International Standard

ISO/IEC 18670

First edition

Information technology — SoftWare Hash IDentifier (SWHID) Specification V1.0

PROOF/ÉPREUVE

Reference number
ISO/IEC 18670:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by JDF [as The SoftWare Hash Identifier (SWHID) Specification Version 1.0] and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Modern software relies heavily on open source components that are developed collaboratively in a distributed setting, and that are assembled to create complex systems that evolve at a fast pace.

This has strengthened the need to precisely track, ensure availability, and guarantee integrity of the components that go into a given system for a variety of stakeholders. Academia needs to ensure that research results are reproducible, industry needs to improve the traceability of the software supply chain, and developer communities need tools to cope with the increasing complexity.

A key building block for addressing this issue is a system of intrinsic identifiers that allows users to precisely pinpoint the exact version of any software artifact, at all levels of granularity, without relying on any central registry or naming authority.

With this specification, the SWHID working group makes such a system of intrinsic identifiers, originally developed for the Software Heritage universal source code archive,[1] available to all stakeholders.

For the sake of clarity, examples have been drawn directly from the Software Heritage archive; however, it is important to note that systems for the persistent archival of software artifacts, as well as resolution of SWHIDs, are outside the scope of this specification, which does not require the use of Software Heritage.

Information technology — SoftWare Hash IDentifier (SWHID) Specification V1.0

1 Scope

This specification defines a standard data format for referencing software artifacts that match the data model of modern distributed version control systems.

This format includes the typical tree-like structure of a filesystem hierarchy, but also, special nodes to track revisions and releases, as well as the full status of a version control system, with all its development branches.

A key property of SWHIDs is that they can be computed using cryptographically strong functions directly from the digital objects they refer to, by anyone that has access to a copy of those objects. This enables decentralised and independent verification of integrity, without relying on a registry or a central authority.

The computation of the SWHID identifiers is based on Merkle Acyclic Directed Graphs, a natural generalization of Merkle trees.

The resolution of SWHIDs, that is, the process of obtaining a copy of a digital artifact corresponding to a given SWHID, is outside the scope of this specification.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

RFC-3174, *US Secure Hash Algorithm 1 (SHA1), The Internet Society Network Working Grouphttps://tools.ietf.org/html/rfc3174*

RFC-3986, *Uniform Resource Identifier (URI): Generic Syntax, The Internet Society Network Working Grouphttps://tools.ietf.org/html/rfc3986*

RFC-3987, *Internationalized Resource Identifiers (IRIs), The Internet Society Network Working Grouphttps://tools.ietf.org/html/rfc3987*

RFC-5234, *Augmented BNF for Syntax Specifications: ABNF, The Internet Society Network Working Grouphttps://tools.ietf.org/html/rfc5234*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
branch
parallel line of development in a *version control system* (3.7), that stems from the main line

3.2
Git
distributed *version control system* (3.7) created by Linus Torvalds in 2005

3.3
hierarchical file system
method of organizing and managing files in a computer where data is stored hierarchically

3.4
intrinsic identifier
identifier that can be computed directly from the object that it identifies, without needing access to a registry

3.5
repository
storage location for software development artifacts (3.8) including but not limited to source code, build scripts, and documentation

3.6
SHA1
SHA1
Secure Hash Algorithm 1
hash function that takes as input a sequence of bytes and produces a 160-bit (20-byte) hash value

Note 1 to entry: The returned value is called *SHA1 checksum*, or simply *SHA1* when there is no risk of ambiguity between the function and the returned value. A detailed description of how to compute SHA1 is available in RFC-3174.

In the wake of the Shattered attack of 2017 (see[3]), it is now possible to produce collision-prone files that are different but return the same SHA1 checksums. It is however possible to detect, during SHA1 computation, such SHA1-colliding files using counter-cryptanalysis (see[2]).

As collision-prone files are problematic from the point of view of unequivocal identification and integrity verification, the SWHID standard takes measures to avoid that such files are referenced using only SHA1 checksums. For the purpose of this specification, the SHA1 function is therefore considered to be a *partial* function, that only returns a value when a Shattered-style collision is not detectable using the techniques described in[2] and the reference implementation of it available at https://github.com/cr-marcstevens/sha1collisiondetection (Git commit ID `38096fc021ac5b8f8207c7e926f11feb6b5eb17c`, or version `stable-v1.0.3`). When such a collision is detected during SHA1 computation, no SHA1 can be obtained for the object in question and hence, depending on the context, a valid SWHID might not exist for it.

In most cases, SHA1s in this specification are computed on objects after adding specific headers to them, making "trivial" collision-prone files still perfectly valid and hence referenceable using SWHIDs.

3.7
version control system
revision control system source control system software tool that helps manage different versions of *software development artifacts* (3.8)

3.8
software artifact
object
representation of a distinct entity identifiable by a SWHID

3.9
metadata
supplementary information associated with a *software artifact* (3.8)

3.10
UNIX epoch
time reference point that denotes the precise moment at 00:00:00 Coordinated Universal Time (UTC) on 1 January 1970

4 Syntax

A SWHID consists of two separate parts: a mandatory *core_identifier* that can identify any software artifact, and an optional list of *qualifiers* that allows specification of the context where the object is meant to be seen and that points to a subpart of the object itself.

Syntactically, SWHIDs are generated by the `<identifier>` entry point in the following grammar (which uses notation defined by RFC-5234):

```
<identifier> ::= <core_identifier> [ <qualifiers> ] ;

<core_identifier> ::=
    "swh" ":" <scheme_version> ":" <object_type> ":" <object_id> ;
<scheme_version> ::= "1" ;
<object_type> ::=
    "snp"  (* snapshot *)
  | "rel"  (* release *)
  | "rev"  (* revision *)
  | "dir"  (* directory *)
  | "cnt"  (* content *)
  ;
<object_id> ::= 40 * <hex_digit> ; (* intrinsic id, hex-encoded *)
<dec_digit> ::=
    "0" | "1" | "2" | "3" | "4"
  | "5" | "6" | "7" | "8" | "9" ;
<hex_digit> ::=
    <dec_digit>
  | "a" | "b" | "c" | "d" | "e" | "f" ;

<qualifiers> ::= ";" <qualifier> [ <qualifiers> ] ;
<qualifier> ::=
    <context_qualifier>
  | <fragment_qualifier>
  ;
<context_qualifier> ::=
    <origin_ctxt>
  | <visit_ctxt>
  | <anchor_ctxt>
 | <path_ctxt>
  ;
<origin_ctxt> ::= "origin" "=" <url_escaped> ;
<visit_ctxt> ::= "visit" "=" <core_identifier> ;
<anchor_ctxt> ::= "anchor" "=" <core_identifier> ;
<path_ctxt> ::= "path" "=" <path_absolute_escaped> ;
<fragment_qualifier> ::= "lines" "=" <range> | "bytes" "=" <range> ;
<range> ::= <number> ["-" <number>] ;
<number> ::= <dec_digit> + ;
<url_escaped> ::= (* RFC 3987 IRI *)
<path_absolute_escaped> ::= (* RFC 3987 absolute path *)
```

The last two symbols are defined as:

— `<url_escaped>` is an `IRI` as defined in RFC-3987; and

— `<path_absolute_escaped>` is an `ipath-absolute` from RFC-3987.

In both of these, all occurrences of `;` (and `%`, as required by the RFC) have been percent-encoded (as `%3B` and `%25` respectively). Other characters may be percent-encoded, for example, to improve readability and/or embeddability of SWHID in other contexts.

5 Core identifiers

5.1 General

A core identifier is composed of four fields, each separated by a colon (`:`), as follows:

The first field is the *type of the identifier*, and it is defined to be `swh`.

The second field is the *version of the identifier scheme* and for this version of the specification it is defined to be `1`.

The third field is a tag corresponding to the *type of object* Identified, as follows:

— `cnt` for **contents** (see 5.2)

— `dir` for **directories** (see 5.3)

— `rev` for **revisions** (see 5.4)

— `rel` for **releases** (see 5.5)

— `snp` for **snapshots** (see 5.6)

The fourth field is the *intrinsic identifier* of the object. This is a hex-encoded (using lowercase ASCII characters) hash value computed from the content and relevant metadata of the object.

5.2 Contents

A *content* is a byte sequence, typically, the content of a file. For this type of object, the intrinsic identifier is the hash of it; that is, the SHA1 of the byte sequence obtained by concatenating:

— the ASCII string `"blob"` (4 bytes),

— an ASCII space,

— the length of the content as ASCII-encoded decimal digits,

— a NULL byte,

— and the actual content of the file.

No metadata is used for this type of object (in particular, notice that there is no file *name* mentioned here).

As an example, `swh:1:cnt:94a9ed024d3859793618152ea559a168bbcbb5e2` is the SWHID computed from the full text of the GPL3 license.

5.3 Directories

Directories are data structures commonly used in hierarchical file systems to group together files and other directories, and to hold relevant metadata about them, in the form of directory entries.

This specification adopts the same convention as the Git version control system, and only takes into account as metadata the name of the directory entries (as a sequence of arbitrary bytes, excluding ASCII '/' and the NULL byte) and a simplified representation of the access rights.

The names of entries in a directory shall be distinct from one another.

In order to compute the intrinsic identifier of a directory, it is necessary to compute first the SWHID of each object listed in that directory.

Then a serialization of the directory is created, as follows:

1. sort the directory entries using the following algorithm:
    a. for each entry pointing to a *directory*, append an ASCII '/' to its name
    b. sort all entries using the byte order of their (modified) name
2. for each entry, with a given *name* (unmodified), add a sequence of bytes composed of
    a. the normalized access rights, encoded as a sequence of ASCII-encoded octal digits ('100644' for regular [i.e., non-special] files, '100755' for executable files, '120000' for symbolic links, and '40000' for directories),