability of metadata

Apart from the information that needs to be contained in the metadata, decisions also need to be made on the way in which the information is described. It is important to achieve the highest quality and consistency for preservation metadata in order for future users of the archive to be able to interpret the information even decades after software has gone out of common use. Archives can increase quality, interoperability and automation by using metadata standards, controlled vocabularies, linked data registries and tools for automation.

Metadata standards
Metadata standards such as METS and PREMIS also improve the interoperability of preservation metadata. Interoperability of preservation metadata becomes more and more important in regards to the effort needed to create a functioning software environment. Even if the environment could not be shared for copyright reasons, at least its metadata could be shared and would enable other external users to assemble an environment either manually or automatically. Chapter 4.2 will explain how system requirements can be modelled in PREMIS.

Controlled vocabularies
There is also the question of the contents of these metadata fields and the form it takes. There are a number of ways to increase the quality of the metadata by using terminology consistently throughout a collection, using controlled vocabularies.
A domain specific example for controlled vocabulary is https://gamemetadata.soe.ucsc.edu/ to describe games. The Software Ontology (SWO) is a resource for describing software tools, their types, tasks, versions, licences, provenance and associated data[2]. The software tools listed in the Software Ontology support mainly scientific research. CodeMeta is another controlled vocabulary to describe scientific software[3].

Linked data registries
Linked data allows the flexible aggregation and interpretation of metadata across different repositories. It also makes it easier to handle semantic shift. If terms get out of use they can easily be linked to a new term that is more contemporary. The use of linked data registries can be applied to various types of metadata:

- Descriptive metadata. For named entities such as producers, locations, titles, etc. linked data can be used. However, since this is not specific to software preservation we will not include more information about that in this report.
- Preservation metadata. For preservation metadata a linked data database such as Wikidata could be used. Wikidata is essentially a linked data version of the online encyclopedia Wikipedia. Chapter 4.1 will address the use of Wikidata for describing system requirements, as many softwares have a wikipedia page that often provides useful information for preservation.

[2] Software ontology tool: https://www.ebi.ac.uk/ols/ontologies/swo
[3] CodeMeta: minimal metadata schema for science software and code, in JSON and XML https://github.com/codemeta/codemeta

Means for automation

To further increase the quality and completeness of the metadata, as well as keeping the workflows of keeping such repositories manageable, various means for automatic metadata generation can be used.

- File format identification. Common practice is the use of the PRONOM file format registry. File identification tools like Siegfried or DROID are based on the PRONOM file registry. This registry provides persistent links for the items that it contains. This allows for third parties to persistently link to the PRONOM repository with additional information about specific file types, rather than generating this information themselves each time they ingest a specific file format.

- Tools for administrative and provenance metadata generation. Digital preservation systems such as Archivematica or Preservica produce certain administrative and provenance metadata automatically. In particular, such systems automatically document preservation actions executed or triggered through them. As there is no unified description of these preservation actions between different preservation systems, Addis et al. (2018)[4] suggest to create and use the vocabulary of the preservation action registry (PAR[5]) that will be integrated in Archivematica and Preservica. This preservation actions registry is - amongst others - based on PREMIS events[6]. This registry was created with in mind more traditional, non-interactive archival objects. Typical actions are file format normalisation, checksum calculation, or the creation of a compressed display copy. According to Addis et al. (2018) it is planned to extend the preservation action registry to software preservation[7].

- Machine readable metadata file formats. If metadata is to be used in automated processes it should be machine readable. There are several metadata file formats such as JSON, CSV, XML, RDF-XML and JSON-LD for this purpose[8]. PREMIS and Wikidata can be serialised in such file formats. Automated processes such as checksum calculation for bit preservation or normalisation of audiovisual files are already common today. Complexer processes such as the automatic assembly of a software environment out of single digital objects is not a practice yet, but will be a goal in the future. Machine readable, precise and complete preservation metadata is a prerequisite for this.

[4] Addis M, Simpson J, Tilbury J, O'Sullivan J, Stokes P (2018) Digital Preservation Interoperability through Preservation Actions Registries. In: iPres 2018
[5] https://parcore.org/
[6] https://id.loc.gov/vocabulary/preservation/eventType.html
[7] Addis et al. (2018), p.4: "This [the extension of PAR] includes use of techniques such as emulation and containerisation as part of capturing/preserving the original environment for digital content and maintaining this environment over time. There is already interesting work in this area, e.g. ReproZip, Singularity, Encapsulator and Research Objects that we would seek to build upon.
[8] Dappert A, Guenther RS, Peyrard S (eds) (2016) Digital Preservation Metadata for Practitioners: Implementing PREMIS. Springer, Cham. P. 162

2. Metadata representing system requirements

Metadata for software preservation has the potential for a seemingly endless scope depending on the intended use, audience and context of said software. The possibility for multiple configurations of the same software creates a logistical problem for institutions looking to document and maintain certain pieces of software going forward. Regardless of an institution's plans for future access to obsolete applications, the key piece of information that needs to be gathered at as early a stage as possible is the system requirements. System requirements generally accompany a particular piece of software at the time of publication and usually contain both a minimum and recommended set of fields that are necessary for the application to run. These requirements can be a combination of both hardware components and additional software, an example being that of an operating system which depends on a certain machine specification and then a software application which depends on a certain operating system. There can be some overlap when determining the layered nature of such system requirements but for the purpose of this investigation we will look at operating systems and software applications separately, with specific examples.

2.1 Examples of System requirements supplied by software developers

Operating systems form the base layer upon which other software applications can be installed. The operating system will be referenced in the system requirements for any subsequent software applications but they also count as software in their own right and should be provided the same care when it comes to documentation. Below is a table with three common operating systems and their respective system requirements.[9]

[9] System requirements have been taken from various sources online, either through official documentation or otherwise. Links for each will be provided but it's worth highlighting the variations and scope of what is included for each individual example. Where possible the best source of this information would be directly from the source media documentation or manual.

Windows XP[10]	OS X (10.8 Mountain Lion)[11]	Ubuntu 14.04[12]
• Pentium 233-megahertz (MHz) processor or faster (300 MHz is recommended) • At least 64 megabytes (MB) of RAM (128 MB is recommended) • At least 1.5 gigabytes (GB) of available space on the hard disk • CD-ROM or DVD-ROM drive • Keyboard and a Microsoft Mouse or some other compatible pointing device • Video adapter and monitor with Super VGA (800 x 600)or higher resolution • Sound card • Speakers or headphones	• Mac computer with an Intel Core 2 Duo, Intel Core i3, Intel Core i5, Intel Core i7, or Xeon processor • 2GB of memory • 8 GB of disk space • A working internet connection • Latest version of Mac OS X Lion (10.7.x)	• 1 GHz processor (for example Intel Celeron or better) • 1.5 GB RAM (system memory) • 7 GB of free hard drive space for installation • Either a CD/DVD drive or a USB port for the installer media • Internet access is helpful (for installing updates during the installation process).

Table 1: Three common operating systems and their system requirements.

From these examples, only Windows XP specifies the need for peripherals (ie. keyboard, mouse, speakers) but it can generally be assumed that this kind of information would be required for the other operating systems also. Thus, system requirements defined by software developers may assume tacit knowledge that might not be obvious 20 years later.

Software applications can include a wide and varied range of system requirements depending on their purpose. All will generally require a specific operating system and some configuration of hardware resources to run. The below table details several examples of software applications and the system requirements.

Microsoft Office 200313	QuarkXpress 9.014	LibreOffice 4.3.115
• Min Operating System Microsoft Windows 2000 SP3 or later, Microsoft Windows XP or later • Processor Type Pentium	• Mac OS® 10.5.8 (Leopard®), Mac OS 10.6.4 (Snow Leopard®) or later • Tested on Citrix Hardware • Mac® Intel® processor	• Linux kernel version 2.6.18 or higher; • glibc2 version 2.5 or higher; • gtk version 2.10.4 or higher;

[10] https://www.24hoursupport.com/windows-xp-hardware-requirements/
[11] https://kb.wisc.edu/helpdesk/page.php?id=25648
[12] https://wiki.ubuntu.com/TrustyTahr/ReleaseNotes/UbuntuGNOME
[13] https://www.cnet.com/products/office-professional-2003/specs/
[14] http://files.quark.com/download/documentation/QuarkXPress/9/English/QXP_9.0_ReadMe_en-us.pdf
[15] https://sourcedigit.com/12430-install-libreoffice-4-3-1-ubuntu-14-04/

• Processor Speed 233 Hz • Min RAM Size 128 MB • Min Hard Drive Space 400 MB	• 2GB RAM (1GB minimum) • 2GB hard disk space • An Internet connection for activation • DVD-ROM drive for installation from DVD (not required for installation from download)	• Pentium-compatible PC (Pentium III, Athlon or more-recent system recommended); • 256Mb RAM (512Mb RAM recommended); • Up to 1.55Gb available hard disk space; • X Server with 1024×768 resolution (higher resolution recommended), with at least 256 colors; • Gnome 2.16 or higher, with the gail 1.8.6 and at-spi 1.7 packages (required for support for assistive technology [AT] tools), or another compatible GUI (such as KDE, among others).

Table 2: Examples of software and their system requirements

These software system requirements supplied by software developers partly repeat the system requirements necessary for the installation of the operating system.

2.2 Types of system requirements

From the above examples, although not an exhaustive depiction of all possible system requirements, some key attributes can be identified as being integral parts of the documentation of operating systems and software applications.

The primary (generic) system requirements relate to the host system on which the application will run:

- Operating system[16]
- Computer architecture / CPU / instruction set
- Minimum CPU- performance (clock speed, CPU model)
- GPU (minimum video memory, API such as OpenGL, DirectX etc.)
- Minimum system memory (RAM)
- Minimum free storage space
- Dependency on software platform / library / runtime
- Required machine type (for instance desktop computer, laptop, electronic device, mobile device)

Additionally, an operating system or software application might rely on specific peripheral items or network connection to work.

- External storage devices such as floppy, CD-ROM, DVD-ROM drive[17]
- Video Adapter (connector for video projector or monitor) and monitor (resolution specified)
- Input devices such as keyboard, mouse, joystick, touch screen
- Sound interfaces such as speakers, headphones, microphones, sound card
- Other peripherals and its ports such as SCSI, RS-232, USB port*
- Internet connection, external web services or data

[16] When dealing with a software application

[17] Not so much an issue when dealing with disk images of software

These requirements represent the legacy setup. They depend on the way the software was used in the past, particularly the requirements for the peripheral hardware such as the monitor type and resolution and non-standard input and output devices. Artworks or video games are examples that might need very specific peripherals or hardware set up for authentic display. On the other hand, for a demonstration of a software such as QuarkXpress, the system requirements for the host computer suffice.

2.3 Challenges and Remarks

As is apparent from the above examples, for both operating systems and software applications, the lack of standardisation when it comes to detailing system requirements presents a challenge for institutions looking to capture this information in a structured way. Using the above recommendations is a good starting point to focus on and see how it relates to specific software within a collection. As software, in particular games and other graphically intensive titles, continue to improve the system requirements may expand to include additional hardware resources such as video and sound cards.

When compiling system requirements for software, particularly obsolete or unsupported titles, it can often be a challenge to locate a trusted source for such information. The best case scenario would be to reference the original software documentation or packaging. Where this is not possible, look to official sources online. Some companies will maintain this type of information on their own website. Otherwise, and as was the case in this research, there are often secondhand providers of such information. There are many websites that have captured and made available the system requirements for common commercial software. In the above examples some of these websites are referenced but it should be noted that the process consisted of examining and cross checking multiple sources for the same title, resulting in a more reliable and thorough outcome.

Another challenge facing the documentation of software is the dual nature of system requirements, often including both a set of minimum requirements and recommended requirements. In general, the inclusion of both sets has the potential to create information redundancy. While hardware and resource considerations may need to balance cost and performance for newer titles, it is likely that the rendering of old software will rely on specific emulators on top of contemporary hardware that easily outperforms the original requirements. For this reason it makes sense to aim to capture the recommended system requirements rather than minimum.

3. Metadata models for system requirements

This chapter will first evaluate the PREMIS dictionary as a model for describing system requirements. Using a dictionary such as PREMIS creates clarity, because the metadata fields have been carefully defined. PREMIS also has potential to describe relationships between various elements which can clearly display environment hierarchies. By looking at a specific use case the chapter will illustrate how to go about deciding which PREMIS fields to use software and software environments. Another element of structuring software metadata is to look at standard vocabularies for the *contents* of the metadata fields. The second half of this chapter will look at Wikidata and the role that it can play in describing software.

3.1 PREMIS for system requirements

3.1.1 Introduction to PREMIS and computing environments

The PREMIS Data Dictionary offers a comprehensive resource for institutions looking to implement preservation metadata within their digital preservation systems[18]. Since 2005, the standard has seen several major iterations, the latest being version 3.0. A significant shortfall of previous versions of PREMIS was how it dealt with computing environments. Environment information is critical to understanding how a specific file or object can be accessed or rendered correctly. Until PREMIS version 3.0, the environment semantic unit was rarely used[19]. Changes to what an environment actually constitutes within PREMIS and how relationships can be expressed between different environment components has made it much more viable for institutions to document and preserve this information to be reused into the future.

PREMIS makes use of semantic units which, in turn, can be linked to 4 distinct entities. An entity is one of the core components with regard to digital preservation activity and, for PREMIS, they include Objects, Events, Agents and Rights. For the purpose of this research, and the description of computing environments, the Objects entity is a prime focus. Objects can further be subdivided into four subcategories, which are: Intellectual Entity, Representation, File, and Bitstream. An Intellectual Entity is a distinct intellectual or artistic creation that is considered relevant to a designated community in the context of digital preservation, such as a particular book, map, photograph, database and even hardware and software.

[18] http://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf pg. 1
[19] https://www.researchgate.net/publication/262280940_Describing_and_Preserving_Digital_Object_Environments

3.1.2 System requirements within PREMIS

PREMIS 3.0 offers an approach where each layer of this software/hardware stack can be described as individual Intellectual Entities and linked to a specific non-environment object[20]. A non-environment object can range from a target file that is the focus of a rendering attempt, to a specific representation of a necessary operating system or piece of software that is stored as a disk image or file executable[21]. When linked these can point directly or indirectly to the three key Intellectual Entities that constitute an environment, as seen in the below diagrams.

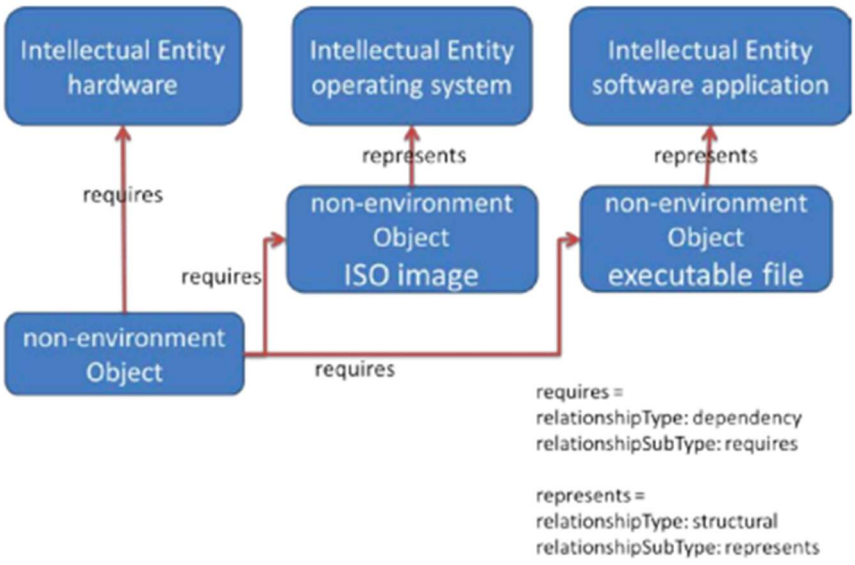


Figure 1: An object and its rendering environment: direct link (source: PREMIS Data Dictionary)

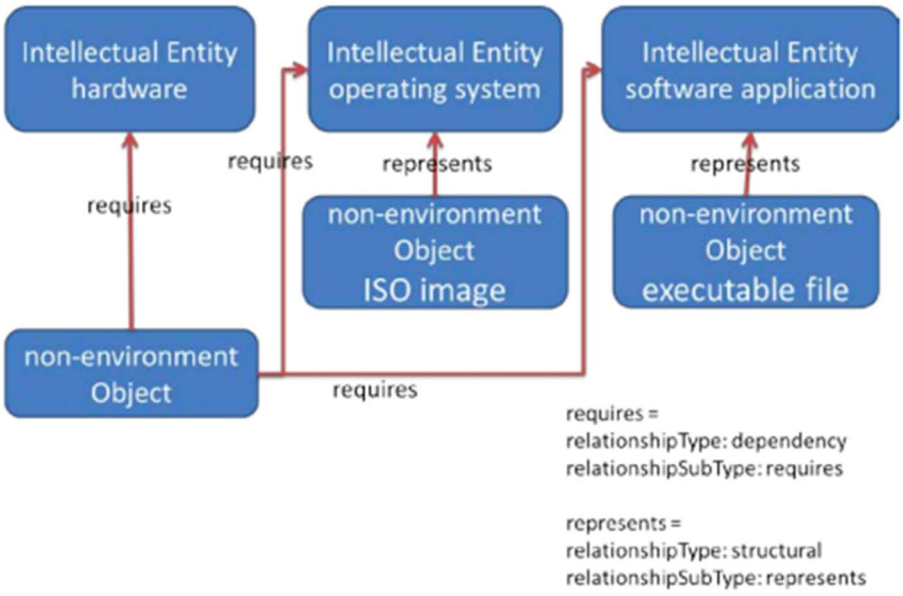


Figure 2: An object and its rendering environment: indirect link (source: PREMIS Data Dictionary)

[20] https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf pg. 251
[21] https://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf pg. 251

For the purposes of this investigation, the primary semantic units of interest are those that relate to the Object and the Environment. Using specific use cases, we will attempt to map the necessary information to the above diagram, whereby the non-environment object is linked to three distinct Intellectual Entities. It should be noted that due to the level of complexity involved in documenting all possible information for each entity, including system requirements, that the most efficient model for collecting and maintaining environment metadata is a centralized registry (or series of registries)[22]. First, we will look at the key semantic units that relate to Objects and Environments and how specific examples might fit within each.

3.1.3 Use case: QuarkXpress

The use case we will take a look at is that of a QuarkXpress Datafile and its associated environment stack. The below tables detail the three Intellectual Entities that an example data file can be linked to. In each case there are Object semantic units and, where applicable, the Relationship semantic units[23]. In this example, the environmentDesignationNote semantic unit is used to provide access to an external source which details the system requirements. Another option for this field is to provide such information in an unstructured free text way.

Software Application

Semantic Unit	Value
1.1 objectIdentifier	
1.1.1 objectIdentifierType	local
1.1.2 objectIdentifierValue	002
1.2 objectCategory	intellectual entity
1.9 environmentFunction	
1.9.1 environmentFunctionType	software
1.9.2 environmentFunctionLevel	1
1.9 environmentFunction	
1.9.1 environmentFunctionType	software application
1.9.2 environmentFunctionLevel	2
1.10 environmentDesignation	
1.10.1 environmentName	QuarkXpress
1.10.2 environmentVersion	8.5
1.10.3 environmentOrigin	Quark Inc
1.10.4 environmentDesignationNote	Documentation at http://files.quark.com/download/documentation/QuarkXPress/8.5/English/QXP_8.5_ReadMe_en-us.pdf

[22] Data Dictionary for Preservation Metadata: PREMIS version 3.0, p. 255 http://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf

[23] Examples were mapped using the PREMIS 3.0 handbook and the examples given at https://www.loc.gov/standards/premis/examples.html They do not represent a full PREMIS record but more a look at the specific semantic units relating to environments.

1.10.5 environmentDesignationExtension	Possible link to external registry
1.13 relationship	
1.13.1 relationshipType	dependency
1.13.2 relationshipSubType	requires
1.13.3 relatedObjectIdentifier	
1.14.3.1 relatedObjectIdentifierType	local
1.14.3.2 relatedObjectIdentifierValue	003
1.14.3.3 relatedObjectSequence	
1.13.4 relatedEventIdentifier	
1.13.5 relatedEnvironmentPurpose	execution
1.13.6 relatedEnvironmentCharacteristic	

Table 3: Semantic units related to the software application.

Operating System

The following table describes two different operating systems for the installation of QuarkXpress 8.5: an Apple Macintosh version and a Windows version. Which one is chosen depends on the available hardware and the use of the software. Other possible operating system versions can be mentioned in environmentDesignationNote. This field (semantic unit) can also be used to describe what this setup was or is used for.

Semantic Unit	Value
1.1 objectIdentifier	
1.1.1 objectIdentifierType	local
1.1.2 objectIdentifierValue	003
1.2 objectCategory	intellectual entity
1.9 environmentFunction	
1.9.1 environmentFunctionType	software
1.9.2 environmentFunctionLevel	1
1.9 environmentFunction	
1.9.1 environmentFunctionType	Operating system
1.9.2 environmentFunctionLevel	2
1.10 environmentDesignation	
1.10.1 environmentName	Mac OS
1.10.2 environmentVersion	10.4.11

1.10.3 environmentOrigin	Apple Inc
1.10.4 environmentDesignationNote	Other versions: 10.5.8 and 10.6.4
1.10.5 environmentDesignationExtension	Possible link to external registry
1.10 environmentDesignation	
1.10.1 environmentName	Windows XP
1.10.2 environmentVersion	Service Pack 3
1.10.3 environmentOrigin	Microsoft
1.10.4 environmentDesignationNote	Other versions: Windows Vista 1.0 and Windows 7
1.10.5 environmentDesignationExtension	Possible link to external registry
1.13 relationship	
1.13.1 relationshipType	dependency
1.13.2 relationshipSubType	requires
1.13.3 relatedObjectIdentifier	
1.14.3.1 relatedObjectIdentifierType	local
1.14.3.2 relatedObjectIdentifierValue	004
1.14.3.3 relatedObjectSequence	
1.13.4 relatedEventIdentifier	
1.13.5 relatedEnvironmentPurpose	execution
1.13.6 relatedEnvironmentCharacteristic	

Table 4: Semantic units related to the operating system.

Hardware

The following table describes two different hardware setups for the installation of QuarkXpress 8.5: an Apple Macintosh computer and a Windows PC. Which hardware is chosen depends on the available hardware, operating system and the use of the software. In this instance the environmentDesignationNote has been used to document the specific hardware components that are necessary for this environment.

Semantic Unit	Value
1.1 objectIdentifier	
1.1.1 objectIdentifierType	local
1.1.2 objectIdentifierValue	004
1.2 objectCategory	intellectual entity
1.9 environmentFunction	

1.9.1 environmentFunctionType	hardware
1.9.2 environmentFunctionLevel	1
1.9 environmentFunction	
1.9.1 environmentFunctionType	hardware architecture
1.9.2 environmentFunctionLevel	2
1.10 environmentDesignation	
1.10.1 environmentName	Macintosh Computer
1.10.2 environmentVersion	n/a
1.10.3 environmentOrigin	Apple Inc
1.10.4 environmentDesignationNote	G5 or faster PowerPC processor or Mac Intel processor, 1GB RAM (256MB minimum), 2GB hard disk space.
1.10.5 environmentDesignationExtension	Possible link to external registry
1.10 environmentDesignation	
1.10.1 environmentName	Windows PC
1.10.2 environmentVersion	n/a
1.10.3 environmentOrigin	Microsoft
1.10.4 environmentDesignationNote	1GB RAM (256MB minimum on Windows XP, 512MB minimum on Windows Vista), 1GB hard disk space.
1.10.5 environmentDesignationExtension	Possible link to external registry

Table 5: Semantic units related to the hardware environment.

3.1.4 Conclusions and recommendations for the use of PREMIS

Although PREMIS offers a structured and controlled way of describing specific versions of software and hardware, it is still unclear how to approach these descriptions on a far more granular level. In many of the examples explored, the detailed system requirements information for software, operating system and hardware were documented through linked eternal registries through the environmentDesignationExtension semantic unit. Where PREMIS succeeds is in its ability to document the hierarchical nature of a computing environment and the relationships between each level. Though specific system requirements information can be included in free text fields such as environmentDesignationNote, this may not always be the most sophisticated option. Where possible, it is always worth looking into controlled vocabularies for this form of data and, as appears to be the favourable option within PREMIS, create or source an external registry that can hold this information. First and foremost, institutions and implementers need to make sure that their computing environment information, including system requirements, is correct. From that point the level of description and granularity that is actually required of the PREMIS record needs to be explored. It is perfectly acceptable for PREMIS to capture environment components and relationships at a relatively high level. For further information relating to specific system requirements there are multiple registries and sources available that can potentially help. Pronom and Wikidata both offer software and hardware registries that offer a linked data solution for PREMIS. The following section will take a look at the latter example, Wikidata.

3.2 Wikidata for system requirement

3.2.1 Introduction to Wikidata and preservation metadata

Linked Open Data is more and more often used for the documentation of cultural heritage. It creates relationships between objects in different knowledge bases and enables the aggregation of data across collections and institutions. An important example for Linked Open Data is Wikidata: a project by the Wikimedia Foundation which is best known for Wikipedia, a multilingual and collaborative encyclopedia. Wikidata provides structured data, initially mostly to facilitate multilinguality on Wikipedia articles, but Wikidata is being used in an increasing number of cases that are not directly related to Wikipedia. Wikidata, just like Wikipedia, is an open knowledge database which means everyone can use it and contribute to it. This makes the barrier for entry low for archives.

For many years Linked Data was mainly used for descriptive metadata such as the use of Linked Data on the Europeana platform[24]. Wikipedia pages about video games and software mainly contain descriptive metadata such as the creator of the game or developer of software and the creation date and type of game or software. In 2017, Thornton et al. made us aware that Wikidata not only contains descriptive metadata, but also preservation metadata. They created a very useful online tool presenting both descriptive and preservation metadata of a Wikidata item: https://wikidp.org/.

System requirements are a subset of preservation metadata. In this report, they are understood as the "original" setup, the hardware and software that was required to run software at the time of its publication. As described above, these system requirements can sometimes be hard to find and judge on their quality due to the variety of sources. Wikidata offers the potential to centralise the collection of such information. Wikidata already contains numerous structured descriptions of software and hardware items with detailed information about its development and application.

Each Wikidata item has a unique identifier, a number that starts with a capital Q. Each item has properties, which are uniquely identifiable by a number starting with a capital P. These properties or semantic units can be browsed in the Wikidata property explorer[25]. A statement about a Wikidata item, a so-called information triple, consists of one Wikidata item, one property and one value (see following example). The value can consist of another Wikidata item, a string, or other data formats[26].

Wikidata item (subject)	Property / semantic unit (predicate)	Value (object)
Firefox (Q698)	instance of (P31)	web browser (Q6368, Wikidata item)
Firefox (Q698)	software version identifier[27] (P348)	10.0.6esr (string)
Firefox (Q698)	platform (P400)	x86_64 (Q272629, Wikidata item)

Table 6: Three examples of statements in Wikidata.

[24] For instance Linked open data in Europeana: https://pro.europeana.eu/page/linked-open-data accessed June 2021
[25] Wikidata property explorer: https://prop-explorer.toolforge.org/ accessed June 2021
[26] Data types of Wikidata properties: https://www.wikidata.org/wiki/Help:Data_type accessed June 2021
[27] The triple "Firefox (Q698)" "Software version identifier (P348)" "10.0.6esr" means that Firefox has version 10.0.6esr. As Firefox has multiple versions, the statement has to be iterated for each version.

As Thornton et al. (2017) missed certain preservation metadata properties, they investigated and described how properties in Wikidata[28] are created. This process is collaborative and prevents the generation of overlapping or same properties. It also encourages the consistent use of properties. On the other hand, this process slows down the creation of new properties, which needs to be factored in when using Wikidata. The following section will explore system requirements as a subset of preservation metadata in Wikidata.

3.2.2 System requirements as Wikidata properties

This chapter maps the system requirements identified in chapter 2.2 on the Wikidata properties. It lists Wikidata properties that fully match the identified system requirements, Wikidata properties that are partly matching with these requirements and finally properties that are needed to describe system requirements but are still missing from Wikidata.

Matching system requirements

System requirement / semantic unit	Wikidata property / semantic unit	Comments
Dependency on software platform / library / runtime	P1547 Depends on software	The triple “software A” “depends on software” “software B” means that software A requires software B to run.
Operating system	P306 Operating system	The triple “software A” “Operating System” “Operating system B” means that software A requires Operating System B to run.
Computer architecture / CPU / instruction set	P1068: Instruction set (instruction set on which the processor architecture is based) P880: Central processing unit (CPU)	
CPU performance	P2149: Clock speed	In combination with P880 this property describes the CPU performance.
External Storage Device Type	P4788 Storage device	The triple “software A” “Storage Device” “Floppy disk” means that software A requires storage device “Floppy disk”.
Sound interfaces	P7501 Audio system	In example “Surface Studio” (Q27580119), first desktop PC by Microsoft, this property has the values: “loudspeaker”, “stereo”, “phone connector”, “microphone”. This describes rather input and output devices. However, in relation to system requirements, ”audio system” could mean a specific sound card or a sound card with specific features such as a specific bit depth, sampling frequency, number of input and output channels. These single features (properties) do not exist yet in Wikidata.

[28] Thornton K, Cochrane E, Ledoux T, Caron B, Wilson C (2017) Modeling the Domain of Digital Preservation in Wikidata. In: ipres 2017 (ed) Proceedings

Graphics processing unit (GPU)	P2560 Graphics processing unit (GPU) P6948 Video system	Often, not a specific brand is required, but a certain compatibility with a runtime/API (for instance Open GL, DirectX, Vulkan etc.) or a specific feature (output port, video resolution, performance). The runtime / API depends on the OS. It can be described as follows: "GPU A" has "instruction set" "GPU API", for instance: "Radeon HD5750" has "instruction set" "OpenGL" "version 4.4" (qualifier "software version identifier") For video games with specific hardware property P6948 (video system) can be used.

Table 7: System requirements matching system requirements in chapter 2.2

Partly matching Wikidata properties

System requirement / semantic unit	Wikidata property / semantic unit	Comments
Minimum Free Storage Space	P3575 Data size (size of a software, dataset, neural network, or individual file)	The data size of the software might be smaller than the required minimum disk space. It would be better to have an additional property "minimum installation space"
Minimum RAM	P2928 Memory capacity	Memory capacity is usually maximum capacity. Hence, a property "minimum RAM" would be useful.
Minimum Internet Speed	P6711: data transfer rate (transfer speed through a bus or a communication medium)	Data transfer rate is not limited to the internet. If it was used with the (missing) property "internet access required" P6711 could be used as a qualifier.
Peripherals: Input Device Type	P479: Input method P2935: Connector P9192: Sensors	Wikidata does not differentiate between standard and non-standard peripherals. Instead it uses input method and output method. Input method is much broader than just the mouse and keyboard. An input method can also encompass microphones and other sensors. And it can mean input through user interface or input through command line. The latter use of "Input method" is not a system requirement[29]. If non-standard, the connector of the input device can be described with P2935.

[29] In appendix 3.1, example Adobe Flash (Q165658), "input method" (with the values "graphical user interface", and "script") is *not* used as a system requirement. In appendix 3.2, example World of Warcraft (Q131007), "input method" (with the values "keyboard" and "mouse") is used as a system requirement.

		For sensors the property P9192 should be used.
Peripherals: Output device	P5196: Output method P2935: Connector	Wikidata does not differentiate between standard and non-standard peripherals. Instead it uses input method and output method. Output method is much broader than just the monitor or projector. It can also encompass a printer, sound, and other sensors. And it can mean that a program uses a standard output or produces a file output. The latter use of "Output method" is not a system requirement. If non-standard, the connector of the output device can be described with P2935.

Table 8: System requirements partly matching with system requirements as listed in chapter 2.2

Missing Wikidata properties

At this moment, Wikidata does not provide any properties for the following metadata fields:

- Required Machine Type (Required type of computing machine necessary to operate software, for instance laptop, mobile phone, electronic device) [Wikidata item]
- Display Resolution vertical (number of pixels) [number]
- Display Resolution horizontal (number of pixels) [number]
- Internet Access Required [yes or no][30]
- Minimum installation space in MB [number]
- Minimum RAM in GB [number]

3.2.3 System requirements of Wikidata items

This section analyses how system requirements are described for individual Wikidata items today (May 2021) and how this could be improved upon. Appendix 3 contains examples of Wikidata items such as software, video games and operating systems. It lists their Wikidata system requirements and other preservation metadata expressed as Wikidata properties. These other preservation metadata are listed to provide an impression of what preservation metadata are collected in Wikidata, how they are used and to give context to the system requirements. The following table provides an example how system requirements of the Firefox web browser[31] are documented in Wikidata:

[30] There is also no Wikidata property "requires" that could be used in a triple like: "Software A" "requires" "internet"

[31] https://wikidp.org/Q698/preview accessed 2021/05/11

Wikipedia item (subject)	property (predicate)	object / value
Firefox (Q698)	Operating system (P306)	Linux (Q388), Microsoft Windows (Q1406), macOS (Q14116), Firefox OS (Q550303), Android (Q94), iOS (Q48493), FreeBSD (Q34236)
Firefox (Q698)	Platform (P400)	IA-32 (Q262238), x86_64 (Q272629), ARM (Q218864)
Firefox (Q698)	Depends on software (P1547)	Yasm (Q2547156), Netscape portable runtime (Q3509073), … (about 30 entries)

Table 9: Properties of Firefox specifying system requirements for Firefox

The problem with the Wikipedia entry of Firefox (Q698) above is that the properties are not version specific, they apply to all versions at once. However, system requirements differ from version to version: For instance, Firefox 3 requires Windows 2000 or Windows XP, Firefox 90 requires Windows 10. The following table shows, how Firefox versions are described in Wikidata:

Wikipedia item (subject)	property (predicate)	object / value	qualifier
Firefox (Q698)	P348: Software version identifier	Versions 1 to 90 are listed as strings. The versions are not Wikidata items (they do not have a unique identifier)	Each version is described with several qualifiers such as P577: publication date P548: version type
Firefox (Q698)	P747: Has edition or translation	Firefox Nightly (Q56316099), Firefox 3.6 (Q2615631)	

Table 10: Properties of Firefox specifying Firefox versions

The software versions listed under P348 (software version identifier) cannot take Wikidata items as a value, only strings. Hence, software versions entered in P348 are not linked to Wikidata items and therefore cannot be described individually. In contrast, P747 (has edition or translation) lists software versions with their own Wikidata identifier. In the case of Firefox, over 90 versions are described with P348 (software version identifier) while P747 (has edition or translation) describes just two versions. Consequently, first, the Wikidata pages for the single Firefox versions (or groups of versions with the same system requirements) would have to be created before the system requirements could be added in a useful way.

In the examples mentioned in appendix 3.1, except for Firefox, none of the described softwares (Firefox, Adobe Flash, Adobe Flash Player, yEd, QuarkXpress) and video games (Commodore 64 Games System, Rollerboard, 3D Tanx, Menace, Grachten Racer, World of Warcraft) specified software versions with separate unique identifiers (using the property P747, “has edition or translation”). While certain video games might not have software versions as they are one entity with their hardware, the mentioned softwares have many versions and hence version dependent system requirements. Thornton et al. 2017 described a similar problem where one Wikidata item represents several semantic entities: In their case, it was not several software versions represented by one Wikidata item, but a software and its associated file format represented by a single Wikidata item.

This leads to the question: what kind of software objects does Wikidata (potentially) describe. Information is only eligible for Wikidata if it qualifies according to the notability guidelines of the platform[32]. One such criterion is for the object to have an equivalent page on Wikipedia or one of the other Wikimedia projects. For an object or topic to have its own Wikipedia page a topic needs to have received "significant coverage in reliable sources that are independent of the subject". Since this is not usually the case for specific versions of software, it is unlikely that these would make it onto Wikipedia, and by extension to Wikidata. Also, non-commercial and not widely distributed software objects created by researchers, artists, designers or small game producers are not described in Wikipedia and their description might not be desirable from a Wikipedia point of view. Therefore, Wikidata might have its limitations when it comes to covering software objects that have limited notability. The criteria also state however that information is also permitted if it "fulfills a **structural need**, for example: it is needed to make statements made in other items more useful." This seems to create some space for the type of information we are here concerned with.

A related question is, how specific one can get with the description of software objects. A software object might be part of a specific setup that an archive wants to preserve, whether it be an artwork installation or a specific use of a software. Hence, **the system requirements for a specific use of a software might differ from the generic system requirements** of this software object. This is very difficult if not impossible to describe with Wikidata. First, such a specific setup most likely does not have its own Wikidata item. Secondly, even if it had, the specific system requirements of a software would contradict the generic ones. It could be argued that such a contradiction could be prevented by using qualifiers. However, the first qualifier might be used to qualify a system requirement as in the following hypothetical example, but Wikidata only allows one qualifier level. Nested qualifiers are not possible.

Wikipedia item (subject)	property (predicate)	object / value	qualifier	value
Firefox (Q698)	Internet access required (Property does not exist yet)	[yes or no] value does not exist for Firefox in Wikidata	P6711: Data transfer rate	value does not exist for Firefox in Wikidata

Table 11: Wikidata system requirements using a qualifier. It is not possible to add another qualifier level to specify requirements for a specific use case (or a specific Firefox version).

Another challenge of Wikipedia/Wikidata is the **consistent use of properties** for different software objects. For instance, it is not clear when to use the property "platform" (P400), when to use the property "operating system" (P306) and when to use the property "depends on software" (P1547). In Firefox (Q698, see table 9 and WindowsXP (Q11248, see appendix 3.3) "platform" is used to describe computer architectures. In Adobe Flash Player (Q857177) "platform" is filled in with "Microsoft Windows", hence an operating system. For yEd, a graphics software, "platform" is filled in with "Java Virtual Machine" which could instead be described as "depends on software" (P1547). Conventions such as not to use the property "platform" (P400) or "instruction set" (P1068) if an operating system is specified that implies a certain computer architecture would support a consistent use of properties.

[32] https://www.wikidata.org/wiki/Wikidata:Notability

A similar issue poses the **description of relationships, hierarchies and structures** between Wikidata items. There are many similar properties defining relationships and it is sometimes difficult to know which properties to use or search for. The relationship class - sub-class (P279) is used to describe the hierarchies between classes (a “web browser” is a subclass of “application”). Instance of (P31) means that a Wikidata item is an instance of a certain class (“Firefox” is an instance of the class “web browser”). The property ”has edition or translation” (P747) can be used if a software has a version (“Firefox” has edition or translation “Firefox 3.6”). The consistent use of these relationships depends on the users.

If a search result does not provide the expected results, it is not always because the search did not encompass all the possible (correctly or incorrectly used) properties. Often, the **relationships between software versions are not described at all**. For instance, Windows XP (Q11248) has several editions and service packs. When querying Wikidata for all the instances (P31) and (or) sub-classes (P279) and/or editions (P747) of Windows XP, only two editions come up:Windows XP Professional x64 Edition (Q245793) and Windows XP 64-Bit Edition (Q6072277). All other editions described in Wikipedia such as Windows XP Home (Q26161904), Windows XP Media Center Edition (Q2643528) or Windows XP Starter (Q10393871) are not linked to Windows XP (see also appendix 3.3).

Finally, when going through the Wikidata entries of the video game examples in appendix 3.2, especially the ones running on a Windows computer, it becomes clear how **little preservation metadata is filled in** and how much is still missing. Information about the necessary video cards, graphics APIs or video card performance is missing in all the examples. Also requirements for display equipment or joysticks cannot be found.

3.2.4 Conclusions and recommendations for the use of Wikidata for system requirements

Even though Wikidata has potential for the description of system requirements, using it in practice would still be very experimental at this stage. The description of most software items is incomplete, be it on a structural level (creation of Wikidata items, relationships between Wikidata items, consistent use of properties) or content (values for system requirement properties) level. From the section above it becomes clear that the data structure of Wikidata is a work in progress and still needs a big and continuous effort for preservation metadata, not only to describe new software, but also for very commonly used software of the past. It would need a big effort to complete Wikipedia entries, introduce new properties and create new Wikidata items (separate Wikipedia pages) for software versions.

Thornton et al. 2017 explain the collaborative processes to adapt and develop the data structure of Wikidata. When working with Wikidata, these collaborative processes need to be taken into account and a certain flexibility timewise but also regarding the data model is a precondition. The great advantage of using Wikidata are the synergies generated for the whole software preservation and Wikipedia community in terms of knowledge production and access (structuring and sharing software preservation metadata). Finally, the use of Linked Data allows to adapt quickly to changing semantics of the technical and descriptive preservation vocabulary.

The biggest question mark is to which degree Wikidata would accept the creation of separate pages for software versions. Thornton et al. 2017 mention that Wikipedia has different objectives than the preservation community and that these might be incompatible. While any user can create a new Wikidata item, for instance for a specific version of a software application, it is the Wikipedia community that ultimately decides whether the information contributed fulfills the requirements of notability *or* whether it fulfills a structural need as described above. Using Wikidata for preservation metadata might introduce a level of uncertainty which archives are not willing to accept.

If it is not possible or desirable to use Wikidata, it is an option to just use the Linked Data software of Wikidata, called Wikibase[33], to create one's own data model. Rhizome, an institution collecting netart in New York, can serve as an example. They created their own preservation metadata model[34], focussed on the preservation of netart and their collection, as they were not able to adapt the Wikidata model to their needs. In this way, they can still link certain properties (semantic units) and items (values) to Wikidata, but they have the full freedom to apply their own data model.

[33] Homepage of the wikibase software: https://wikiba.se/ accessed June 2021
[34] Rhizome's artbase data model: https://sites.rhizome.org/artbase-re-design/data-models.html accessed June 2021

4. Conclusion and Recommendations

In this report we explored the ideas and solutions that can be found with regards to metadata for software preservation. It is a fragmented field, which can be difficult to navigate. Software preservation doesn't happen at such a scale that best practices have already been fully established. The most comprehensive way to describe software would be to include in the description information about the environment that can run the software in the present. If the software is somewhat older, this is likely to be an emulation environment. However, emulation environments will change over time, so it will require a lot of monitoring and adjusting metadata to keep track of developments. This effort is only justified if an archive wants to provide the highest service level with regards to providing access to its software collection. In this report we have instead focused on describing the system requirements that were provided at the time of publication. These will remain the same and can always be used to construct an (emulation) environment for access.

The PREMIS metadata dictionary offers possibilities for describing software, software versions and the relations they have with their environment. However, the level of detail that can be provided in PREMIS in a structured way is limited. It is therefore necessary to explore the possibilities for referencing external registries that do include such detail.

We looked at Wikidata to see to which degree it can function as such an external registry. Although Wikidata offers a lot of the functionality that one might need, it would at present be quite an experimental endeavour to use Wikidata in practice. A lot of the information needed is probably missing, and adding it to the database comes with a level of uncertainty about the degree to which the community would accept such specific information. Also, the relational properties have not been carefully defined and are therefore used in inconsistent way.

Wikibase, the database software on which Wikidata is based, can be set up as a separate instance: including Wikidata information where desired, but also giving the hosting organization full control over which information it contains.

For every archive that is looking to start the preservation of software, a careful analysis is required of their specific context and the purposes they have with software preservation.The appendices that have been included are there to aid archives to dive deeper into the subject matter and make their own choices based on their needs. In this report we explored the ideas and solutions that can be found with regards to metadata for software preservation. It is a fragmented field, which can be difficult to navigate. Software preservation doesn't happen at such a scale that best practices have already been fully established.

Appendices

Appendix 1: Metadata checklist for software preservation

Minimum metadata for bit preservation are marked with a cross (x)

Discovery (descriptive metadata)

- x Description of game or software (what does it do?)
- x What is the name / alternative name of the software?
- What was the software used for?
- Who used it?
- When was it popular?
- When was it developed?
- Who developed it?
- What version is it? Potential reference or link to versioning control system repository or to existing software repository.
- Part of a package or series
- FRBR[35] hierarchy: *Work* (is realised through) *Expression* (is embodied in) *Manifestation* (is exemplified by) *Item*
- Replaces, supersedes other software

Behaviour / Significant properties (preservation metadata)

- "Original" behaviour: What does / did it do? What does / did it look and feel like? Description of interactivity, behaviour, functionality.
- Significant properties are characteristics of an object that should be maintained through preservation actions. Which properties of the "original behaviour" have to be maintained? Which limitations of functionality / interactivity / behaviour are accepted (editable, read-only, fully or partly functional etc.)?
- Necessary I/O hardware (for instance mouse, joystick, type of keyboard, touchscreen, printer, speakers, etc.)
- Documentation of significant properties, for instance with screenshots and screencasts. This field can refer to the user instructions in "Object structure and technical description → Ancillary material"
- What knowledge does the user need to interact with the object? This field can refer to the user instructions in "Object structure and technical description → Ancillary material"
- Does the object change through interaction with the user? Will each user session be reset or can the user save this state?
- Does the user have to be able to import and export data?

Provenance, chain of custody (preservation metadata)

- Original media where it was submitted on
- Who submitted?
- When was it submitted?

[35] https://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records

- Submission context? (For instance “is part of collection” A)
- Derivatives (tree of derivatives, is parent of / is child of)
- How were the derivatives produced (preservation actions / events → refer to registry of preservation actions)
- x Last modified date
- Additional description of changes of derivatives, in particular in the case of snapshots of a virtual machine / emulation (why was the snapshot made, what has changed compared to the previous snapshot)

Object structure and technical description (preservation metadata)

- Structure of composite objects (parts, how are they linked, subsystems)
- x Size of (composite) object
- x File format of input and output files of software (reference to file format registries such as PRONOM)
- File formats of input and output files of a software (reference to file format registries such as PRONOM)
- Functional description of important files or folders. For instance whether a file is a source code or an executable, installer or package
- Ancillary material from the software vendor or producer like user manual and installation instructions. It is linked to the software object for instance by the relationship “is documented by”.
- Ancillary material produced by the heritage institution: user instructions given by the heritage institution, documentation of significant properties (screenshots, screencasts).It is linked to the software object for instance by the relationship “is documented by”.
- Other ancillary material

Dependencies (preservation metadata, representation metadata)

- x Required computing hardware (system requirements)
- x Required software environment (system requirements)
- Other dependencies (data sources, protocols etc.)
- Required emulator
- Configurations

Administrative

- x Unique identifier
- Rights management (software licenses)
- Internal and external access management
- x Monitoring (bit preservation, checksum, checksum type)
- Preservation planning
- Quality control after preservation actions (functional tests, significant properties)
- Documentation of preservation actions / workflow (s. also “Provenance, chain of custody”)

Appendix 2: References software preservation metadata

This Appendix contains a list of literature and other resources that will hopefully help readers to dive further into topics that are more specific to their usecase. Software preservation can take many forms, and the requirements for metadata will take shape according to the context in which preservation happens.

Digital preservation metadata

- Brown, Adrian (2013): Practical Digital Preservation : A How-to Guide for Organizations of Any Size. Chapter about metadata p.155 ff
- Dappert A, Guenther RS, Peyrard S (eds) (2016) Digital Preservation Metadata for Practitioners: Implementing PREMIS. Springer, Cham
- The DPC Digital Preservation Handbook, chapter about "Metadata and Documentation" https://www.dpconline.org/handbook/organisational-activities/metadata-and-documentation

Domain specific metadata (discovery, significant properties)

Software (mainly in the context of Science / Research)

- Software ontology tool: https://www.ebi.ac.uk/ols/ontologies/swo mainly (but not limited) to the bioinformatics community.
- CodeMeta: minimal metadata schema for science software and code, in JSON and XML: https://github.com/codemeta/codemeta
- Acquisition process of software: https://www.softwareheritage.org/swhap/
- Software citation principles: Smith AM, Katz DS, Niemeyer KE, (None) (2016) Software citation principles. PeerJ Computer Science https://peerj.com/articles/cs-86/
- Source code of software: https://schema.org/SoftwareSourceCode
- Software Application https://schema.org/SoftwareApplication

Video games

- GAMECIP The game metadata and citation project (2017) https://gamecip.soe.ucsc.edu/. This project created a controlled vocabulary for computer game platforms and their media formats. They can be accessed here:
  - List of computer game platforms (software and hardware): https://gamemetadata.soe.ucsc.edu/platform
  - List of media formats for computer games: https://gamemetadata.soe.ucsc.edu/media
- Wikidata: proposal to integrate the controlled vocabulary of GAMECIP in wikidata: https://www.wikidata.org/wiki/Wikidata:Property_proposal/GAMECIP_platform_ID
- Giovanni Carta, about significant properties of video games: "Metadata and video games emulation: an effective bond to achieve authentic preservation?" (2017) https://www.deepdyve.com/lp/emerald-publishing/metadata-and-video-games-emulation-an-effective-bond-to-achieve-meBq7kyR9F

Media art

- Media art research thesaurus: https://vocabularyserver.com/mediaart/
- Taxonomy of interactive art. Kwastek, Katja; Spörl, Ingrid (2009): http://www.kwastek.de/pdf/taxonomy_IA_200706.pdf

Objects that need software to be interpreted

- Spreadsheets: Veenendaal R. et al. (2019) Significant Properties of Spreadsheets: An update on the work of the Open Preservation Foundation's Archives Interest Group. In: iPres2019 (ed) iPres2019: Conference Proceedings, Amsterdam

Technical preservation metadata

- Metadata for disk images: Chassanoff A, Woods K, Lee C (2016) Chapter 8. Digital Preservation Metadata Practice for Disk Image Access. In: Dappert A, Guenther RS, Peyrard S (eds) Digital Preservation Metadata for Practitioners: Implementing PREMIS. Springer, Cham, pp 99–109
- Metadata for disk images: Ensom T., (2021): Disk imaging report - Tate, Time-based media conservation. https://www.tate.org.uk/about-us/projects/software-based-art-preservation
- Metadata for disk images:Meister (2017): Generate Filesystem Metadata as DFXML https://confluence.educopia.org/display/BC/Generate+Filesystem+Metadata+as+DFXML
- Metadata for emulation: Delve J, Anderson D (2012) The Trustworthy Online Technical Environment Metadata database - TOTEM. Kölner Beiträge zu einer geisteswissenschaftlichen Fachinformatik, vol 4. Kovač, Hamburg
- Description of Environments in PREMIS http://www.loc.gov/standards/premis/v3/premis-3-0-figures.pdf
- Identification of containers: Blog about how to create a container signature file (Spencer, 2016): https://openpreservation.org/blogs/droid-container-signature-files-what-they-are-and-how-to-create-them-a-template-and-an-example-or-few/
- Metadata model of the Emulation as a Service Infrastructure, Version5. Spreadsheet with detailed metadata: Bartczak, Jeremy. (2020, June 26). University of Virginia: EaaSI Metadata Model V5. Software Preservation Network. https://www.softwarepreservationnetwork.org/university-of-virginia-eaasi-metadata-model-v5/

Metadata standardisation

- Preservation Action Registry (PAR) https://parcore.org/
- Preservation Action Registry (PAR): Addis M, Simpson J, Tilbury J, O'Sullivan J, Stokes P (2018) Digital Preservation Interoperability through Preservation Actions Registries. In: iPres 2018 (ed) iPres 2018
- PRONOM file format registry: https://www.nationalarchives.gov.uk/PRONOM/Default.aspx
- PREMIS: Dappert A, Guenther RS, Peyrard S (eds) (2016) Digital Preservation Metadata for Practitioners: Implementing PREMIS. Springer, Cham
- PREMIS in combination with METS: Guenther R (2010) Metadata to Support Long-Term Preservation of Digital Assets: PREMIS and its use with METS. In: iPres 2010 (ed): Proceedings of the 7th International Conference on Preservation of Digital Objects, Vienna

Conceptual model for versions and instantiations

- IFLA Study Group on the Functional Requirements for Bibliographic Records (2009). Functional Requirements for Bibliographic Records. Series: IFLA Series on Bibliographic Control 19. Publisher: Munich: K.G. Saur Verlag, 1998. https://www.ifla.org/publications/functional-requirements-for-bibliographic-records
- Rossenova L, Espenschied D, de Wild K (2019) Provenance for internet art.: Using the W3C PROV data model. In: iPres2019 (ed) iPres2019: Conference Proceedings, Amsterdam, pp 297–304. https://osf.io/6xd4g/

Linked metadata

- Thornton K, Cochrane E, Ledoux T, Caron B, Wilson C (2017) Modeling the Domain of Digital Preservation in Wikidata. In: ipres 2017 (ed) iPres 2017: Proceedings
- Thornton K, Seals-Nutt K (2018) Wikidata for Digital Preservation. In: iPres 2018 (ed) iPres 2018
- Hooland S van, Verborgh R (2014) Linked Data for Libraries, Archives and Museums: How to clean, link and publish your metadata. Facet Publishing, London
- Wikidata. https://wikidp.org/ provides the digital preservation metadata that is registered in Wikidata about an object
- Specific information about the use of Wikidata for the domain informatics / software preservation https://www.wikidata.org/wiki/Wikidata:WikiProject_Informatics
- Data model of the Rhizome artbase (based on the software Wikibase, but with their own data model): https://sites.rhizome.org/artbase-re-design/data-models.html

Appendix 3: System Requirements described in Wikidata: Software examples.

Appendix 3 lists a few examples of softwares to demonstrate how the system requirements are described as Wikidata and what other preservation metadata Wikidata provides. Other preservation metadata are necessary to understand better the relationships between the software objects. Hence, structural metadata such as "subclass of (P279)" "instance of (P31)" "software versions (P348)", "has edition or translation (P747)","Topic's main category (P910)", "use (P366)", "commons category (P373)", "different from (P1889)", "followed by (P156)", "follows (P155)", "based on (P144)", and "part of the series (P179)" are listed in "other metadata".

Metadata that are important to provide the right software environment for a file such as readable (P1072) and writable (P1073) file format are also listed in "other metadata". To be able to create an environment, it can be helpful to know the "package management system (P3033)" of a software. This is included in "other metadata".

The examples are extracted from Wikidata on 11 May 2021. They are divided in software (appendix 3.1), video games (appendix 3.2) and operating systems (appendix 3.3)

Appendix 3.1 Wikidata examples software

Firefox (Q698)
https://wikidp.org/Q698/preview

System requirements
Operating system (P306): Linux (Q388), Microsoft Windows (Q1406), macOS (Q14116), Firefox OS (Q550303), Android (Q94), iOS (Q48493), FreeBSD (Q34236) Platform (P400): IA-32 (Q262238), x86_64 (Q272629), ARM (Q218864) Depends on software (P1547): Yasm (Q2547156), Netscape portable runtime (Q3509073), Network Security Services (Q7000904), libpng (Q838041), Hunspell (Q176291), … Expat (Q2074084) (about 30 entries)
Other metadata
Readable file format (P1072): Web Open Font Format, version 2 (Q18413771), Mork (Q6912474), JSON (Q2063), WebP (Q62617958), Ogg (Q188199), Vorbis (Q11885120), WebM (Q309440), VP9 (Q3815169), Opus (Q1466199), MP3 (Q42591). H.264/MPEG-4 AVC (Q212633), Cascading Style Sheets (Q46441), Brotli (Q18205644), gzip (Q283647), Writable file format (P1073): Free Lossless Audio Codec (Q27881556) Instance of (P31): news aggregator (Q498267), VoIP software (Q15614021), web browser (Q6368), FTP client (Q3503189), mobile browser (Q845620), free software (Q341) Software versions (P348): *summary: 1.0 to 90.0* Software engine (P408): Gecko (Q487834), Quantum (Q28457708), SpiderMonkey (Q1848400) Has edition or translation (P747): Firefox Nightly (Q56316099), Firefox 3.6 (Q2615631) Topic's main category (P910) Category: Firefox (Q8459937) Copyright license (P275): Mozilla Public License, version 2.0 (Q25428413), GNU General Public License (Q7603), GNU Lesser General Public License (Q192897) Package management system (P3033): dpkg (Q305892), RPM Package Manager (Q492650), Flatpak (Q22661286), Portage (Q908542)

Adobe Flash (Q165658):
https://wikidp.org/Q165658/preview

System requirements
Operating system (P306): Microsoft Windows (Q1406), Linux (Q388), macOS (Q14116)
Other metadata
Readable file format (P1072): FLA (Q28756039), Small Web Format (Q594447), FLV (Q27967488) Writable file format (P1073): FLA (Q28756039) Input method[36] (P479): graphical user interface (Q782543), script (Q30581237) Instance of (P31): application (Q166142), multimedia framework (Q1186471), software engine (Q2622299), vector graphics editor (Q1155404), computing platform (Q241317) Followed by (P156): Adobe Animate (Q4291129) Copyright license (P275) proprietary license (Q3238057) Use (P366): 2D animation software (Q23442338) Commons category (P373): Adobe Flash Topic's main category (P910): Category:Adobe Flash (Q8221662) Different from (P1889): Adobe Shockwave (Q1061837), Adobe Flash Player (Q857177)

Adobe Flash Player (Q857177)
https://wikidp.org/Q857177/preview

System requirements
Operating system (P306): BlackBerry Tablet OS (Q2160367), Android (Q94), Microsoft Windows (Q1406), macOS (Q14116), Linux (Q388), Solaris (Q14646), Chrome OS (Q79531), BSD (Q58636917) Platform (P400): Microsoft Windows (Q1406)
Other metadata
Readable file format (P1072): Small Web Format (Q594447) Software version identifier (P348): *various 32.x.x versions, but no older versions* Instance of (P31): media player (Q210337), browser extension (Q874411), multimedia framework (Q1186471) Use (P366): run-time system (Q1004415) Different from (P1889): Adobe Shockwave Player (Q28955963), Adobe Flash (Q165658) Copyright license (P275): freeware (Q178285)

yEd (Q2598628)
https://wikidp.org/Q2598628/preview

System requirements
Operating system (P306): cross-platform

[36] The way "Input method" is used here, is not a system requirement. But it can be used as a system requirement if it is used as a hardware requirement (as for instance in "**World of Warcraft**" where input method is "computer keyboard (Q250)" or "mouse (Q7987)")

Platform (P400): Java Virtual Machine
Other metadata
Readable File format (P1072): GraphML (Q1543325), Apache Ant (Q385970) Writable file format (P1073): GraphML (Q1543325), Portable Network Graphics (Q178051), Scalable Vector Graphics (Q2078), Graph Modelling Language (Q1543319), Trivial Graph Format (Q4051789) Instance of (P31) : freeware (Q178285), diagramming software (Q3307487), proprietary software (Q218616) software version identifier (P348): (Summary: versions 3.16 to 3.20) Use (P366): diagram (Q959962) Copyright license (P275): end-user license agreement (Q725920)

QuarkXPress (Q678036)
https://wikidp.org/Q678036/preview

System requirements
Operating system (P306): macOS (Q14116), Microsoft Windows (Q1406)
Other metadata
Readable file format (P1072): Quark Xpress Data File, version 9 (Q47524799), Quark Xpress Data File (Q51913355), Quark Xpress Report File (Q60371443), Quark Xpress Data File, version 10 (Q60371646), Quark Xpress Data File, version 6 (Q60372734), Adobe InDesign Document (Q59914742), QuarkXPress Document 3.1 (Q89344774), QuarkXPress Document 4 (Q89347372), QuarkXPress Project 7 (Q89777428), QuarkXPress Project 8 (Q89897874), QuarkXPress Project 9.1 (Q90559776), QuarkXPress Project 2015 (Q90801872), QuarkXPress Project 2016 (Q91226396), QuarkXPress Project 2017 (Q91322362), QuarkXPress Project 2018 (Q92204260), QuarkXPress Project 2019 (Q92744208), QuarkXPress Document 1-2 (Q100705816), QuarkXPress Document 3 (Q100706036), QuarkXPress Document 3.2 (Q100706066), QuarkXPress Project 9.2 (Q100706334), QuarkXPress Project 10.1 (Q100707279) Writable file format (P1073): Quark Xpress Data File, version 9 (Q47524799), Quark Xpress Data File (Q51913355), EPUB (Q475488), Quark Xpress Report File (Q60371443), Quark Xpress Data File, version 10 (Q60371646), Quark Xpress Data File, version 6 (Q60372734), QuarkXPress Project 7 (Q89777428), QuarkXPress Project 8 (Q89897874),QuarkXPress Project 2015 (Q90801872) Software version identifier (P348): 2015 Instance of (P31); software (Q7397), desktop publishing software (Q29364197) Use (P366): desktop publishing (Q188610) Copyright license (P275): proprietary license (Q3238057)

Appendix 3.2 Wikidata examples video games

Commodore 64 Games System (Q1115883) (video game console)
https://wikidp.org/Q1115883/preview

System requirements
-
Other metadata
Instance of (P31): model (Q10929058) Subclass of (P279): home video game console (Q17589470) Part of (P361): third generation of video game consoles (Q129758) Follows (P155): Commodore TV Game 2000K and Commodore TV Game 3000H (Q370911) Followed by (P156): CDTV (Q955368)

Commodore 64 (Q99775) (home computer)
https://wikidp.org/Q99775/preview

System requirements
Operating system (P306) Commodore BASIC V2 (Q1115899) CPU (P880): MOS Technology 6510 (Q378246) Memory capacity (P2928): 64 Has part (P527): primary memory (Q11140433), MOS Technology VIC-II (Q1639225), Commodore Datasette (Q767429), MOS Technology CIA (Q350467)
Other metadata
Instance of (P31) computer model (Q55990535) Follows (P155) Commodore VIC-20 (Q918232) Followed by (P156) Commodore 128 (Q1115919), Commodore 64C (Q12744958) Subclass of (P279) home computer (Q473708) Commons category (P373) Commodore 64 Topic's main category (P910) Category:Commodore 64 (Q7573297)

Rollerboard (Video game on Commodore 64 platform)
(https://www.regionaalarchiefalkmaar.nl/games)
not registered in Wikidata.

3D Tanx (Q55566) (Video Game on Commodore 64 computer from 1982)
https://wikidp.org/Q55566/preview

System requirements
Platform (P400): ZX Spectrum (Q23882)
Other metadata
Instance of (P31): video game (Q7889)

Game mode (P404): single-player video game (Q208850)

Menace (Q1749543) (Video Game on Commodore 64 computer from 1988)
https://wikidp.org/Q1749543/preview

System requirements
Platform (P400): Commodore Amiga (Q100047), Atari ST (Q627302), Commodore 64 (Q99775), DOS (Q170434)
Other metadata
Instance of (P31): video game (Q7889) Game mode (P404): single-player video game (Q208850) Distribution format (P437): floppy disk (Q5293)

Grachten Racer (Q13646402) (PC-Video game from 2000)
https://wikidp.org/Q13646402/preview

System requirements
Platform (P400): Microsoft Windows (Q1406)
Other metadata
Instance of (P31): video game (Q7889)

World of Warcraft (Q131007) (PC-Video game from 2004)
https://wikidp.org/Q131007/preview

System requirements
Platform (P400): Microsoft Windows (Q1406), macOS (Q14116) Input method (P479): computer keyboard (Q250), mouse (Q7987) Data size (P3575): 88.4[37]
Other metadata
Game mode (P404): multiplayer video game (Q6895044) Distribution format (P437): digital distribution (Q269415) Software version identifier (P348): 8.3, 9.0.2 Instance of (P31): video game (Q7889) Part of the series (P179): Microsoft Windows (Q1406), macOS (Q14116) Copyright license (P275): proprietary license (Q3238057) Commons category (P373): World of Warcraft Uses (P2283): ray tracing (Q619942)

[37] No unit is provided in the case of World of Warcraft. However, for the property "data size", not only a number, but also a unit needs to be provided in Wikidata, otherwise the entry is unclear.

Appendix 3.3: Wikidata examples operating system (Windows XP)

Windows XP (Q11248)

https://wikidp.org/Q11248/preview

System requirements
Platform (P400): IA-32 (Q262238), x86_64 (Q272629), IA-64 (Q916994)
Other metadata
Readable file format (P1072): Windows Portable Executable file format, 32-bit (Q47455466) Software version identifier (P348): 5.1.2600.5687 (only one version registered in Wikidata!) Instance of (P31): Operating system (Q9135) Topic's main category (P910): Category:Windows XP (Q7368007) Commons category (P373): Microsoft Windows XP Based on (P144): Windows 2000 (Q483881) Follows (P155): Windows Me (Q484892), Windows 2000 (Q483881) Followed by (P156): Windows Vista (Q11230) Part of the series (P179): Windows NT (Q486487) Subclass of (P279): Windows NT (Q486487), Microsoft Windows (Q1406) Edition or translation of (P629): Microsoft Windows (Q1406) Copyright license (P275): shareware (Q185534), commercial software (Q1340793), volume licensing (Q4016359), Microsoft Software Assurance (Q16251378)

Windows XP Professional x64 Edition (Q245793)

https://wikidp.org/Q245793/preview

System requirements
Platform (P400): x86_64 (Q272629)
Other metadata
Instance of (P31): Operating system (Q9135) Based on (P144): Windows Server 2003 (Q11246) Part of the series (P179): Windows NT (Q486487) Subclass of (P279) Windows XP (Q11248), Windows Server 2003 (Q11246) Edition or translation of (P629): Microsoft Windows (Q1406)

Windows XP 64-Bit Edition (Q6072277)

https://wikidp.org/Q6072277/preview

System requirements
-
Other metadata
Instance of (P31): Operating system (Q9135)

Subclass of (P279) Windows XP (Q11248) Edition or translation of (P629): Microsoft Windows (Q1406)

Windows XP Home (Q26161904)
https://wikidp.org/Q26161904/preview

System requirements
-
Other metadata
Instance of (P31): operating system (Q9135)

Windows XP Media Center Edition (Q2643528)
https://wikidp.org/Q2643528/preview

System requirements
-
Other metadata
Instance of (P31): proprietary software (Q218616), operating system (Q9135) Edition or translation of (P629): Microsoft Windows (Q1406)

Windows XP Starter (Q10393871)
https://wikidp.org/Q10393871/preview

System requirements
-
Other metadata
-

Credits

Authors

Eoin O'Donohoe
Preservation Analyst - the Netherlands Institute for Sound and Vision

Claudia Röck
Preservation Analyst - the Netherlands Institute for Sound and Vision

Jesse de Vos
Product Manager - the Netherlands Institute for Sound and Vision

This report was published by the Dutch Digital Heritage Network (NDE) in September, 2021.
For further information, see: netwerkdigitaalerfgoed.nl

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