

EIDR: ID FORMAT

Ver. 1.3

July 21, 2015



Copyright © 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	INTRODUCTION.....	5
1.1.1	Document Version Information	5
2	CANONICAL FORM	5
2.1	Standards References	5
2.1.1	SMPTE.....	5
2.1.2	IETF and IANA	6
2.1.3	ISO and NISO.....	6
2.1.4	DOI Proxy	6
2.2	Representation	6
3	STANDARD AND LOSSLESS ALTERNATE REPRESENTATIONS.....	8
3.1	Binary.....	8
3.1.1	Compact Binary	8
3.1.2	Full Binary	9
3.1.3	Other Binary	9
3.2	Base64URL	9
3.3	Canonical, no hyphens.....	10
3.4	Short DOI.....	10
3.5	URN	11
3.5.1	Standard URN.....	11
3.5.2	DOI URN Format	11
3.5.3	Including EIDR IDs in non-EIDR URNs.....	12
3.6	URI.....	12
3.7	Use in Filenames.....	13
3.7.1	Human-intelligible	13
3.7.2	Not Human-intelligible.....	13
4	POTENTIALLY LOSSY REPRESENTATIONS	13
4.1	Classes of Potentially Lossy Representations	14
4.1.1	Non-standard Binary	14
4.1.2	Non-standard URN.....	14
4.1.3	Non-standard Use in Filenames (EIDR-F)	15

4.2	Current Uses	16
4.2.1	EIDR-S.....	16
4.2.2	EIDR-X	16
5	SIZE SUMMARY	17

1 Introduction

This document describes various ways to represent EIDR IDs, and provides example use cases for some of those representations. It assumes a basic knowledge of the EIDR system.

1.1.1 Document Version Information

If new standard representations of an EIDR ID are developed, they will be included in future versions of this document.

Changes in Version 1.3:

- Add RFC 7302 and IANA references for URN
- Deprecate escaped URN form
- Separate sections for urn:eidr and urn:doi
- Add https for DOI Proxy
- Make EIDR-F normative, rather than a mere example
- Update EIDR-S and EIDR-X sections

New in Version 1.2:

- Add SMPTE references
- Explicit character set restriction
- Canonical formats for Party and Service IDs
- Compact Binary format for Party and Service IDs.
- Base64URL Encoding of Compact Binary
- Clarify description of Full Binary and Compact Binary
- EIDR URN format for DOI proxy

New in Version 1.1

- New DOI URN Format
- New preferred name for the DOI proxy
- EIDR-S
- EIDR-X

2 Canonical Form

2.1 Standards References

2.1.1 SMPTE

The Society of Motion Picture and Television Engineers (SMPTE) has defined text and binary representations for both DOI names and EIDR Content IDs in "Digital Object Identifier (DOI) Name and Entertainment ID Registry (EIDR) Identifier Representations", issued as SMPTE Recommended Practice 2079 (generally referred to as SMPTE RP 2079).

SMPTE RP 2079 is the standards body document that defines the canonical form of an EIDR ID. It also defines URI, full binary, and compact binary representations. All of these are compatible with the equivalent definitions in this document.

2.1.2 IETF and IANA

[RFC 7302](https://tools.ietf.org/html/rfc7302)¹ describes the URN form of an EIDR Content ID, and ‘eidr’ is registered as its formal namespace identifier at the [IANA Registry of URN Namespaces](http://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml).²

2.1.3 ISO and NISO

EIDR is a specialized form of a DOI. The DOI System is standardized in ISO 26324. The following have useful background and information for users of DOI-based identifiers:

- DOI Handbook – http://www.doi.org/doi_handbook/TOC.html
- DOI Factsheet – <http://www.doi.org/factsheets/DOIIdentifierSpecs.html>

The DOI syntax is a NISO standard. See the DOI Handbook, Appendix 1, [ANSI/NISO Z39.84-2000 Syntax for the Digital Object Identifier](http://www.doi.org/ANSI/NISO_Z39.84-2000_Syntax_for_the_Digital_Object_Identifier).

EIDR IDs are fully compliant with the ISO and NISO specifications. The canonical form of an EIDR Content ID is its DOI form as specified in SMPTE RP 2079.

2.1.4 DOI Proxy

The DOI Proxy is <http://doi.org>. See “[EIDR and the DOI Proxy](#)” for details about general EIDR resolutions; supported types; and content negotiation policies. For example, the DOI Proxy will resolve the EIDR ID for “The Great Train Robbery” as follows:

<http://doi.org/10.5240/7791-8534-2C23-9030-8610-5>

The DOI Proxy supports https as well as http.

NOTE: The above replaces the old proxy at <http://dx.doi.org>. The old proxy is still available, but users of the proxy service should switch to the new URL as soon as possible. Formal specifications with references to the old proxy should update them to the new proxy in their next revision.

2.2 Representation

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

Type	Format
Content ID	10.5240/XXXX-XXXX-XXXX-XXXX-XXXX-C
Party ID	10.5237/XXXX-XXXX

¹ <https://tools.ietf.org/html/rfc7302>

² <http://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml>

Service ID	10.5239/XXXX-XXXX
------------	-------------------

Standard nomenclature is:

- The string before the “/” (e.g. 10.5240) is the DOI prefix for an EIDR record (“prefix” for short.) The “10” indicates that this Handle³ is a DOI.
- The string between the “.” and the “/” (e.g. 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. The EIDR Registry is responsible for 5237, 5239, and 5240.⁴
- 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.

Character Set:

- The EIDR prefix and suffix are restricted to the following set:
 - ASCII alphabetic characters A-Z and a-z (0x41 – 0x51, 0x61 - 0x71)
 - ASCII digits 0-9 (0x30 – 0x39),
 - “-” (0x2D), “.” (0x2E), and “_” (0x5F)“_” is not legal for EIDR Content IDs (prefix 10.5240) and is not currently used with other EIDR IDs either.

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.

³ The Handle System provides location-independent resolution. It forms part of the DOI infrastructure, but DOI applications do not need to know about the implementation details. However, interested parties can consult <http://www.doi.org/factsheets/DOIHandle.html>.

⁴ It is also responsible for 10.5238, indicating an EIDR user ID, which cannot be used outside of the context of the EIDR Registry except for perfunctory resolution.

3 Standard and Lossless Alternate Representations

Even though the Registry and DOI Proxy do not accept other forms of the ID (with the exception of the DOI Standard URN forms for the Proxy, *vide infra*) it is sometimes necessary or convenient to present 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 one for a particular kind of ID. 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.

3.1 Binary

3.1.1 Compact Binary

Compact binary applies to all EIDR IDs, and is used when space is at a premium. This form always takes 96 bits:

- 16-bit sub-prefix: Interpret the sub-prefix as a number, and convert it to binary. For example, “10.5240” converts to 0x1478.
- 80 bits:

- For Content IDs, the suffix, without the checksum and dashes, represented as 20 nibbles/10 bytes.
- For Party IDs and Service IDs, 20 nibbles/10 bytes, with the suffix in the first 8 nibbles and 0 in the remainder. This is equivalent to right-padding the suffix with '-0000-0000-0000' and then converting.

For example, the compact binary form for the Content ID 10.5240/F85A-E100-B068-5B8F-B1C8-T is (in hexadecimal) 0x1478F85AE100B0685B8FB1C8 and compact binary for the Party ID 10.5237/9DD9-E249 is 0x14759DD9E249000000000000.

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 “/”.
- If the first 16 bits are a Content prefix,
 - 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.
- If the first 16 bits are a Service or Party prefix
 - Appending the value of each of the first 8 nibbles as a hex digit, adding a “-” after the fourth one.

3.1.2 Full Binary

This format is defined only for Content IDs. It is larger than Compact Binary, but still smaller than the full representation.

- 8 char prefix “10.5240/” encoded as US-ASCII.
- 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.

3.1.3 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.

3.2 Base64URL

Some applications need to include an EIDR ID as a URL parameter. The recommended practice is to

- Convert the EIDR ID into the compact binary format
- Use base64URL encoding of the result (as defined in <http://www.ietf.org/rfc/rfc4648.txt>). Since the input is 96 bits long, the output will never have a pad character.

For example:

- EIDR ID: 10.5240/F85A-E100-B068-5B8F-B1C8-T
- Compact Binary: 0x1478F85AE100B0685B8FB1C8
- Base64URL: FHj4WuEAsGhbj7HI

Or for a Party ID

- EIDR ID: 10.5237/9DD9-E249
- Compact Binary: 0x14759DD9E249000000000000
- Base64URL: FHwd2eJJAAAAA

3.3 Canonical, no hyphens

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

3.4 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 a TinyURL for DOIs. The original DOI is mapped into a string of three or more characters, taken from this 27-character set:

bcdfghjklmnpqrstvwxyz23456789

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 IDs⁵. 35 bits is an awkward number⁶, but even encoding each one as a full byte only takes 56 bits. To be even more future-proof, an application should 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/F85A-E100-B068-5B8F-B1C8-T, you get back 10/53p, which can be resolved with these:

- <http://doi.org/53p>
- <http://doi.org/10/53p>

Both of these take the usual DOI proxy resolution flags⁷ as described in *EIDR and the DOI Proxy*, e.g. <http://doi.org/10/53p?locatt=type:Simple>

⁵ Although this space is used by all DOI registries, not just EIDR.

⁶ The world never really adopted the DEC-20's 36-bit word.

⁷ Additionally, http://doi.org/10/f77?ignore_aliases returns information about the shortDOI itself rather than resolving the DOI to which it refers.

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.

3.5 URN

Many systems like to use URNs as IDs. EIDR IDs have a / in them, which is not a legal character in a URN. There are two lossless ways of dealing with this.

3.5.1 Standard URN

The standard URN form of an EIDR Content ID, specified in [RFC 7302] is

```
urn:eidr:EIDR-PREFIX:EIDR-SUFFIX
```

This is currently only defined for the 10.5240 prefix. For example, the URN form of the EIDR ID for “Mr. Edison at Work in His Chemical Laboratory” is

```
urn:eidr:10.5240:B17A-4DAF-9496-C586-C1F5-9
```

Note: In addition to taking the canonical form the DOI Proxy accepts this standard urn:eidr form, e.g.

<http://doi.org/urn:eidr:10.5240:B17A-4DAF-9496-C586-C1F5-9>

The EIDR API only accepts the canonical form of the ID, not the URN form.

3.5.2 DOI URN Format

The DOI proxy supports a URN-like form which complies with URN character set restrictions but does not have a registered URN prefix. This form can be used in DOI-based applications where no other representations are suitable, for example a system requiring a URN-ish ID for an identifier that also uses other URNs including non-EIDR DOIs.⁸

To do this, replace the “/” between the prefix and suffix with a “:”, giving urn:doi:10.sub-prefix:suffix.⁹ An example using an EIDR Content ID is:

```
urn:doi:10.5240:3466-F12C-391A-D60B-206B-Y
```

The DOI Proxy accepts this form as well as the canonical form for resolutions¹⁰, e.g.

<http://doi.org/urn:doi:10.5240:3466-F12C-391A-D60B-206B-Y>

⁸ For systems that only need EIDR-based URNs, use RFC 7302.

⁹ Any other “/” characters in a DOI name have to be escaped to make the URN legal. Users of EIDR DOIs do not have to worry about this.

¹⁰ See <http://www.doi.org/factsheets/DOIIdentifierSpecs.html> for more details.

3.5.3 Including EIDR IDs in non-EIDR URNs

3.5.3.1 Preferred – Using RFC 7302

Some applications need to include an EIDR ID in a URN. The preferred method is to use `eidr-prefix:eidr-suffix`, the EIDR-NSS as specified in [RFC 7302], e.g.,

```
urn:myscheme:eidr:10.5240:CA51-02D0-3269-23C9-DB5A-E
```

And when the EIDR ID in such representations needs further specialization (see EIDR-X below), the ‘eidr’ component of the URN should be changed to something different, e.g., for a hypothetical `urn:trackid` namespace,

```
urn:trackid:eidr-x:10.5240: B17A-4DAF-9496-C586-C1F5-9:aud:en
```

3.5.3.2 Deprecated – Escaped Encodings

A generic way of handling the problem is to escape the / in the EIDR ID as %2F, so an EIDR Content ID represented as a URN might look like:

```
urn:schemename:eidr:10.5240%2FCA51-02D0-3269-23C9-DB5A-E
```

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

Resolution of this form is dependent on the implementation of `schemename` but the DOI Proxy can resolve an escaped EIDR ID in three ways:

<http://doi.org/10.5240%2FCA51-02D0-3269-23C9-DB5A-E>

<http://doi.org/urn:eidr:10.5240%2FCA51-02D0-3269-23C9-DB5A-E>

<http://doi.org/urn:doi:10.5240%2FCA51-02D0-3269-23C9-DB5A-E>

3.6 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/CE43-9B6A-2C41-35C3-42CA-V` is a legal URI.
- You can represent a DOI (and hence an EIDR ID) as a URI using the DOI proxy.¹¹ <http://doi.org/10.5240/CE43-9B6A-2C41-35C3-42CA-V> 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.

¹¹ See http://www.doi.org/doi_handbook/3_Resolution.html#3.7.3 for more details.

- 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/CE43-9B6A-2C41-35C3-42CA-V` is at least syntactically legal as a URI.

3.7 Use in Filenames

It may be necessary or useful to embed an EIDR ID in a filename. Some systems use filenames that convey information to human readers, and some systems use filenames that are just strings of characters.

3.7.1 Human-intelligible

The canonical representation of an EIDR ID contains a `/`, which is a special character in some operating systems. It also contains a `.` which is usually a legal character, but some systems are ill at ease with filenames that contain more than a single dot.

Therefore, when using an EIDR ID as a component of a filename, you should replace both the slash and the dot with `-`.

For example, `10.5240/7481-838B-59CA-63D0-B9A8-E` becomes `10-5240-7481-838B-59CA-63D0-B9A8-E`

3.7.2 Not Human-intelligible

If there is no requirement for human readability, the `base64url` format is well-suited for use in filenames.

4 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 is presented in the canonical format¹². 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

¹² Some applications may want to communicate the ID as a URN or URI, in which case the respective standard representations should be used.

in order to ensure successful DOI resolvability, interoperability with other systems, and successful conversion into a form recognized by the EIDR registry.

4.1 Classes of Potentially Lossy Representations

EIDR members have expressed interest in two types of non-standard representations – binary and URN. So far, there are no implementations of potentially lossy binary forms, but see the next section for two implementations of alternate URN-based formats.

Future versions of this document will include potentially lossy formats as they come into common use. Such formats will generally be specific to a particular application or ecosystem.

4.1.1 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), so an ID of this form cannot be reconstituted without knowing the extra information. For example, 3 bits can encode 8 prefixes, 4 bits can encode 16 prefixes, and so on.

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 or a translation library) that knew how that class of devices and media dealt with EIDR prefixes.

4.1.2 Non-standard URN

Some systems, due to bugs or historical accidents, may not accept even an escaped “/” in a URN. For similar reasons, some systems may not accept the standard EIDR URN format because of the extra colon. 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-undr: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 or sub-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 encodes the prefix in the scheme name and leaves the hyphens:

`urn:schemename:eidr-5240:5FD4-FEE1-22F5-583E-FECC-O`

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

- It must be known that `eidr-5240` 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-5240` implies the prefix is 10.5240; a different EIDR prefix would need a new indicator, e.g. `eidr-5239`.

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 proprietary URN-based ID into a canonical EIDR ID.

4.1.3 Non-standard Use in Filenames (EIDR-F)

Some systems may have length limits for filenames, or pre-existing requirements or naming conventions. In these cases, it may be necessary to replace the standard EIDR prefix with something shorter or more locally appropriate. It is essential that there be some kind of prefix or preamble in the name, since there is no guarantee that there will not be other EIDR prefixes in the future.

For example, the EIDR-F format for filenames is defined to be

`"EIDR-F-"EIDR-suffix`

e.g. `EIDR-F-7481-838B-59CA-63D0-B9A8-E`.

For this this kind of encoding to be lossless, the following must be true:

- It must be known, e.g., that `EIDR-F` indicates that the prefix is 10.5240
- If the representation removes or changes the punctuation, such as the hyphens, this must be known in order to reconstruct the full canonical form.
- The new prefix and the 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. `EIDR-F` implies the prefix is 10.5240; a different EIDR prefix would need a new indicator, e.g., `EIDR-G`.

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 proprietary URN-based ID into a canonical EIDR ID. Use of just ‘EIDR’ as the mapped prefix is strongly discouraged; it will run afoul of the introduction of new prefixes, and risks confusion in applications that accidentally come across some other application’s filenames.

4.2 Current Uses

4.2.1 EIDR-S

EIDR-S is a special URN-encoded form of the EIDR ID developed for use as a DECE (Digital Entertainment Content Ecosystem) Content Identifier for the UltraViolet digital content system; it is now also used in the EMA (Electronic Merchants Association) Content Availability Metadata spec.¹³ The DECE Content ID spec does not allow slash (“/”) characters

The EIDR-S format replaces the “10.5240/” prefix in a standard EIDR content ID with “eidr-”. The result is appended to an implementation-specific namespace or prefix. For example:

- urn:dece:cid:eidr-s:1E63-2E9A-11AB-FE88-1B89-M
- urn:dece:alid:eidr-s:50A5-34E1-4FFF-0BBD-17C9-G
- urn:dece:apid:eidr-s:8BAD-E17A-BD9D-0B5F-C6F8-R
- md:cid:eidr-s:1012-7947-21D5-9D24-CC5F-H

4.2.2 EIDR-X

EIDR-X is an ID with extended version encoding in the form of one or more unique alphanumeric suffixes separated from the standard EIDR suffix by colons. It is used when a workflow requires a distinction between two objects that cannot normally be made using the EIDR content ID structure. For example, if an UltraViolet content provider required two different ALIDs for distributing the same EIDR Edit as part of different offers, multiple ALIDs could be created from the same EIDR Edit using the EIDR-X format.

EIDR-X is used by DECE, EMA Avails, and SCSA.

An EIDR-X is constructed like an EIDR-S, except a suffix is appended to the suffix of the EIDR ID. For example:

- urn:dece:cid:eidr-x:1E63-2E9A-11AB-FE88-1B89-M:Sony
- urn:dece:alid:eidr-x:50A5-34E1-4FFF-0BBD-17C9-G:UK
- urn:dece:apid:eidr-x:8BAD-E17A-BD9D-0B5F-C6F8-R:vudu
- md:availalid:eidr-x:1012-7947-21D5-9D24-CC5F-H:aug_Europe
- tag:scsallc.com,2014:CMPIID:eidr-x:F381-038C-F777-CDBC-A61F-D:0
- tag:scsallc.com,2014:PCID:eidr-x:F381-038C-F777-CDBC-A61F-D:cfhd:0
- tag:scsallc.com,2014:SAPID:eidr-x:F381-038C-F777-CDBC-A61F-D:vid.cfxd.avc3:0

¹³ <http://movielabs.com/md/avails/>

NOTE: EIDR does not manage EIDR-X suffixes as part of its controlled vocabulary. It is up to each adopter of EIDR-X to develop rules, practices, and systems that enforce uniqueness within a particular domain. For example, the UltraViolet Coordinator ensures that two different parties do not use the same EIDR-X.

5 Size summary

Content ID Format	Size	Needs external information?
Canonical, with hyphens	34 bytes	No
URN, standard format	34 bytes, plus length of scheme identifier (e.g. urn:eidr or urn:doi)	No
URN, escaped	36 bytes, plus length of scheme identifier	No
info:doi	43 bytes	No
http://doi.org/...	50 bytes	No
URN, replace "/" with a single character rather than encode it as %2F	34 bytes, 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
Base64URL	16 bytes	No
shortDOI	8 bytes	Yes