INTERNATIONAL LASER
DISPLAY ASSOCIATION

Technical Committee

The ILDA Digital Network

File Format Specification

Standard Identifier: IDN-File

Draft 2021-01-15

Table of Contents

1 Introduction

This technical standard is part of the International Laser Display Association Digital Network (ILDA Digital Network, IDN) protocol suite and describes the format for storing sequences of IDN-Stream messages into files. These sequences can be used to exchange image data frames, animation sequences or entire shows including multiple laser projectors, audio, video and lights.

1.1 Nomenclature and Structure

Throughout this document, the word “SHALL” is used in capitals to stress required conformance. The word “SHOULD” in capitals indicates suggested conformance.

1.2 Octets / Bytes / Endianness / Bits

Generally, the term “byte” is avoided as it is ambiguous. Alternatively, the term “octet” is used as it unambiguously specifies a size of eight bits.

Octet/Byte Order

For multi octet/byte data words, network byte order (big endian byte order) is used. This specifies that the most significant octet (the octet containing the most significant bit) is stored first (has the lowest address) or sent first. Then the following octets are stored or sent in decreasing significance order, with the least significant octet (the one containing the least significant bit) stored last (having the highest address) or sent last.

Alignment

Alignment of 16 bit (2 octets), 32 bit (4 octets), and for 64 bit formats 64 bit (8 octets) values is kept for this standard. Alignment is important for many CPU implementations as they can’t handle misaligned data accesses (these accesses would have to be done with two bus transfers instead of one).

Bit Numbering

The bit numbering follows the bit significance. This numbering scheme labels the bits with bit 0 referring to the least significant bit. This is useful, because bit n then has a logical value 2^n and corresponds to a left shift.

1.3 Value Representation

In this document, number are expressed in decimal representation unless predeced with 0x, which refers to hexadecimal notation. For example: 127 is 127 decimal whereas 0x1F is 31 decimal. Unless otherwise noted, multi-octet fields are treated as unsigned integers.

Strings

String fields contain non-terminated UTF-8 strings. For strings that are shorter than the field size, the field SHALL be padded with binary zeros (0x00). The fields are specified unterminated to ensure that reading code doesn't rely on a termination character that may be missing because of broken implementations.

File Pointers

[Recheck/Rephrase]

All file pointers SHALL/SHOULD stay in the positive range of their signed data type. This is because seeking in files is usually a signed operation.

1.4 Format and Version

[Note: This document is in draft and not yet officially released/adopted by ILDA]

Depending on the format, files created according to this standard SHALL have the Version field set according to the table found in section Format and Version. The version is introduced for compatibility reasons and therefore mandatory. The version may not be related to the revision number of the standard. Changes of the major version number SHALL indicate significant enhancements or modifications, that could affect compatibility. Changes of the minor version number SHALL indicate changes that do not affect general compatibility.

1.5 Reserved Fields

For padding and future extension reasons, some additional fields have been added. These fields are usually unnamed and shall contain the value of 0. Named additional fields are reserved for a special purpose that may not be fully specified yet.

1.6 Help Improving

This standard has been elaborated and put together very carefully. However, it is complex and may miss explanations or contain ambiguity. Please tell ILDA in case you, as an implementer, come across such passages. This way ILDA can clarify with the next revision and implementations stay compatible.

2 File Format

The IDN-File format has been developed to exchange sequences of IDN-Stream messages between systems. You can record or export an IDN file from one system and play or import the file on any other system that supports this format.

An IDN file consists of a format header section and a linked list of anchors, each describing a sequence of IDN-Stream messages. With this approach, a file can store multiple streams to be played in parallel, shifted, one after another or mixed in any way.

[Note: Default playback just plays Anchor 1 ?]

2.1 Format Header

The format header starts with a common header for all formats and is extended by a variable section depending on the format and/or version fields.

Octet	0	1	2	3	4	5	6	7
0x00	Signature				Title			
0x08	Title							
0x10	Title							
0x18	Title							
0x20	Title							
0x28	Title							
0x30	Title							
0x38	Title							
0x40	Delimiter		Format	Version	0x00000000			
0x48	Creation Time							

Header extension for 64-Bit format, version 1.

Octet	0	1	2	3	4	5	6	7
0x50	Properties FP							
0x58	Editor FP							
0x60	First Anchor FP							
0x68	Free List FP							
0x70	0x0000000000000000							
0x78	0x0000000000000000							

Signature

The file signature. These octets SHALL contain the ASCII characters “IDN:” (0x49, 0x44, 0x4E, 0x3A). Some operating systems use the first four octets of a file as a signature to (besides using the file name extension) identify a file type to be handled by an applicaton. IDN-File readers SHALL use this signature to ensure reading an IDN file.

Title

File title and company name.

The field type is a non-terminated UTF-8 string. For strings that are shorter than the field size, the field SHALL be padded with binary zeros (0x00).

Delimiter

Marker to separate the binary part. The marker conststs of two consecutive End Of File (EOF) characters (0x1A, 0x1A). When being read by purely text oriented applications, processing of such EOF characters may end reading the file (which is binary). There is no guarantee text oriented applications behave that way but having this here makes sense. In addition, IDN-File readers SHALL use this marker as a second check for ensuring the file is an IDN file.

Format and Version

File format and version

[version: major/minor? - uppler/lower nibble?]

Format	Version	
1	1	64-Bit file pointer format

The 64-Bit format uses 64-Bit file pointers. When storing data for multiple devices and different media types (Laser, Video, Audio), and maybe an assembly of a larger scale show, files can get large pretty fast. The 64-Bit pointers avoid file seeking problems when crossing the 2GB boundary.

Creation Time

Creation date and time in milliseconds since January 1, 1970, 00:00:00 UTC.

Properties FP

File pointer to properties (reserved, set 0)

Editor FP

File pointer to editor data (reserved, set 0)

First Anchor FP

File pointer to the first anchor message, containing the description for the first stream. A file may contain multiple anchors, organized in a linked list of anchor messages.

Free List FP

File pointer to list of unused file areas (reserved, set 0)

2.2 IDN Message

This is the common format of all IDN messages

Octet	0	1	2	3
0	Total Size		Content ID	

Total Size

The total size of the message (including the header)

Content ID

The ID of the message content

2.3 IDN-File Message Content IDs

ID	
0x3800	Managed free space
0x3802	Unmanaged free space
0x3815	Movie recovery: Start beacon
0x3816	Movie recovery: Recurring beacon
0x3817	Movie recovery: End beacon
0x3880	64-Bit file: Anchor message
0x3884	64-Bit file: Movie locator

2.4 Recovery Beacon Message

Octet	0	1	2	3	4	5	6	7
0x00	Message Header				CID Copy		CRC16	
0x08...	Random Data							

Message Header

4 octet IDN message header with the contentID set accordingly.

CID Copy

A copy of the content ID of the message for fast checks.

CRC16

The CRC16 of the random data block, standard polynom 0x8005, initial value of 0x0000.

Suggested pycrc commandline args:
python pycrc.py –model crc-16 –algorithm table-driven –generate c -o crc.c

Random Data

Arbitrary length random data. As a suggestion, random data of 24 octets can be used, leading to a total message size of 32 octets.

3 IDN-Stream Anchors

Anchors provide the description for an IDN message stream. Each anchor contains control information, file pointers for the view on the messages and a table for individual tracks. A file may contain multiple anchors. These can for example be used to store data for separate but synchronized sites or overdubs containing data for different types of media.

[basic player: Play first anchor in movie view]

[properties: Comments on use and origin/creator]

3.1 64-Bit Anchor Message

Anchor message for the 64-Bit format, using 64-Bit file pointers.

Octet	0	1	2	3	4	5	6	7
0x00	Message Header				Stream Name			
0x08	Stream Name							
0x10	Stream Name							
0x18	Next Anchor FP							
0x20	Properties FP							
0x28	Editor FP							
0x30	Stream Duration				0x00000000			
0x38	0x0000000000000000							
0x40	0x0000000000000000							
0x48	0x0000000000000000							
0x50	0x0000000000000000							
0x58	0x0000000000000000							
0x60	Movie First FP							
0x68	Movie Last FP							
0x70	Movie Length							
0x78	Movie Locator FP							
0x80...	Track Entry							

The fields starting at offset 0x60 define the movie view for a stream. In case it exists, Movie First FP SHALL point to the first message of the movie stream, Movie Last FP and Movie Length SHALL be valid and the stream SHALL at least have one channel message. In case there is no movie view, all related fields SHALL be set to 0.

Starting at offset 0x80, this header structure is followed by an array of Track entry structures. Each of these structures describes an endpoint for an IDN message stream and is used by recording and playback systems for service name mapping. In movie view, the serviceID of the track entry is used as the referral from IDN-Stream channel configuration routings. In indexed view, each track keeps a tree of messages associated with that track. The table SHALL have a maximum of 255 valid track entries. This is because readers and writers will have to multiplex/demultiplex and interleave the messages. According to the IDN-Stream specification, the limits are 255 individual services per device and 64 channels per stream.

Message Header

4 octet IDN message header with the contentID set accordingly.

Stream Name

The name of the stream.

The field type is a non-terminated UTF-8 string. For strings that are shorter than the field size, the field SHALL be padded with binary zeros (0x00).

Next Anchor FP

File pointer to the next anchor in the list.

Properties FP

File pointer to properties (reserved, set 0)

Editor FP

File pointer to editor data (reserved, set 0)

Stream Duration

The duration of the stream in milliseconds. This field is used by players and editors for cue or display calculations. A value of 0 is only valid for empty anchors (no associated streaming data) and a value of 0xFFFFFFFF SHALL be used to indicate an overflow.

The minimum duration equals the timestamp difference between the first IDN-Stream channel message and the last IDN-Stream channel message. Fractions SHALL be rounded up to the next millisecond to include the full last interval and a duration of 1 millisecond SHALL be the shortest duration. In case a movie view has a locator, the stream duration SHALL match. Please note that the timestamp of channel messages may wrap.

In case the stream ends regularly with void channel close messages, the stream duration SHOULD be the minimum duration. Since the last messages could contain data, that duration SHOULD be taken into account though. The most elegant solution would be for writers to move the channel close and/or append a void channel close message.

In case the duration of the last message needs to be estimated, that duration SHALL be less than 1 second, leading of a maximum stream duration that SHALL be less than the minimum duration plus one second.

Movie First FP

File pointer to the first message of a contiguous message stream (the movie). For future compatibility, the message pointed to may be any IDN message. Not all messages make sense though and players SHOULD choose to abort when reading unknown messages. The stream SHALL contain at least one IDN-Stream channel message though.

Starting the stream with the first IDN-Stream channel message is fine for most writers but starting with a start beacon message makes sense for builtin stream recovery. In this case, the stream SHALL end with an end beacon message (pointed to by Movie Last FP).

In case this field reads 0, the anchor does not provide a contiguous message stream. Other movie view fields SHALL contain 0 in this case as well.

Movie Last FP

File pointer to the last message of the contiguous message stream (the movie). The message can be any IDN message. Most writers will very likely just end the stream with an IDN-Stream channel message. In case the stream was started with a start beacon message, this pointer SHALL point to an end beacon message.

Movie Length

The total length of the contiguous message stream in octets. This takes the size of the last message into account and SHOULD be used for verification purposes by checking against (lastFP - firstFP) + lastMessageTotalSize.

Movie Locator FP

File pointer to the head of the movie locator table. The table which can be split into multiple messages, connected in a single linked list. The locator table is optional but strongly recommended since seeking inside a movie stream may not be possible without it.

3.1.1 Track entry

Tracks define a virtual recording and playback setup for a stream of IDN channel messages. For recording, each incoming channel is mapped to a specific track through its serviceID. The track then receives all channel messages associated with the service. For playback, each track is mapped to a specific service through track name and the corresponding service name in the playback environment. This service then receives all channel messages associated with the track.

Octet	0	1	2	3	4	5	6	7
0x00	SID	SType	Flags	0x00	Track Name			
0x08	Track Name							
0x10	Track Name							
0x18	Properties FP							
0x20	Editor FP							
0x28	Index FP							
0x30	0x0000000000000000							
0x38	0x0000000000000000							

SID

The serviceID. Unless voided (ID set to 0), the ID SHALL be unique for all tracks of an anchor (and therefore the stream). In case the anchor doesn't contain a movie view, voided track entries may be used in an indexed view.

Movie view: IDN-Stream channel messages that contain a configuration header with routing information refer to this serviceID and thus route a channel to a track. In a playback environment, this track can then be routed to a service through track and service names. The use of the serviceID is essential as there wouldn't be an other way for knowing which message should go where. A value of 0 voids the entry for movie views.

Index view: The serviceID is not needed for indexed views because the indexes already group the messages through individual tree structures. In case both views are present, the serviceID SHALL be valid though.

SType

The service type. In case of a valid track, this field SHALL be valid and be set to the expected/required IDN-Stream service type.

Flags

Status flags and options.

Track Name

The name of the track. In playback environments, this name is used to route the track to a playback service.

The field type is a non-terminated UTF-8 string. For strings that are shorter than the field size, the field SHALL be padded with binary zeros (0x00).

Properties FP

File pointer to properties (reserved, set 0)

Editor FP

File pointer to editor data (reserved, set 0)

Index FP

File pointer to index (reserved, set 0)

4 Movie View

The movie view is a storage format where all IDN messages for all media types that belong to an IDN stream are stored in a contiguous and uninterrupted sequence of messages, the movie. This sequence is characterized by a first message, a last message, and a total length. The movie SHALL contain at least one IDN-Stream channel message and is not limited in length.

Since IDN messages and associated data are of variable nature and therefore can't be calculated in terms of duration, a movie view can have a locator for seeking and positioning at at intervals of 1 second. The locator is optional but strongly recommended since seeking inside a movie stream may not be possible without it.

With the end of the movie, all IDN-Stream channels SHOULD be closed. The last close messages for each channel SHOULD not contain a data chunk (void channel close message). Otherwise, stream duration calculation might be imprecise.

A movie stream mainly consists of IDN-Stream channel messages, containing chunks of media data. The movie can however, especially for future compatibility, contain other, non-channel IDN messages. One type of these messages, the recovery beacons is specified in this standard. These messages SHOULD contain informative data but since their use cannot be forseen, players SHOULD choose to abort when reading unknown messages.

4.1 Locator Message

Locator messages contain tables of pointer entries in intervals of 1 second that reference positions in the movie. The pointers SHALL point to IDN-Stream channel messages and the interval between these messages may differ from the interval of the entries. In case the table doesn't fit in one message it can be split across multiple fragments in multiple messages.

The first locator entry in the first table fragment SHALL point to the first IDN-Stream channel message of the movie. The last locator entry in the last table fragment SHALL point to the last IDN-Stream channel message of the movie. The locator table SHALL not have surplus length. This means, the last entry SHALL not be equal to its penultimate.

- In case there is only one channel message, the locator table SHALL contain only one entry. The stream duration (in milliseconds) SHALL be in range 1 <= duration < 1001.
- In case there are only two channel messages with the same timestamp, the locator table SHALL contain two entries. The stream duration (in milliseconds) SHALL be in range 1 <= duration < 1001.
- In case there are only two channel messages with their timestamp differing a difference d greater than 0 seconds and less than 1 second, the locator table SHALL contain two

entries. The stream duration (in milliseconds) SHALL be in range d <= duration < d + 1000.

- In case there are only two channel messages with their timestamp differing exactly 1 second, the locator table SHALL contain two entries. The stream duration (in milliseconds) SHALL be in range 1000 <= duration < 2000.

Please note that all channels SHOULD be closed with the end of the movie, preferably with void channel close messages. In other cases, smart writers might inspect the last channel messages, determine a duration, move the channel close message (eventually adding entries), and set the stream duration to precisely match the movie length.

Following the locator entry point into the movie, IDN-Stream messages for all channels SHOULD contain configuration headers with renewed routing information and decoder configuration. This is optional but strongly recommended to get the stream up and running fast for all open channels after repositioning.

4.1.1 64-Bit Locator Message

This message contains the movie locator for the 64-Bit file format.

Octet	0	1	2	3	4	5	6	7
0x00	Message Header				0x00000000			
0x08	Next Locator FP							
0x10...	Locator Entry FP							

Starting at offset 0x10, the basic header structure is followed by an arbitray (non zero) amount of locator entry file pointers into the movie. Each of these pointers SHALL point to an IDN-Stream channel message.

Message Header

4 octet IDN message header with the contentID set accordingly.

Next Locator FP

File pointer to the next locator table fragment. In case the entire locator table doesn't fit into a single message, it can be split across a linked list of table fragments.

Locator Entry FP

File pointer to an IDN-Stream channel message.

5 Compatibility

[basic player - just play first anchor]

[standard track/service names for default playback mapping]

[list of messages that software implementing this standard shall be aware of]

[Normalization - size, geometry, color]

6 Revision History

6.1 Revision xxx, XXXX 20xx

The initial publication

Contributors (Revision 001)

Dirk Apitz, DexLogic

