Software Metadata Recommended Format Guide

Version 1.1.0
October 2023

Authors and Contributors

Allan Christophersen (The Royal Danish Library); Elena Colón-Marrero (Bentley Historical Library); Dianne Dietrich (Cornell University Library); Patricia Falcao (Tate); Claire Fox (Yale University Library); Karen Hanson (Portico); Allen Kwan (Government of Ontario); Matthew McEniry (Texas Tech University)

DOI: 10.5281/zenodo.10015050

Table of Contents

DOI: 10.5281/zenodo.10015050

DOI: 10.5281/zenodo.10015050

Abstract

The Software Metadata Recommended Format Guide (SMRF) describes and represents metadata elements identified by the Software Preservation Network that are appropriate to describe software materials in the context of a wide range of collections. SMRF aims to be adaptable, so that it can be used in different contexts and systems across libraries, museums, archives, and repositories. It is not meant to be exhaustive; instead SMRF is meant to provide a framework for cultural institutions and collections to determine which metadata to capture, and how to capture it, for their own collections.

Introduction

The Software Metadata Recommended Format Guide (SMRF) provides guidance on descriptive practice for software materials from the perspective of those working in libraries, museums, archives, and repository managers. For the purpose of this document, we are broadly defining software as "a collection of instructions that tell a computer how to work"[1]. The guide identifies technical and descriptive components of software that support discovery, access, and reuse of a range of software types ranging from, but not limited to, commercially produced software to code supporting research data analysis.

To enhance the usability of this guide, these elements have been assigned categories and labels, but this is not meant to suggest that catalogers must use the elements exactly as they have been named or arranged within the guide. The guide is broad; its aim is not to capture all possible variations and complexities of software collections (or software within mixed-media collections), but rather to provide the cataloger a guiding framework to use to handle their materials.

The guide is schema agnostic, so that it can be adapted to a variety of situations and systems. For example, the metadata used to describe the technical aspects of a software item can be mapped to metadata encoding standards such as Dublin Core, MODS, etc.,

[1] "Software." Wikipedia. Wikimedia Foundation, October 4, 2021.

DOI: 10.5281/zenodo.10015050

(some of which we have examples for and others we do not) and may align with your institution's existing processes for distributing such metadata in XML or RDF.

When working through this guide, consider whether you are cataloging at the work level or the expression level.[1] The authors attempted to identify a wide range of elements that are important to software cataloging but there may be gaps in this guide. Users are encouraged to incorporate additional elements, as needed, to complete the cataloging of their materials.

The guide is organized as follows:

1. Overview of metadata elements organized thematically
2. Case Studies
3. Usage Notes and Use Cases
4. Metadata crosswalk (MARC, Dublin Core, MODS, Wikidata, CodeMeta)

Since a broad guide like SMRF may not fulfill all users' needs, we recommend also consulting the following resources for additional software metadata guidelines:

Di Cosmo, R. (2020). biblatex-software style. https://www.ctan.org/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-software (Accessed November 2, 2021.)

CIDOC CRM Special Interest Group. CIDOC Conceptual Reference Model (CRM). International Council of Museums. http://www.cidoc-crm.org/ (Accessed November 2, 2021.)

The CodeMeta Project. CodeMeta Terms. https://codemeta.github.io/terms/ (Accessed November 2, 2021.)

Delve J., A. Ciuffreda, L. Konstantelos. (2011). TOTEM: Trusted Online Technical Environment Metadata - A Long-Term Solution for a Relational Database / RDF

[1] "Functional Requirements for Bibliographic Records." Wikipedia. Wikimedia Foundation, June 2, 2023.

Ontologies. iPRES 2011 - Proceedings of the 8th International Conference on Preservation of Digital Objects. http://hdl.handle.net/11353/10.294265

O'Donohoe, E., C. Röck, J. de Vos. (2021). Preservation Metadata for Software - Describing Software in Archives. Zenodo. https://doi.org/10.5281/zenodo.5503994

Research Data Alliance. FAIR for Research Software (FAIR4RS) WG. https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg (Accessed November 2, 2021.)

This list is not exhaustive, but the referenced documents do touch or expand on the metadata elements that will be presented within this guide.

DOI: 10.5281/zenodo.10015050

Overview of metadata elements

The following section provides an overview of metadata elements identified by the SPN community that support software preservation and long-term access actions. These elements are organized into five scannable tables, which name and provide a brief definition of the element. Each table groups elements based on one of the following broad questions:

1. What is the software?
2. How is the software distributed?
3. Who is responsible for the software?
4. What is required to run the software?
5. What is the software made of?

While SMRF does not suggest any specific mapping to any cataloging system, these elements and guiding questions are meant to provide an appropriate framework for understanding how to approach software cataloging.

DOI: 10.5281/zenodo.10015050

What is the software?

This section includes prompts for metadata that describe the software itself; how to uniquely identify it or disambiguate it from related software. The use of the elements in this section is left to the discretion of individual institutional policies, but the scope of "software" can vary as needed. For example, it is possible to describe a software suite (i.e. Adobe Creative Suite), or individual applications found within a software suite (i.e. After Effects), or specific software libraries (i.e. scikit-learn).

An institution may also wish to contextualize the software in relation to other material in one of the elements below. For example, a software item held by the institution may have been included as part of a software bundle, or it may have been included with the release of another text, such as a CD released with a magazine or textbook.

Element	Definition
Title	The title of the item, as it appears associated with the item being cataloged.
Description	A narrative description of the software's purpose, core functionality, and/or relevant history.
Date	An indication of the period(s) of time associated with the software.
Genre / Type	Category of the software based on either common function, type, or field of use.
Identifier	A unique string used to permanently identify an object within a system or context.
Version	The number, letter, or other schema that identifies a unique state of computer software during development and release.
Accompanying Documentation and/or Materials	Any documentation and/or materials supplied with or about the software.

How is the software distributed?

This section includes elements that aim to describe how the software is made available to a collecting institution, or more broadly to users. It should allow stakeholders to assess how they may get access to software, and, when relevant, limitations to that access. For some implementations, it may be appropriate to describe the layers within the top-level distribution format. For example, software may be distributed in a tar.gz file, virtual machine, compilation disc, or container software (e.g., Docker) but the resources within these may also be considered useful to document. In these instances, the appropriate elements within this section can be repeated or expressed hierarchically as needed.

Element	Definition
File Format	The format of the digital file or files that comprise the software.
Physical Media Format	Refers to a "physical piece of computer-readable hardware that contains some number of digital files."[1]
Size	Refers to the displayed size of a file or aggregate set of files; alternatively size might refer to the number or dimensions of physical media objects.
Condition	Describes the specific digital or physical condition of the media format held in the collection.
Download URL	The URL to download the software, if available.
License	Governs the use of the software, including the right to modify, reuse, and redistribute the software.
Distribution Mechanism	Describes how the software is, or was originally, made available to users, or accessioned into a collection.
Location / Website / Repository	Refers to the location, website, or repository where the software's binaries, source code, metadata, and/or documentation are stored.

[1] Digital Archives Technical Glossary (DANNNG)

DOI: 10.5281/zenodo.10015050

Who is responsible for the software?

This section includes elements that identify who is involved with the software. The elements in this section are specifically tailored to be software-specific roles. Note that there may be additional roles that might be helpful to identify; it is expected that the cataloger will use their judgment as to whether additional roles are appropriate for their material and metadata system of record.

Element	Definition
Creator	The individual or entity that has created or designed the software or conceived of its concept or idea.
Programmer / Developer	The individual or entity responsible for writing any original code in the software and/or overseeing the project. This may be an individual or a group.
Maintainer	The individual or entity responsible for the regular maintenance of the software. They may be distinct from the original creator or programmer and may be a person or a corporate entity.
Publisher	The individual or entity that makes the content available for sale or for free.
Copyright Owner	The individual or entity who owns select intellectual rights associated with an original work.
Contact Information	Describes how to get in touch with the person or entity responsible for the software (e.g., phone number, email address, mailing address, etc.)

DOI: 10.5281/zenodo.10015050

What is needed to run the software?

This section includes elements that specify the hardware and software required for a piece of software to run, and if applicable, under which terms a user can operate the software.

Element	Definition
System Requirements	Refers broadly to the configuration that a system must have for a hardware or software application to run smoothly and efficiently.
Operating System	Software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.
Runtime Environment	The environment in which a program or application is executed.
System Libraries	Programming code, and/or packages that have a well-defined interface from which behaviors can be invoked by a variety of computer programs.[1]
Hardware	Refers to physical components of a computer or any peripherals, including the processor, motherboard, monitor, mouse or pointing device, sound and graphics cards, and storage devices.
Additional Dependencies	Any other requirements for the software that do not fit into the above categories.

[1] "Library (computing)." Wikipedia. Wikimedia Foundation, September 28, 2023.

DOI: 10.5281/zenodo.10015050

What is the software made of?

This section includes elements that describe the makeup and composition of software.

Element	Definition
Programming Language Codebase	The underlying rules that allow a set of prewritten instructions to execute a specific task.
Configuration Language	The language used in the setup, documentation, and/or interface of the software.

DOI: 10.5281/zenodo.10015050

Usage Notes and Use Cases

On the following pages, each element in SMRF will be described in additional detail, beyond the overview definitions offered in the previous section. Each element may include usage notes — additional language meant to clarify the existing definition and offer additional guidance — and example values. In addition, each element includes context for how it can support five key use cases for software metadata, which are defined below.

Definition of use case categories

Execution and Emulation: Metadata that specifies what hardware and software prerequisites are required to interact with, run, or compile a particular piece of software. This metadata may describe a physical or virtual machine.

Discoverability: Metadata to support the ability to locate, acquire, and interact with software using catalogs, indexes, finding aids, or other tools.

Preservation: Metadata to support activities that ensure the authenticity, integrity, reliability, and long-term accessibility of software resources.

Administrative: Metadata that supports the management of software collection materials, including identification of materials, rights management, licensing terms, copyright status, and condition.

Citation: Metadata that supports giving credit or attributing a source for software.

DOI: 10.5281/zenodo.10015050

Detail by element

Title

Example values

- VGA Planets 3
- EaglePOPd Web Interactive: Software to investigate the demography of Bald Eagles in the Northeast, USA from 1990-2018
- Blade Runner
- Subtitled Public

Use case	How this element can support this use case
Execution and Emulation	Though it may not be critical for execution or emulation, title provides useful context to disambiguate from other software and may support additional research into how to run the software.
Discoverability	Title offers key information for discoverability, many searches start with a title search.
Preservation	Title is important information for preservation so that the archive has a record of what is being preserved.
Administrative	Title is important information for understanding and recording what is being managed.
Citation	Title is important for distinguishing which software is being referenced.

DOI: 10.5281/zenodo.10015050

Description

Usage notes: Provide the coverage and level of detail that is most appropriate for your local standards and use cases.

Use case	How this element can support this use case
Execution and Emulation	Through a description, understanding general parameters of what the software was designed to do can help determine whether a particular access strategy is appropriate or not.
Discoverability	Capturing a narrative description can help users evaluate the suitability of the software for their purposes.
Preservation	Having a narrative description of the software can help preservationists understand the nature of the material that is being preserved.
Administrative	Narrative description might not be helpful in administering a collection for individual items, but it might have greater utility in aggregate (e.g., this collection contains software products from a particular era, or research software from a specific lab).
Citation	A narrative description of software is typically not found in a citation, though it might be found in an annotated bibliography, footnote, or endnote.

Date

Usage notes: This guide's definition is deliberately broad, so that the cataloger can capture the date and date type (e.g., copyright, publication, release, version, creation) that makes the most sense for the item that is being cataloged. There are different types of dates that a cataloger might want to capture. It is helpful to capture at least one date or date range and specify what type it is. For example, a piece of software might have a date its copyright was registered or a date it was published (note that criteria for publication may vary, depending on the discipline).

Example values

- Copyright Date: 1993, 1994
- Publication Date: 2019
- Creation Date: 2006; Software has versions from 2006, 2008, and 2018

Use case	How this element can support this use case
Execution and Emulation	Dates provide useful context if you need to determine a "typical" emulation configuration for a piece of obsolete software.
Discoverability	Allows users to sort and filter records by relevant dates.
Preservation	Supplying a date can provide context to identify the object and to support future access.
Administrative	Dates can be highly relevant and helpful in some cases, for example to support management of licenses.
Citation	Date is required for a range of citation use cases outlined in FORCE guidelines.[1]

[1] See Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group. (2016) Software Citation Principles. PeerJ Computer Science 2:e86. DOI: 10.7717/peerj-cs.86

DOI: 10.5281/zenodo.10015050

Genre / Type

Usage notes: This description can be broad, such as video games, application software, programming tools, system software, etc., or more specific, such as business software, computer-aided design, spreadsheets, etc.

Example values

- Video game
- Interactive installation
- Word processing software

Use case	How this element can support this use case
Execution and Emulation	The genre/type can be useful to know an item's intended behavior or functionality.
Discoverability	The genre/type adds value for locating different types of software in a discovery system.
Preservation	Knowing the genre/type of the software provides context into the possible required functionality of the software and impact preservation decisions.
Administrative	Genre can provide a big picture overview of the content in the collection and can assist with more specific tasks that are genre-specific. For example, including genre for all items can assist in identifying video games within a collection when assessing risk and rights management and conditions for access (e.g., permitted on or off-site use).
Citation	Genre/Type is typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

Identifier

Usage notes: Identifiers may be global, such as a DOI or ARK (a unique alphanumeric string used to permanently identify an object and provide a persistent link to its location on the web), or it may be local to a particular system. It could refer to one of a range of source types, some of which have different uses (e.g., noting a version, or publication). If an identifier is local, it is helpful to indicate its context.

Example values

- DOI: https://doi.org/10.7298/q4m1-se95
- PUID: x-sfw/54
- Wikidata: Q2634567

Use case	How this element can support this use case
Execution and Emulation	Identifiers are not typically required to interact with, run, or compile a piece of software, though it may be important information for managing software within an emulation platform like Emulation-as-a-Service Infrastructure (EaaSI).
Discoverability	Identifiers are useful as search terms within discovery environments, and additionally allow users to disambiguate between items that might otherwise seem similar
Preservation	Identifiers are helpful for disambiguation of the objects in a repository and can help with management of duplicates.
Administrative	An identifier in an administrative context can refer to a unique item level, collection, donation, or accession number to assist in tracking and management of an object or larger collection.
Citation	An identifier provides an unambiguous link to the software, and when available, should be included in a citation, as per most citation styles.

Version

Usage notes: Version is used to identify which specific edition or release of the software is being documented. Version information can contain multiple parts, including but not limited to a major version identifier, minor version identifier, a revision identifier, and/or a release identifier. It may be helpful to indicate if a version complies with a standard, such as semantic versioning.

Use case	How this element can support this use case
Execution and Emulation	Knowing and recording the version of a piece of software can support execution and emulation use cases. For example, knowing the version can help confirm specific system requirements, if these were published for a particular version of the same software title. When the software refers to pre-requisites needed to run other software, knowing the version can be extremely helpful in confirming the pieces needed for an appropriate configuration environment.
Discoverability	The version number can help users disambiguate between multiple versions of the same software title.
Preservation	Version supports preservation as metadata describing the content being preserved, helping to disambiguate between multiple versions of the same software title.
Administrative	Recording the version assists with collection management by providing additional information to disambiguate between multiple versions of the same software title.
Citation	Inclusion of version in a citation facilitates the identification and access to a specific version of the cited software.[1]

[1] See Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group. (2016) Software Citation Principles. PeerJ Computer Science 2:e86. DOI: 10.7717/peerj-cs.86

DOI: 10.5281/zenodo.10015050

Accompanying Documentation and/or Materials

Usage notes: This can refer to physical objects or online resources that provide information about the software, its functionality, and how to use it.

Example values

- Printed Manual for Photoshop 8
- Setup information guide; Quick start course; User's Guide; Pocket Reference

Use case	How this element can support this use case
Execution and Emulation	This can identify the presence of additional material, such as installation documents, that can be helpful for using the software and/or building a suitable emulated computing environment.
Discoverability	There may be information in accompanying materials that can supplement records in a discovery system.
Preservation	Listing accompanying documentation can indicate other materials that need to be preserved in addition to the software object. It may also provide useful context for how to preserve the material.
Administrative	It may be helpful to record the presence of accompanying materials that may provide information that supports maintaining the software.
Citation	Accompanying documentation and materials are not typically referenced in a citation.

File Format

Usage notes: This value can be narrative or can use a standard form such as MIME/Media Type or PRONOM format IDs.

Example values

- application/tar, application/exe, image/jpeg, application/pdf, plain/txt
- info:pronom/fmt/726
- ZIP file of a Git repository containing source code.
- Disk Image of the computer supplied by the artist, contains Windows XP and 3 executable files

Use case	How this element can support this use case
Execution and Emulation	File format information can be used to infer system requirements (e.g., "MS-DOS executable") for the software.
Discoverability	File format information can be helpful as part of a record in a discovery system, as it provides information to users that might not be apparent from other descriptive fields.
Preservation	File format, in combination with other technical metadata fields, can be used to support preservation actions necessary to preserve and maintain access to the item.
Administrative	In combination with system requirements, file format information can be used to survey an institution's collection to understand what may be required in order to continue to provide access to the collection and to preserve the collection.
Citation	File format is generally not found in a citation.

DOI: 10.5281/zenodo.10015050

Physical Media Format

Example values

- 3.5” or 5.25” Floppy disk

Use case	How this element can support this use case
Execution and Emulation	Physical media format can be used to identify specific hardware that may be required to access the media containing the software.
Discoverability	Physical media format provides information to users about the physical extent of a piece of software and informs them of specific hardware needed to access the media containing the software.
Preservation	Physical media format can be used to determine any required preservation actions necessary to preserve the item and to maintain access to the item, including any hardware that may need to be preserved to access the media containing the software.
Administrative	Physical media format can be used to survey an institution’s collection to understand what may be required to continue to provide access to the collection and to support management activities.
Citation	Physical media format is typically not found in a citation.

DOI: 10.5281/zenodo.10015050

Size

Usage notes: When identifying size, it can be helpful to note when it refers to size before expansion, installation, or compressing, or when a file is known to be compressed. It is also useful to size on disk with proper units (e.g., MB, GiB, etc.)

Example values

- Approximately 1 MB, compressed
- 1 TB
- 5 CD-ROMs

Use case	How this element can support this use case
Execution and Emulation	Knowing size will help determine the capacity of the system needed to run the software.
Discoverability	In systems where users can directly access content, it can be important to include this information so the user can ensure they have adequate space to download or interact with the software.
Preservation	Knowing size helps plan for storage of an item within a preservation system.
Administrative	Knowing size helps plan for storage of an item within a collection. For example, knowing size may inform decisions about necessary tooling, resourcing, and digital storage, which in turn may inform budget planning.
Citation	Size is not generally included in a citation.

DOI: 10.5281/zenodo.10015050

Condition

Usage notes: Condition can be used for highlighting variations in the condition of the material that may have an impact on use or playback.

Example values

- The outer packaging of the second copy is marked Volume 2, Number 1; however, the CD itself is Volume 2, Number 2
- Missing documentation
- Complete and fully functional
- 3 of the 14 files that form the software were corrupted in the supplied package Repairs were successfully applied and both versions are available

Use case	How this element can support this use case
Execution and Emulation	Condition can be used to document problems such as missing or corrupt components that might affect running the software.
Discoverability	Condition may provide important information to guide access and communicate potential access challenges to patrons.
Preservation	Condition can help determine if, and what kind of, preservation actions are required. In the case of a physical object, it can determine whether access may need to be restricted to continue preserving the object.
Administrative	Condition is likely to be important for administration of the collection. For physical items that are loaned, for example, it would be important to note missing parts for when it gets returned.
Citation	Condition is typically not found in a citation.

DOI: 10.5281/zenodo.10015050

Download URL

Usage notes: This can be used to alert users that the software referenced is available online; using a URL is most helpful when this is the case, though this may not be practical in all situations, so it may be necessary to use free text to specify or clarify how one can download the software. Use a last access date, as appropriate.

Example values

- https://mediaarea.net/en/MediaInfo (accessed 21/10/2020)
- https://archive.org/details/vp320

Use case	How this element can support this use case
Execution and Emulation	If not already acquired, a current download URL allows the user to acquire the software to be run.
Discoverability	Download URL provides patrons with the information needed to acquire the software on their own.
Preservation	Capturing download URL can provide provenance metadata to support preservation of software.
Administrative	Download URLs can be used to show how much of the collection is available for download.
Citation	Download URL is typically required when applicable in a citation, though an identifier may serve the same function.

License

Usage notes: Software may have a primary license and additional licenses based on the original platform it was published or hosted on, or within secondary software dependencies incorporated in the software. SPDX[1] provides a list of licenses that can be referenced by URL. A license is not the same as a license key, which is a piece of code or token that allows a user to authenticate as legally authorized to use the software.

Example values

- GNU General Public License
- Copyright Rafael Lozano-Hemmer (artist)
- https://creativecommons.org/licenses/by/4.0/

Use case	How this element can support this use case
Execution and Emulation	Software licenses can range from extremely permissive to completely proprietary and can impact the level of access an institution can provide to a piece or collection of software. This information can be useful in cases where the license may have implications for use.
Discoverability	Useful in cases where the license could impact a patron's ability to acquire/gain access to the software. For example, the license may indicate which software cannot be provided to the public, or if there might be conditions for access.
Preservation	Software licenses may provide context to identify the object's legal status, including actions that may be required to circumvent DRM for preservation.[2]
Administrative	Software license helps administrators understand what can be done with the software in collections.
Citation	License information is not usually found in a citation.

[1] SPDX License List
[2] A Preservationist's Guide to the DMCA Exemption for Software Preservation, 2nd Edition

DOI: 10.5281/zenodo.10015050

Distribution Mechanism

Usage notes: Distribution mechanism can reflect complexity that the elements file format and physical media format do not cover.

Example values

- Included as a CD-ROM insert to an instructional textbook
- Downloadable with a subscription
- Via Bulletin Board System (BBS) distribution
- Docker-generated tar file with built images included, available for internet download

Use case	How this element can support this use case
Execution and Emulation	Distribution mechanism can provide context for selecting or building an environment to run the software.
Discoverability	Knowledge about the distribution mechanism can communicate to users how the software might be accessed, and in turn could make patrons aware of potential access challenges.
Preservation	Can be used to document provenance of software that is important for supporting long-term preservation of software (e.g., distributed via Docker Container, or software broadcast across radio waves and taped to cassette).
Administrative	Can be used to document provenance of software that is important to retain for collection accession documentation.
Citation	Distribution mechanism is not typically used in citations.

DOI: 10.5281/zenodo.10015050

Location / Website / Repository

Usage notes: This could be internal to the cataloging organization or an external link.

Example values

- Github page: https://github.com/codemeta/codemeta
- Project page: https://mediaarea.net/en/MediaInfo
- Computer History Museum

Use case	How this element can support this use case
Execution and Emulation	If you or your users do not already possess the software, knowing where it can be acquired, and optionally, where its documentation lives, will enable them to obtain or access it, so that it can be executed. It may also provide additional insight helpful for execution: some software platforms may have additional requirements for running the file (e.g. software from Steam requires Steamworks).
Discoverability	Providing repository information lets users filter for specific repositories or platforms.
Preservation	While preservation activities generally require possession of the software, it can be helpful to record the location of access copies that exist external to the preservation system.
Administrative	Similar to the preservation use case, it can be helpful when administering a collection to include the location, website, or repository where binaries or documentation are stored (which might also include a preservation repository).
Citation	Location information, if part of the means by which a user obtains access to the software, is likely to be included in a citation.

DOI: 10.5281/zenodo.10015050

Creator

Usage notes: The creator — the person or persons who have created or designed the software — is sometimes distinct from a publisher entity.

Example values

- Creator: Westwood Studios (1997)
- Artist: Rafael Lozano-Hemmer

Use case	How this element can support this use case
Execution and Emulation	Knowing the creator may help situate the software within a body of work, providing an opportunity to learn helpful contextual information to support a faithful rendering of the software
Discoverability	One or more of the creator elements (creator, programmer, developer, maintainer, publisher) can be a useful entry point for discovery.
Preservation	Creator may be helpful to identify preservation-related issues across the creator's body of work.
Administrative	Creator may be useful in determining or assessing the copyright status of the software.
Citation	All creator elements (creator, programmer, developer, maintainer, publisher) are typically included in a citation.

DOI: 10.5281/zenodo.10015050

Programmer / Developer

Example values

- Programmer: Louis Castle[1]

Use case	How this element can support this use case
Execution and Emulation	Knowing the programmer may sometimes help determine how to run the software where there are features that are common among software created by the same organization/individual.
Discoverability	One or more of the creator elements (creator, programmer, developer, maintainer, publisher) can be a useful entry point for discovery.
Preservation	Knowing the programmer may sometimes help identify preservation issues common among software created by the same organization/individual.
Administrative	The programmer information is unlikely to have a direct application for administering a collection.
Citation	All creator elements (creator, programmer, developer, maintainer, publisher) are typically included in a citation.

[1] "Louis Castle", Wikipedia.

DOI: 10.5281/zenodo.10015050

Maintainer

Example values

- Nightdive Studios[1]

Use case	How this element can support this use case
Execution and Emulation	Knowing the maintainer may sometimes help determine how to run the software where there are features that are common among software created by the same organization/individual. Knowing the maintainer can enable making contact with them if they are still available to help determine how to emulate or execute the software.
Discoverability	One or more of the creator elements (creator, programmer, developer, maintainer, publisher) can be a useful entry point for discovery.
Preservation	Knowing the maintainer may sometimes help identify preservation issues common among software created by the same organization/individual.
Administrative	The maintainer information is unlikely to be important for administering a collection.
Citation	All creator elements (creator, programmer, developer, maintainer, publisher) are typically included in a citation.

[1] See "Remastered Blade Runner adventure available now on PC and consoles" for additional context for this example.

DOI: 10.5281/zenodo.10015050

Publisher

Usage notes: Software publishers often have additional roles, including licensing software from developers, production of physical components of the software (i.e., physical media, instruction manuals), providing technical support, and marketing. The publisher is typically the entity who has made the software available, as identified at the time of the release of the software. This may be the name of a company, or it may be an individual. Include the year of publication when available and appropriate.

Use case	How this element can support this use case
Execution and Emulation	Knowing the publisher can help provide contextual clues related to other software the publisher has made, particularly if the publisher was active during a specific time frame or supported specific platforms.
Discoverability	One or more of the creator elements (creator, programmer, developer, maintainer, publisher) can be a useful entry point for discovery.
Preservation	The publisher information may be helpful to identify preservation-related issues across the publisher's catalog.
Administrative	The publisher information may be useful in determining or assessing the copyright status of the software.
Citation	All creator elements (creator, programmer, developer, maintainer, publisher) are typically included in a citation.

DOI: 10.5281/zenodo.10015050

Copyright Owner

Usage notes: The Copyright Owner may refer to the entity that owns the copyright at the time of the software's initial publication, or it may refer to the current copyright owner if the rights to the software have been sold or otherwise relinquished since the initial publication date.

Use case	How this element can support this use case
Execution and Emulation	Knowing the copyright owner is not typically helpful to run the software. However, like other creator fields, this element can provide helpful context in some cases.
Discoverability	While it is typically not required to know who the copyright owner is to run software, knowing who owns the copyright can assist organizations when providing access to patrons to the software.
Preservation	While it may not be required to know who the copyright owner of the software is in order to preserve it, it can be helpful information to retain to fully understand what rights your organization has to preserve the item.[1]
Administrative	Knowing the copyright owner may be useful for determining or assessing the copyright status of the software.
Citation	Copyright owner is typically not included in a citation for software.

[1] For an example in the US, see the Preservation: Copyright section from the American Library Association.

DOI: 10.5281/zenodo.10015050

Contact Information

Usage notes: Use of this element depends on the local cataloging system. It can include information about the person or entity responsible for creating or maintaining the software, or the contact information for the institution where the software is stored.

Use case	How this element can support this use case
Execution and Emulation	If contact information relates to a software maintainer or creator, this information could allow asking direct questions about intended software functionality.
Discoverability	Names attached to contact information could serve as a point of discovery across collections (less likely this would be the case for email addresses or phone numbers). Contact information related to institutional repositories could help with discovery across aggregated discovery platforms like the Digital Public Library of America (DPLA).
Preservation	If contact information relates to a software maintainer or creator, this information could allow asking direct questions about the production process or known preservation issues.
Administrative	Contact information provides the means to contact a person responsible for the software should any additional questions emerge.
Citation	Detailed contact information is typically not found in a citation.

DOI: 10.5281/zenodo.10015050

System Requirements

Usage notes: System requirements are specified in multiple ways: for example, sources may indicate minimum, recommended, and/or required system requirements. Requirements can be derived from the software packaging, manuals, or other supplied ways and/or supplied based on observation or knowledge of the cataloger. It should be denoted when system requirements are exact wording supplied by some aspect of the entity itself (e.g. creator or maintainer notes, packaging) and when it is based on observation or knowledge of the cataloger.

Example values

- R studio
- Microsoft Access database version 1.1 or later
- For best quality text, use Apple's TrueType fonts or Adobe Type Manager fonts

Use case	How this element can support this use case
Execution and Emulation	System requirements information is needed to configure the emulated or physical computer environment.
Discoverability	System requirements information can be used to identify specific versions of the item in question. It can indicate what a user might need to run the software.
Preservation	System requirements information can provide context to support access to the item, including what other items may need to be preserved in order to provide access to an item.
Administrative	System requirements information may be used to administer a collection, such as determining the extent of software that requires a particular type of hardware or emulated environment.
Citation	System requirements are typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

Operating System

Usage notes: Operating system can refer to a distinct component of a set of system requirements for a piece of software. It is also definable as a piece of software in its own right and may itself have specific hardware requirements. When used as a distinct component of a set of system requirements, operating system can either be a limit or a range: software could have multiple operating systems that it is compatible with.

Example values

- DOS 5 or Windows 3.1
- Windows XP
- iOS

Use case	How this element can support this use case
Execution and Emulation	Operating system is a critical component for configuring and/or providing an appropriate environment (including emulated environments) to run the software.
Discoverability	This information can provide helpful context for users when provided in a discovery platform. When indexed, it can support filters for specific types of content (e.g., "show me all software dependent on Windows 95", etc.)
Preservation	When a part of System Requirements, Operating System provides information on a component of the environment needed to run the software, which is critical for long-term preservation of the software beyond bit fixity.
Administrative	When part of System Requirements, this can help collection managers understand what systems are required to render and preserve the software in their collections.
Citation	Operating System is typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

Runtime Environment

Usage notes: Runtime environment can include any runtime platforms or dependencies that would be necessary for running or emulating the software at a later point, such as .NET framework version, virtual machines, or compilers.

Example values

- .NET Framework 4.8
- Java virtual machine
- Android Runtime
- Adobe Flash

Use case	How this element can support this use case
Execution and Emulation	Runtime environment can be used to configure an emulated environment or computer hardware in order to execute the item in question.
Discoverability	Runtime environment could be useful to guide access and communicate potential access challenges to patrons.
Preservation	Runtime environment provides context to support access to the item, including what other items may need to be preserved to provide access.
Administrative	Runtime environment may not be needed to administer a general software collection, but may be applicable for more specialized collections.
Citation	Runtime environment is typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

System Libraries

Usage notes: While system libraries can be seen as a type of system requirement, there are some cases where it may be helpful to identify them specifically in their own or a separate field.

Example values

- ActiveX

Use case	How this element can support this use case
Execution and Emulation	System libraries are useful to provide further clarity for the system requirements that are necessary to run the software.
Discoverability	System libraries information could be useful to guide access to specific items in the collection and to communicate potential access challenges to patrons.
Preservation	System libraries information provides context to support access to the item, including components of an environment that may need to be preserved to ensure long term accessibility of the item.
Administrative	System libraries information may be useful to provide a survey of an institution's collection to identify groups of items that depend on specific system libraries, which may not be needed for a general software collection but may be useful for a specialized collection.
Citation	System libraries are typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

Hardware

Usage notes: It can be helpful to specify hardware requirements when indicated in documentation, when known by the cataloger, or when it is appropriate for the collection or institution.

Example values

- PlayStation 4
- DOS version requires a colour VGA card and a 286 processor
- IBM Personal Computer & XT; also COMPAQ and 100% IBM compatibles; 192K user memory; one disk drive; Printer and two disk drives recommended

Use case	How this element can support this use case
Execution and Emulation	Hardware information allows you to configure the appropriate emulator or specific hardware to run a software object.
Discoverability	Hardware information provides contextual information on the additional hardware dependencies needed to access a particular software object. For example, a Virtual Reality based game may need a specific set of VR headset to work.
Preservation	If the software is reliant on specific hardware, this information identifies what may need additional preservation consideration.
Administrative	Specifying hardware may help identify hardware maintenance needs for a collection.
Citation	Hardware is typically not referenced in a citation.

DOI: 10.5281/zenodo.10015050

Additional Dependencies

Usage notes: It can be helpful to have a place to list dependencies that provide useful context for running the software, but that are not covered by the other requirements-related elements. For example, this can be a helpful place to indicate requirements related to digital rights management components such as license keys, hardware dongles, or key servers.

Example values

- System Agent software available for Windows 95 (Plus! Pack) can cause peculiar behavior during the Blade Runner install process; recommend against running System Agent software while installing the game

Use case	How this element can support this use case
Execution and Emulation	Describing additional dependencies can allow you to configure the appropriate environment to run a software object.
Discoverability	Listing additional dependencies can provide contextual information on the additional dependencies a user may need to source in order to access a particular software object.
Preservation	Listing additional dependencies can provide context to support access to the object, including what other items may be needed for preservation in order to provide access down the line.
Administrative	When additional dependencies may be required to provide access to software, an administrator may need to determine how these dependencies will be acquired (e.g., allocate a budget to purchase required dependencies).
Citation	Additional dependencies are typically not included in a citation.

DOI: 10.5281/zenodo.10015050

Programming Language Codebase

Usage notes: The element can be used to indicate the programming language or languages used by a programmer or developer in the creation of a piece of software. Note that programming languages have versions and may be important to capture as well here. For certain types of software this may also include higher order frameworks such as game engines for video games.

Example values

- Flappy Bird (2013) was programmed using Objective C
- Half-Life 2 (2004) was programmed using C++ for Source Engine 2004
- Hypercard
- Programmed in R

Use case	How this element can support this use case
Execution and Emulation	If the item consists of source code, knowing what programming language it is written enables one to run (or compile) the code.
Discoverability	Programming language information can be used to guide access to specific items in the collection or to communicate potential access challenges to a user.
Preservation	The programming language can provide context to support ongoing maintenance of an item in the collection, particularly if the item includes source code.
Administrative	Programming language may be useful to understand the nature of the collection as a whole. For example, the institution may identify a need to support a specific programming language due to the nature of its collection.
Citation	Programming language codebase is typically not used in a citation.

DOI: 10.5281/zenodo.10015050

Configuration Language

Use case	How this element can support this use case
Execution and Emulation	The configuration language can be used to inform a user of additional language requirements necessary to execute the software. For example, manuals may need to be translated into the primary language of the institution in order to provide wider access to the software or the operating system may need additional language packs or set to a specific locale in order to properly install and use the software.
Discoverability	The configuration language can be used to help identify software that may have been produced for a specific market or for a specific audience.
Preservation	The configuration language can provide additional information required to support access to the object, such as translated materials if applicable. Similarly, some software may require additional language packs to be installed on an operating system before they can be executed and these packs may need to be preserved as well.
Administrative	The configuration language may be helpful to understand the institution's collection as a whole. For example, it can be used to identify if a specific group of users are being served or underserved by the collection, or if additional resources are required to support parts of the collection.
Citation	Configuration language is typically not referenced in a citation

DOI: 10.5281/zenodo.10015050

Case Studies

This section provides examples of different software items and how SMRF elements can be used to describe them. Each example is presented in a table with three columns. The "Metadata value" and "Original label" columns show the information about the item and how it was presented in its original context. Not all the examples in this section are of formal catalog records, so in some cases, the original context is metadata that is not labeled or structured. SMRF aims to provide element labels where there often are none, so these fields can be gathered where necessary. The "Mapping to SMRF element" column provides a mapping to specific SMRF-recommended element, to illustrate how the information can be represented using this framework.

This section includes the following examples:

- VGA Planets v3.00, an example of disk images created from shareware diskettes, presented on the Internet Archive;
- EaglePOPd Web Interactive: Software to investigate the demography of Bald Eagles in the Northeast, USA from 1990-2018, an example of research data as deposited into a research library's institutional repository (DSpace);
- Blade Runner, an example of a mass-marketed, popular video game from 1997, as displayed on Moby Games;[1] and
- Subtitled Public, an example of artist's software, as referenced on the artist's own webpage describing the work.

[1] This example refers to the original United States 1997 release for this example, and primarily derived metadata from a physical copy of the game supplied by a member of the Metadata Working Group. To see an alternate representation of this item's metadata, see the catalog record from the University of Michigan.

DOI: 10.5281/zenodo.10015050

Shareware Diskettes

Metadata value	Original label	Mapping to SMRF element
VGA Planets v3.00	(Title)	Title
1994	Publication Date	Date, Publication
DOS games, Vintage computer games, Simulation games	Topics	Genre/Type
“VGA Planets is a graphical, multi-player, play by electronic mail, space war game...”		Description
vp320	Identifier	Identifier
shareware	License	License
dosbox	Emulator	System Requirements

DOI: 10.5281/zenodo.10015050

Research Data

Metadata value	Original label	Mapping to SMRF element
EaglePOPd Web Interactive: Software to investigate the demography of Bald Eagles in the Northeast, USA from 1990-2018	dc.title	Title
https://hdl.handle.net/1813/66308.2	dc.identifier.uri	Identifier
Software	dc.type	Genre/Type
2 ZIP fields with code and validation bundle, plus 1 PDF ReadMe file		File Format
Approximately 800 Kb (compressed)		Size
MIT, Open Source		License
eCommons@Cornell		Location / Website / Repository
Brenda Hanley	dc.contributor.author	Programmer
R Studio Version 1.1.463		System Requirements
R packages: shinyBS, shiny, MASS, popdemo, rmarkdown		Additional dependencies

DOI: 10.5281/zenodo.10015050

Mass-marked published software

Metadata value	Original label	Mapping to SMRF element
Blade Runner	(Title)	Title
1997	Released	Copyright Date
Westwood Studios, Inc.	Publishers	Publisher
Intel Pentium	Minimum CPU Class Required	Hardware
Windows 95	Minimum OS Class Required	Operating System
16 MB	Minimum RAM Required	Additional Dependencies
DirectX 5	Minimum DirectX Version Required	System Libraries
CD-ROM	Media Type	Physical Media Format
4X (600 KB/s)	Minimum CD-ROM Drive Speed Required	Hardware
2 MB	Minimum Video Memory Supported	Additional Dependencies
640x480	Video Resolutions Supported	Additional Dependencies
Keyboard, Mouse	Input Devices Supported	Hardware

DOI: 10.5281/zenodo.10015050

Artist’s software

Metadata value	Original label	Mapping to SMRF element
Subtitled Public	(Title)	Title
Radael Lozano-Hemmer	(Artist)	Creator
Conroy Badger	Programming	Programmer
2005	Year of creation	Creation Date
Computer network of at least 2 computers, usually 5; surveillance video cameras and video projectors.		Hardware
ActiveX		System Libraries
Windows XP		Operating System

DOI: 10.5281/zenodo.10015050

Crosswalk

The following tables show how the elements in SMRF can be mapped to various schema, and include mapping to the following:

- MARC 21 Format for Bibliographic Data
- Dublin Core: DCMI Metadata Terms
- Metadata Object Description Schema (MODS)
- Wikidata
- CodeMeta

Empty values in the crosswalk table below indicate that the authors did not find a suitable mapping in the target schema. If you have suggestions for those values, or any other existing mapping, please do let us know. We will review and place it under consideration for an update of this guide.

Note: To locate referenced values from the Wikidata column, enter the text inside the parentheses (e.g., P1476) in the "Search Wikidata" searchbar.

DOI: 10.5281/zenodo.10015050

What is the software?

Element	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Title	245 - Title Statement	Title	<titleInfo>	title (P1476)	name
Description	520 - Summary, etc.	Description	<abstract>		Description
Date, Date Subtype (Creation)	260 - Publication, Distribution, etc. 388 - Time Period of Creation	Date Date Created	<originInfo> <dateCreated>	inception (P571)	dateCreated
Date, Date Subtype (Publication)	260 - Publication, Distribution, etc.	Date	<originInfo> <dateIssued>	publication date (P577)	datePublished
Date, Date Subtype (Release)	260 - Publication, Distribution, etc.	Date Date Available	<originInfo> <dateIssued>	publication date (P577)	datePublished
Date, Date Subtype (Version)	260 - Publication, Distribution, etc.	Date	<originInfo> <dateOther>		datePublished

DOI: 10.5281/zenodo.10015050

Element	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Date, Date Subtype (Copyright)	260 - Publication, Distribution, etc. 542 - Information Relating to Copyright Status	Date Copyrighted	<originInfo> <copyrightDate>	copyright date (Q59584702)	datePublished
Genre/Type		Type	<genre> <typeOf Resource>	genre (P136) instance of (P31)	applicationCategory
Identifier	024 - Other Standard Identifier	Identifier	<identifier>	unique identifier (Q6545185)	identifier
Version			<originInfo> <edition>	software version identifier (P348) version type (P548)	version softwareVersion
Accompanying Documentation and/or Materials			<note>		releaseNotes

DOI: 10.5281/zenodo.10015050

How is the software distributed?

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
File Format	256 - Computer File Characteristics 347 - Digital File Characteristics 516 - Type of Computer File or Data Note	Format	<physicalDescription><form> <physicalDescription><internetMediaType>	file format (P2701)	programmingLanguage runtimePlatform
Physical Media Format	300 - Physical Description	Format	<physicalDescription><form>	distribution format (P437)	
Size	347 - Digital File Characteristics	Extent	<physicalDescription><extent>	data size (P3575)	fileSize
Condition	583 $l - Action Note (Status)		<physicalDescription><note type="condition">		

DOI: 10.5281/zenodo.10015050

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Download URL	856 - Electronic Location and Access		<location><url>	download link (P4945)	downloadUrl
License		License	<access Condition>	copyright license (P275)	License
Accompanying Documentation and/or Materials					referencePublica tion
Distribution Mechanism				distribution format (P437)	codeRepository
Location / Website / Repository	856 - Electronic Location and Access		<location><url>	URL (P2699)	codeRepository

DOI: 10.5281/zenodo.10015050

Who is responsible for the software?

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Creator	100 - Main Entry-Personal Name 110 - Main Entry-Corporate Name	Creator	<name>	creator (P170)	author contributor
Programmer	100 - Main Entry-Personal Name 110 - Main Entry-Corporate Name	Contributor	<name> <name><role>	programmer (P943)	author contributor
Developer	100 - Main Entry-Personal Name 110 - Main Entry-Corporate Name	Contributor	<name> <name><role>	developer (P178)	author contributor
Maintainer	100 - Main Entry-Personal Name 110 - Main Entry-Corporate Name	Contributor	<name> <name><role>	maintained by (P126)	author contributor

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Publisher	100 - Main Entry-Personal Name 110 - Main Entry-Corporate Name 260 - Publication, Distribution, etc.	Publisher	<originInfo> <publisher>	publisher (P123)	author contributor
Copyright Owner	542 - Information Relating to Copyright Status			copyright holder (P3931)	copyrightHolder
Contact Information					maintainer

DOI: 10.5281/zenodo.10015050

What is needed to run the software?

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
System Requirements	538 - System Details Note	Requires	<note>		processorRequirements softwareRequirements memoryRequirements storageRequirements
Operating System	538 - System Details Note	Requires	<note>	operating system (P306)	operatingSystem targetProduct
System Libraries	538 - System Details Note	Requires	<note>	depends on software (P1547)	softwareRequirements
Runtime environment	538 - System Details Note	Requires	<note>		runtimePlatform

DOI: 10.5281/zenodo.10015050

Hardware	538 - System Details Note	Requires	<note>	platform (P400)	processorRequirements
Additional Dependencies	538 - System Details Note	Requires	<note>		softwareSuggestions

DOI: 10.5281/zenodo.10015050

What is the software made of?

Suggested Element Name	MARC	Dublin Core	MODS	Wikidata	CodeMeta
Programming Language Codebase	538 - System Details Note		<note>	programmed in (P277)	programmingLanguage
Configuration Language	538 - System Details Note				

Thank You and Acknowledgements

The Software Metadata Recommended Format Guide was developed and written by past and current members of the Software Preservation Networks' Metadata Working Group. The development of the Guide was also informed by contributions from the broader Software Preservation Network community received via email, comments on drafts, and from Software Preservation Network all member meetings.

References

Di Cosmo, R. (2020). biblatex-software style. https://www.ctan.org/tex-archive/macros/latex/contrib/biblatex-contrib/biblatex-software (Accessed November 2, 2021.)

CIDOC CRM Special Interest Group. CIDOC Conceptual Reference Model (CRM). International Council of Museums. http://www.cidoc-crm.org/ (Accessed November 2, 2021.)

The CodeMeta Project. CodeMeta Generator. https://codemeta.github.io/codemeta-generator/ (Accessed November 2, 2021.)

The CodeMeta Project. CodeMeta Terms. https://codemeta.github.io/terms/ (Accessed November 2, 2021.)

Delve, J., A. Ciuffreda, L. Konstantelos. (2011). TOTEM: Trusted Online Technical Environment Metadata - A Long-Term Solution for a Relational Database / RDF Ontologies. iPRES 2011 - Proceedings of the 8th International Conference on Preservation of Digital Objects. http://hdl.handle.net/11353/10.294265

Force11. Software Citation Implementation Working Group. https://www.force11.org/group/software-citation-implementation-working-group (Accessed November 2, 2021.)

DOI: 10.5281/zenodo.10015050

O'Donohoe, E., C. Röck, J. de Vos. (2021). Preservation Metadata for Software - Describing Software in Archives. https://doi.org/10.5281/zenodo.5503994

Katz, D.S., et al (2019). Software Citation Implementation Challenges. https://arxiv.org/abs/1905.08674

Research Data Alliance. FAIR for Research Software (FAIR4RS) WG. https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg (Accessed November 2, 2021.)

Software Preservation Network Metadata Working Group. (2018). Software Metadata Crosswalk V.2 https://docs.google.com/spreadsheets/d/1Xq3eqblDUkxcxK69iYI6qZ5Ubd39NZ5bTg3ErJKFY60/edit?usp=sharing