perm.pub/dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/1.4
Additional formats and editions available online.

Edition 1.4

Obsolete

Author date: 2023-10-07

Archive date: 2023-10-08

Citation:
E. Castedo Ellerman (2023) "Digital Succession Identifiers" perm.pub https://perm.pub/dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/1.4

Digital Succession Identifiers

E. Castedo Ellerman (castedo@castedo.com)

Abstract

DOCUMENT TYPE: Technical Specification

The Digital Succession Identifier (DSI) is a persistent identifier for bibliographic references. This specification of DSI is specific to the initial implementation using git.

Introduction

For a non-technical background and motivation for this technology, refer to Why Publish Digital Successions.

Digital successions

A digital succession contains digital objects, which are fixed finite collections of bits. Examples of digital objects include computer files and file system directories (folders). Both git [1] and Software Heritage [2] can represent a file system directory as a digital object. Each digital object in a digital succession represents a new edition of a previous digital object and is assigned a number that determines its order. The digital succession expands, but the digital objects within the succession remain unchanged.

In addition, a digital succession is digitally signed. In this specification, the digital signature is made using an SSH signing key via git [1]. When a digital succession is first created, it only consists of a digitally signed genesis record and no digital object editions. A *Digital Succession Identifier* is an intrinsic identifier [2] [3] of the genesis record.

Digital successions implemented via git

In the git implementation, the genesis record is a signed initial git commit with an empty tree (and no parent). To expand the digital succession, additional git commits are made using the same signing key. The git tree of each commit is not a digital object in the succession. Instead, it is a record of all the digital object editions in the succession. The top level directory consists of subdirectories named as non-negative integers. Each subdirectory contains either an entry named `object` or entries named as non-negative integers that are subdirectories. An entry named `object` represents a digital object edition in the digital succession, which can be a file (git blob) or a directory (git tree). For example, when a single file is added as edition 1, the full

succession record is a directory with the path `1/object` leading to a git blob representing edition 1.

Multilevel edition numbering

In the simplest scenario, edition numbers are positive integers. However, multilevel edition numbering may be used for more advanced usage. Multilevel numbering is commonly used in the numbering of chapters, sections, and subsections (e.g. chapter 2, section 2.4, subsection 2.4.3) as well as software release versions (e.g. software release 2.19.2).

An edition number prefix, such as `1`, can specify either a digital object edition or the entire sequence of editions `1.1`, `1.2`, `1.3`, etc... An edition number identifies either a digital object edition or a sequence of subordinate edition numbers, but not both. Larger integers indicate newer editions that obsolete older editions with smaller integers. The DSI specification does not assign any semantic meaning to different number levels.

Textual representation of a Digital Succession Identifier

The textual representation of a Digital Succession Identifier (DSI) is a base identifier followed by an optional slash followed by an optional edition number (possibly multilevel). The edition number is represented as non-negative integers separated by periods.

The base DSI is calculated from the git initial commit (genesis record) of a digital succession. Git-compatible software can calculate a 20-byte binary hash that identifies the digital succession. This 20-byte binary hash is usually represented textually as a 40-digit hexadecimal number. However, for a DSI, this 20-byte binary hash has a 27-character representation in standard base64url format (RFC 4648)[4].

It is worth noting that when a new digital succession (git initial commit) is created, a user can choose not to use it and instead immediately create a new one with a different base64url text representation. There is little cost in not using empty digital successions and recreating new genesis records (git initial commits) until an acceptable base64url identifier is found.

Examples

Base DSI of this specification:
`dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo`

DSI of the first edition:
`dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/1`

DSI of the first digital object (subedition) of the first edition:
`dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/1.1`

Future extensions

To support future enhancements, there are three paths to extending this textual representation. These paths involve using a base DSI where:

- a character is neither a slash (/) nor one of the 64 base64url characters,
- the number of characters is different than 27, or

- the 27th character is one of the 48 base64url characters that never appear as the last character of a base64url encoding of 40 bytes (i.e., any base64url character that is not `A`, `E`, `I`, `M`, `Q`, `U`, `Y`, `c`, `g`, `k`, `o`, `s`, `w`, `0`, `4`, or `8`).

Design Choice Rationales

Separation of git history from edition history

A significant design choice is to not directly rely on git commit history to determine the succession of editions. Git commit history accurately records the actions performed with git, but it can be inflexible and confusing for establishing a clean linear history. Software Heritage automatically preserves git commits, which compounds the risk that git commits do not correspond well, or even prevent, an intended clean linear history. There may be situations where having merge commits and non-linear git commit history will be convenient.

Having edition history separate from git history also provides a potential path for an enhancement akin to retractions of specific editions.

Use of git tree paths instead of git tags

Edition numbers are similar to release versions of software projects. The usual practice in software development is to use git tags to identify releases. In contrast, this specification takes a different approach. Edition numbers are recorded with file paths in git trees rather than git tags.

The main reason for this different approach has to do with how digital successions are recorded and copied. A highly desirable feature for digital successions is the ease of copying without errors. It is convenient to copy digital successions from multiple sources and store them in a single git repository. In this specification, a complete digital succession is captured by just a chain of commits (initial commit to latest commit). In contrast, entire software projects, which often include release tags, are usually copied by cloning an entire repository. If edition numbers were recorded as git annotated tags, copying digital successions properly would be more complicated and error-prone (due to missing tags).

Git repository branches are a convenient tool for managing digital successions. However, branch names are not part of the digital succession record.

Use of base64url rather than hexadecimal

The textual identifier is in base64url (Base64 with URL and filename safe alphabet) as specified in RFC 4648 [4]. Both this format and hexadecimal have pros and cons.

As mentioned in the introduction to the textual representation, there is no obligation to use the Digital Succession Identifier (DSI) generated by a brand new digital succession. If a DSI is unacceptable to the creator of a new digital succession, a new DSI can be easily generated. Helper tools could perform this function for users by default.

The main con to base64url is that it can contain characters that are more prone to copy errors when transmitted via human sight. Certain fonts and handwriting can make poor or no distinction between certain character pairs. For example, some popular sans serif fonts make no visual distinction between capital I and lowercase l.

The main pro to base64url is that it is 27 characters rather than 40. Given that DSIs are used in contexts similar to DOIs, it is likely more acceptable to adopters to use 27 characters, which is very typical for a long DOI, rather than 40 characters. Displaying an ID of 27 characters rather than 40 characters also fits better on web pages rendered on mobile devices. 27 characters are less likely to be hidden by showing ellipsis, horizontal scroll regions, or shrunk into extremely small fonts.

This design decision is partly made on the belief that the copy-via-human-sight issue is increasingly mitigated by a few technology trends:

1. Computer systems are relied upon more and more via more reliable copying mechanisms such as clicking on hyperlinks, copy-and-paste, and QR codes instead of visual copying and handwriting.
2. Websites tend to switch to using appropriate fonts for code like base64url IDs when output into HTML. The same applies to tools that automatically produce PDFs from computer data.
3. Human-to-computer interfaces, such as input fields of Internet-connected software, tend to add features like autocomplete and search disambiguation for typos when sufficient need arises for such a feature.

Use of the term edition

The term edition is used to refer to a digital object in a digital succession. Part of the motivation is to avoid confusion with the many uses of the term version. For instance, version can be used to refer to versions of a digital object that are not added as editions.

This language is partly due to the initial application of digital successions for scientific articles. Although there are rarely new versions of scientific articles after they are published, it is well established that scientific textbooks are published in multiple *editions*.

In the initial application of digital successions, documents are the digital object. A single edition of a document might be presented in different HTML and PDF *versions*, even though they are the same *edition* of a document.

Minutiae

- A multilevel edition number should not have more than four levels (components). Edition number components (per level) must not be negative and should not exceed four decimal digits.

Acknowledgments

Thank you to Valentin Lorentz for questions about design choices and pointing out an important shortcoming in how GPG digital signatures were used in the initial implementation of the hidos library (version 0.3) [5].

Further Reading

- This specification is heavily influenced by the concept of *intrinsic identifier* and related concepts discussed in [2] [3].
- For a discussion on various concepts and proposed terminology about persistent identifiers, see [6]. In the proposed terminology, a DSI is a persistent identifier (PID) that is “frozen” and “waxing” with both intraversioned and extraversioned PIDs depending on the edition number.

Changes

From edition 1.2

- Reference SSH signing keys instead of GPG/PGP signing keys.

From edition 0.2

- Add section about multilevel edition numbering.
- Add design rationale behind git tree paths instead of git tags.
- Note future expansion paths for text representation.

References

1. Git — Wikipedia, the free encyclopedia. 2023. Available: https://en.wikipedia.org/w/index.php?title=Git&oldid=1177307938
2. Cosmo RD, Gruenpeter M, Zacchiroli S. Referencing Source Code Artifacts: A Separate Concern in Software Citation. Computing in Science & Engineering. 2020;22: 33–43. doi:10.1109/MCSE.2019.2963148
3. Di Cosmo R, Gruenpeter M, Zacchiroli S. Identifiers for Digital Objects: the Case of Software Source Code Preservation. iPRES 2018 - 15th International Conference on Digital Preservation. Boston, United States; 2018. pp. 1–9. Available: https://hal.archives-ouvertes.fr/hal-01865790
4. Josefsson S. The Base16, Base32, and Base64 data encodings. RFC Editor; Internet Requests for Comments; RFC Editor; 2006 Oct. doi:10.17487/RFC4648
5. Hidos 0.3. 2022. Available: https://archive.softwareheritage.org/swh:1:rev:b963e5d2366724df6e8c34d864a7984ce4a2e1be;origin=https://gitlab.com/perm.pub/hidos
6. Kunze J, Calvert S, DeBarry JD, Hanlon M, Janée G, Sweat S. Persistence Statements: Describing Digital Stickiness. Data Science Journal. 2017;16: 39–. doi:10.5334/dsj-2017-039