perm.pub/dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/0.1
Additional formats and editions available online.

Edition 0.1

Obsolete

Author date: 2022-08-09

Citation:
E. Castedo Ellerman (2022) "Digital Succession Identifiers" perm.pub https://perm.pub/dsi:1wFGhvmv8XZfPx0O5Hya2e9AyXo/0.1

Digital Succession Identifiers

E. Castedo Ellerman (castedo@castedo.com)

Abstract

STAGE: Early draft

DOCUMENT TYPE: Technical Specification

This is a specification of Digital Succession Identifiers implemented over git.

Introduction

This is a technical specification of *Digital Succession Identifiers*. A non-technical background and motivation for this new technology can be found in the description of the perm.pub pilot project.

Digital successions

A digital succession contains digital objects. A digital object in this context is a fixed finite collection of bits. A computer file is an example of a digital object. A file system directory (folder) can also be represented as a digital object by both git[1] and Software Heritage [2]. Conceptually, each digital object in a digital succession is a new edition of a previous digital object. Each edition is assigned a number which determines its order. Although a digital succession expands, each digital object within the succession does not change.

A digital succession is also digitally signed. In the simplest case, a single user signs using a digital key that is verified to be for that user. In this specification, the digital signature is made using a PGP key [3].

When a digital succession is created, it consists of only a digitally signed genesis record and no digital object editions. The *Digital Succession Identifier* is an intrinsic identifier [4] 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 commits are made using the same signing key. The git tree associated with each additional commit is not a new edition of the digital object. The git tree of a commit is a full succession record. The full succession record is a directory of all editions to date. The top level directory consists of only subdirectories named as non-negative

numbers. Within each number named subdirectory is an entry named `object`. This entry is a digital object edition in the digital succession. This digital object 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 one path of `1/object` to a git blob which is edition 1.

Textual representation of a Digital Succession Identifier

Given a digital succession genesis record, git and open-source software that emulate git, can calculate a 20 byte binary hash which identifies that digital succession. In most uses of git, this 20 byte binary hash is represented textually as a 40 digit hexadecimal number. In this specification, the textual identifier of the 20 byte binary hash is a 27 character representation in standard base64url format (RFC 4648)[5].

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.

Design Choice Rationales

Use of the term edition

The term edition is used to refer to a digital object in the 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. One can also talk about versions of the entire succession record.

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.

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 is conducive to accurately and faithfully recording the actions performed with git. However this often can be inflexible and confusing for establishing a clean linear history. That Software Heritage automatically preserves git commits compounds the risk that git commits do not correspond well, or perhaps even prevent, an intended clean linear history. It seems likely that there will 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 base64url rather than hexadecimal

The textual identifier is in base64url (Base64 with URL and filename safe alphabet) as specified in RFC 4648 [5]. 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 unac-

ceptable 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 which are more prone to copy errors when transmitted through 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. Future upgrades to DSI will likely make those lengths even larger, and the difference greater. 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 like 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 system 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 for 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.

Minutiae

- An edition number must be a positive integer strictly less than 65,536 (2 to the 16th power). Edition number zero is reserved for special future use.

References

1. Git — Wikipedia, the free encyclopedia. 2022. Available: https://en.wikipedia.org/w/index.php?title=Git&oldid=1109231861
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. Pretty good privacy — Wikipedia, the free encyclopedia. 2022. Available: https://en.wikipedia.org/w/index.php?title=Pretty_Good_Privacy&oldid=1108847090
4. Di Cosmo R, Gruenpeter M, Zacchiroli S. 204.4 Identifiers for Digital Objects: The case of software source code preservation. 2022. doi:10.17605/OSF.IO/KDE56
5. Josefsson S. The Base16, Base32, and Base64 data encodings. RFC Editor; Internet Requests for Comments; RFC Editor; 2006 Oct. doi:10.17487/RFC4648