

EIDR: ID FORMAT

Ver. 1.02

30 January 2012



Copyright © 2012 by the Entertainment ID Registry Association

EIDR: ID Format.

The content of this manual is furnished for information use only and is subject to change without notice and should not be construed as a commitment by the Entertainment ID Registry Association. The Entertainment ID Registry Association assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Products and company names mentioned may be trademarks of their respective owners.

Feedback on this document can be sent to support@eidr.org

TABLE OF CONTENTS

1	Canonical Form	4
1.1	Standards References.....	4
1.2	Representation.....	4
2	Standard Alternate Representations	5
2.1	Binary.....	5
2.1.1	<i>Compact Binary</i>	5
2.1.2	<i>Full Binary</i>	6
2.1.3	<i>Other Binary</i>	6
2.2	Canonical, no hyphens	6
2.3	Short DOI	6
2.4	URN	7
2.5	URI.....	7
3	Potentially Lossy Representations	7
3.1	Non-standard Binary.....	8
3.2	Non-standard URN	8
4	Size summary.....	9

1 Canonical Form

1.1 Standards References

The DOI syntax is a NISO standard. See the DOI Handbook, Appendix 1, [ANSI/NISO Z39.84-2000 Syntax for the Digital Object Identifier](#).

The DOI System is an ISO Standard, ISO 26324.

EIDR IDs are fully compliant with the NISO and ISO specifications.

If new standard representations of a DOI are developed, they will be included in future versions of this document.

1.2 Representation

This is the only representation that can properly be called a DOI or an EIDR ID. The canonical form of an EIDR ID is

10.5240/XXXX-XXXX-XXXX-XXXX-XXXX-C

Standard nomenclature is:

- 10.5240 is the DOI prefix for an EIDR asset record ('prefix' for short.) The '10' indicates that this handle is a DOI.
- 5240 is the sub-prefix. It tells the DOI system which registry is responsible for the ID. A registry is usually responsible for multiple sub-prefixes.
- XXXX-XXXX-XXXX-XXXX-XXXX-C is the DOI suffix ('suffix' for short.) This is what the individual Registry uses to find the metadata associated with the ID.

For the suffix,

- X is a hexadecimal digit
- C is the ISO 7064 Mod 37,36 check character. The check is computed as Mod 37,36 rather than Mod 17,16 to allow for future extension of the ID format.
- The check character is computed only over the DOI suffix. It does not include the prefix because if the prefix is wrong, it is highly probable that the DOI will go to an incorrect resolution system anyway. The EIDR registry separately validates the prefix of any DOI sent through its API.

Normalization:

- EIDR IDs are normalized to upper case on input and output from the registry.
- The DOI proxy accepts resolution requests for mixed-case EIDR IDs
- NOTE: Contrary to the EIDR Technical Overview, the hyphens in the ID are required in current versions of the Registry.

2 Standard Alternate Representations

Even though the Registry and DOI Proxy do not accept other forms of the ID, it is sometimes necessary or convenient to represent the ID in a more compact representation. There are three requirements for alternate forms:

- They do not lose any information. Information that can be regenerated without loss is:
 - The '10.', since DOI handles always start with '10.'
 - The '/', as long as you know where to put it
 - The '-' characters, as long as you know where to put them in the suffix
 - The checksum, which can be recomputed.

Information that cannot be regenerated is:

- The sub-prefix. EIDR already uses 4 sub-prefixes (one each for assets, parties, users, and video services and networks) and may in the future allocate others for the same or different purposes. We expect the number of sub-prefixes for assets to be small, but applications must not assume that there will always be just 1. It is not safe to jettison the sub-prefix, but it is safe to map it down to a handful of bits.
- The suffix (excluding the checksum)
- All the systems that exchange a particular non-canonical form of an EIDR ID agree on and recognize the format.
- All systems convert the non-canonical form of the ID to the canonical form when communicating with systems that are not 'in the know' (such as the Registry itself, or other third party and DOI-based applications.)

The remainder of this section covers some standard alternate representations of an EIDR ID. Other lossless formats that are commonly used or are required for using EIDR within other standards will be added as they emerge.

2.1 Binary

2.1.1 Compact Binary

This is for use in systems where space is at a premium, and takes 96 bits:

- 16-bit sub-prefix: Interpret the sub-prefix as a number, and convert it to binary
- 80 bits: The suffix, without the checksum, represented as 20 nibbles/10 bytes.

Converting this to the canonical representation entails:

- Starting with the string '10.'
- Appending the decimal representation of the value of the first 16 bits

- Appending a '/'
- Appending the value of each nibble as a hex digit, adding a '-' after every fourth digit
- Computing the check digit and appending it to the string

2.1.2 Full Binary

This is larger than Compact Binary, but still smaller than the full representation.

- 8 char prefix '10.5240/'
- 80 bit binary representation of the suffix (minus checksum)
- 8-bit ASCII representation of the check character

The conversion to canonical form is simpler than for compact binary, and is left as an exercise for the reader.

2.1.3 Other Binary

The attentive reader will also note that there are several possible gradations between the compact binary and full binary representations; one example is leaving off the '10.' and the check character, leaving just the sub-prefix and the suffix excluding the check digit.

2.2 Canonical, no hyphens

This is the canonical form with all the hyphens removed, which saves 5 bytes.

2.3 Short DOI

The International DOI Foundation provides the shortDOI service, which generates a very compact representation of a DOI. You can think of it as tinyurl for DOIs. The original DOI is mapped into a string of three or more characters, taken from this 27-character set:

bcdfghjkmnpqrstvwxyz23456789

The size of the character set means that each character of a shortDOI needs 5 bits to encode. 7 such characters will support 10+ billion IDs¹. 35 bits is an awkward number², but even encoding each one as a full byte only takes 56 bits. To be even more future-proof, an application could plan for up to 8 characters in a shortDOI.

Instructions for manual and automated use of the service are available at shortdoi.org. For example, if you enter 10.5240/5FD4-FEE1-22F5-583E-FECC-O, you get back 10/f77, which can be resolved with these³:

¹ Although this space is used by all DOI registries, not just EIDR

² The world never really adopted the DEC-20's 36-bit word....

³ http://dx.doi.org/10/f77?ignore_aliases returns information about the shortDOI itself rather than resolving the DOI to which it refers.

- <http://doi.org/f77>
- <http://dx.doi.org/10/f77>

Both of these take the usual DOI proxy resolution flags as described in the *EIDR API Overview*, e.g. <http://dx.doi.org/10/f77?locatt=type:Simple>

Although the shortDOI is even shorter than the compact binary representation, it requires calling an external system to reconstitute the shortDOI as a canonical EIDR ID.

2.4 URN

Many systems like to use URNs as IDs. EIDR IDs have a / in them, which is not a legal character in a URN. The standard way to deal with this is to escape the / in the EIDR ID as %2F, so an EIDR ID represented as a URN might look like:

```
urn:schemename:eidr:10.5240%2F5FD4-FEE1-22F5-583E-FECC-O
```

This is a fully legal URN, and the canonical EIDR ID can be extracted from it with no outside information. However, the use of escaped sequences can be prone to implementation errors in some contexts, e.g. in URLs which require their own escaping, so in some contexts, it may be preferable to use other forms of the ID.

2.5 URI

There are two standard ways to represent an EIDR ID as a URI.

- DOI is a registered URI within the info-URI namespace ([IETF RFC 4452, the "info" URI Scheme for Information Assets with Identifiers in Public Namespaces](#)). For example, `info:doi:10.5240/5FD4-FEE1-22F5-583E-FECC-O` is a legal URI.
- You can represent a DOI (and hence an EIDR ID) as a URI using the DOI proxy. <http://dx.doi.org/10.5240/5FD4-FEE1-22F5-583E-FECC-O> is a legal URI.

There is also a non-standard way to represent an EIDR ID as a URI, included here because, although it is non-standard, it is lossless.

- The use of the lowercase string “doi” complies with the IETF specification, RFC 3986, for representation as a URI (uniform resource identifier). This means that although `doi` is not a scheme registered with IANA, `doi:10.5240/5FD4-FEE1-22F5-583E-FECC-O` is at least syntactically legal as a URI.

3 Potentially Lossy Representations

Although EIDR IDs should never be compressed in a way that loses information, some techniques run the risk of data loss even if they are not inherently lossy. However, some applications may require such representations for reasons of compatibility with existing processes, APIs, or databases.

Such representations are safe as long as they are used carefully in closed environments – the risk of information loss increases when the representation is used outside of the system within which it is defined.

Applications that use potentially lossy representations for EIDR IDs must ensure that if an ID is ever presented to a user or in a document as a DOI or EIDR ID, the ID should be presented in the canonical format. Of course, when presenting the ID within the originating environment, the representation is entirely up to the defining system. When the ID is used in other contexts outside the closed environment, it is important to use a canonical format in order to ensure successful DOI resolvability and interoperability with other systems.

The examples in this section are hypothetical. Future versions of this document will include potentially lossy formats that come into common use. Such formats will generally be specific to a particular application or ecosystem.

3.1 Non-standard Binary

Compact binary could be made even more compact by replacing the 16-bit sub-prefix with a single byte, which is then used to index a table of prefixes. Although this is smaller, it requires extra information (the mapping table), and so an ID of this form cannot be reconstituted without knowing the extra information.

It may be appropriate for entirely closed systems, but should be used only as a last resort and is strongly discouraged in other cases. For example, this case might arise with media for connected devices that have legacy-driven space constraints. Such devices would have to talk to EIDR-cognizant systems through an intermediary (such as a dedicated server) that knew how that class of devices dealt with EIDR prefixes.

3.2 Non-standard URN

Some systems, due to bugs or historical accidents, may not accept even an escaped ‘/’ in a URN. There are two approaches for dealing with this.

- Alternate translation: Translate the / to something else, for example the underscore character. The scheme name for this non-standard escaping *should not* be the same name used for URNs that escape the / in the standard way. For example:

```
urn:schemename:eidr-u:10.5240_5FD4-FEE1-22F5-583E-FECC-O
```

This can also be used when implementations do not properly handle “%” escaping – the important thing to remember is that none of these cases should use undifferentiated `eidr` (which must be reserved for use in a standard URN format) as a name or sub-name.

Although this form requires some external information (which characters get replaced and the substitution character(s)) the pattern of an EIDR ID is simple enough for an application to turn this back into the canonical form pretty easily.

- Truncated URN: Some URN-based systems may have length limits, or problems with any special characters at all, requiring complete removal of the prefix. In that case,

it may be necessary to have a new URN scheme. URN schemes that escape, remove, or replace characters in different ways should each define a different scheme (or sub-scheme) name. For example, this scheme removes the prefix but leaves the hyphens:

```
urn:schemename:eidr-s:5FD4-FEE1-22F5-583E-FECC-0
```

For schemes like this to be lossless, these things must be true:

- It must be known that `eidr-s` indicates that the prefix is 10.5240
- If the representation removes punctuation, such as the hyphens, this must be known to reconstruct the full canonical form.
- The scheme name and suffix should always travel together to ensure that there is enough information present to know which prefix mapping and punctuation replacement to use.
- Each EIDR prefix has a separate indicator in the scheme. In the example, `eidr-s` implies the prefix is 10.5240; a new EIDR prefix would need a new indicator, e.g. `eidr-s1`.

Unless all of these are true, there is a risk that an application will not know what to do if it needs to turn this URN-based ID into a canonical EIDR ID.

4 Size summary

Format	Size	Needs external information?
Canonical, with hyphens	34 bytes	no
URN, standard	36 bytes, plus length of scheme identifier	no
info:doi	43 bytes	no
URN, replace '/' rather than encode it as %2F	34 bytes (min), plus length of scheme identifier	yes
Canonical, no hyphens	29 bytes	no
Full binary	19 bytes (8 bytes + 80 bits + 1 byte)	no
Compact binary	12 bytes (96 bits)	no
shortDOI	8 bytes	yes
shortDOI	7 bytes	yes

