INTERNATIONAL LASER
DISPLAY ASSOCIATION

Technical Committee

The ILDA Digital Network

Hello Protocol Specification

Standard Identifier: IDN-Hello

Draft 2022-03-27

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 basic protocol for networking devices that are IDN enabled. These devices can process messages encoded according to IDN-Stream standard. Devices can be for example Laser projectors, visualization tools or converters to other digital or analogue standards like ISP (ILDA Standard Projector).

IDN-Hello adds unified discovery and real time streaming across packetized networks to the IDN protocol suite. With that, it finally standardizes device connections, like to Laser projectors, in the era of networking.

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) and 32 bit (4 octets) values is kept for this protocol. 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.

1.4 Version

When reporting to a client, a server implementing this revision of the IDN-Hello protocol SHALL report a version number of 0.1 **[Note: This document is in draft and not yet officially released/adopted by ILDA]** . The protocol version is introduced for compatibility reasons and therefore mandatory. The protocol 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 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 IDN-Hello Packets

IDN-Hello communication is datagram based. These datagrams are always composed of an IDN-Hello packet header and an optional payload section. They provide communication across packet-switched networks.

2.1 Packet Header

All IDN-Hello packets start with the 4 octets IDN-Hello packet header. Smaller datagrams SHALL be discarded by servers. The packet header contains a command, flags and a sequence number. The sequence number use depends on the command and can be for tracking or packet loss detection.

Octet	**0**	**1**	**2**	**3**
0	Command	Flags	Sequence	

Command

The command tells the receiver what action.is to be carried out. The remaining header fields and header/payload data that eventually follows the packet header are command-dependent.

Management Commands

Command	
0x00	Void/Reserved, no action
0x08	Ping request
0x09	Ping response
0x0C	Client group: Mask retrieval and modification
0x0D	Client group: Result and current mask

Discovery Commands

Command	
0x10	Discovery: Scan network for units
0x11	Discovery: Result, unit identification and status
0x12	Discovery: Ask unit for services
0x13	Discovery: Result, map of supported services

Parameter Commands

Command	
0x20	IDNCMD_SERVICE_PARAMETERS_REQUEST
0x21	IDNCMD_SERVICE_PARAMETERS_RESPONSE
0x22	IDNCMD_UNIT_PARAMETERS_REQUEST
0x23	IDNCMD_UNIT_PARAMETERS_RESPONSE
0x28	IDNCMD_LINK_PARAMETERS_REQUEST
0x29	IDNCMD_LINK_PARAMETERS_RESPONSE

Realtime Streaming (IDN-RT) Commands

Command	
0x40	Realtime stream: Channel message
0x41	Realtime stream: Channel message with acknowledge request
0x44	Realtime stream: Graceful close
0x45	Realtime stream: Graceful close with acknowledge request
0x46	Realtime stream: Abort
0x47	Realtime stream: Acknowledge

Flags

Packet header flags modify general command processing.

Bit	**7 (MSB)**	**6**	**5**	**4**	**3**	**2**	**1**	**0 (LSB)**
	0000				Client Group			

Client Group

The group, a client is sending packets on. To distinguish traffic in setups where multiple clients are used, a group can be assigned to each client or a set of clients. On a server, this group can then be allowed or excluded. By default, all client groups SHALL be allowed on a server. The client group field SHALL be populated and valid in every packet (a group of 0 SHOULD be used as a default but configurability is strongly recommanded). A server SHALL copy the group received with a request into its response packet.

Sequence

The sequence number for the packet. This number is used for the tracking of datagram transmission and command processing. The use of the this number is command dependent and may require uniqueness or monotonic increments.

For commands that follow the request and response scheme, a server implementation SHALL copy the sequence number from the client request into its response, such that the client can identify the response. For such transactions, uniqueness of the sequence number is not required on server side. However, clients can use the sequence number for transaction tracking.

On the contrary, for streaming actions like IDN-RT, the number has to be incremented monotonically (+1) by clients such that a server can track for lost packets.

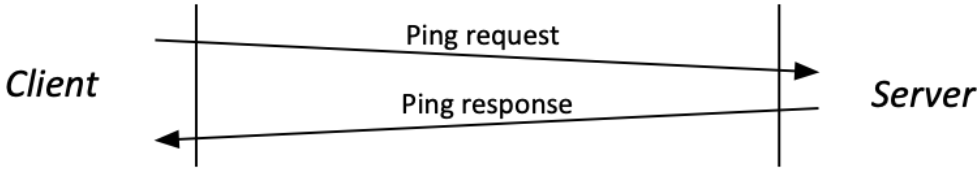
3 Management

The commands in this category are the most basic IDN-Hello commands. They can be used for measurements, connection detection or server configuration.

3.1 Ping

Ping is a administration mechanism used to test the reachability of a host on an network. Although there are usually ping implementations in lower level protocols, this mechanism has been introduced to IDN-Hello for the case that clients may not have full access to these lower level protocols.

A ping sends a packet from the originating host to a destination host that is echoed back to the source. The name comes from active sonar terminology that sends a pulse of sound and listens for the echo to detect objects under water.



Ping can be used for round trip measurements. In this case, the client can for example encode a timestamp into the payload of the request. Since the server SHALL copy the payload exactly to the payload of the response, the client can then calculate the round trip time for a connection.

The payload can be of arbitrary size. This includes empty payloads.

3.1.1 Ping Request

A ping request SHALL be answered with a ping response. A server SHALL copy the request payload to the response payload. The payload SHALL be an exact copy in content and length. This packet type is for servers, clients SHALL not receive it.

Ping request processing SHALL not be affected by the current client group mask.

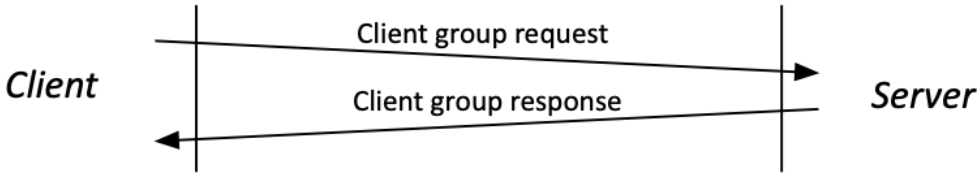
3.1.2 Ping Response

The response to a ping request. Clients SHALL expect an unmodified copy of their ping request payload. This packet type is for clients, servers SHALL not receive it.

3.2 Client Groups

Client groups are a basic mechanism for managing setups where multiple clients share resources on a network. In case a client uses multiple sockets (for example due to multithreading) or distributed hardware (for example due to conversion of multiple channels through separate devices), client groups help servers to separate allowed and excluded traffic by checking the group field in the packet header. It should be noted that this mechanism is based on general cooperation. It does not intend to control misbehaving applications.

By default, all client groups SHALL be allowed on IDN servers. In a setup with multiple clients, a group number is assigned to each client (or client set). At showtime, with the management request, servers can then be told to allow or exclude traffic for specific groups.



3.2.1 Client Group Request

This request retrieves and/or modifies the client group mask on a server and SHALL be answered with a client group response. This packet type is for servers, clients SHALL not receive it.

The group mask controls the acceptance and processing of packets on a server. In case the mask is modified such that clients with open links/sessions get excluded, the links (and the associated sessions) SHALL be voided/closed.

Client group request processing SHALL not be affected by the current client group mask.

For this request, the IDN-Hello packet header is followed by the client group request header:

Octet	0	1	2	3
0	Struct Size	OP Code	Group Mask	
4	Auth Code			
8	Auth Code			
12	Auth Code			

Struct Size

This field contains the size of the header in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

OP Code

The operation code for the request:

Operation	
0x01	Get the group mask
0x02	Set the group mask

Group Mask

For set requests, this field contains the mask for the client groups that shall be enabled. The field is not used for get requests.

A logical ‘1’ SHALL allow processing for a group, a logical ‘0’ SHALL exclude the group from processing. The numbering follows the bit significance. Group 0 refers to the least significant bit (bit 0), Group 15 refers to most significant bit (bit 15).

Auth Code

The authentication code for set operations. This is a very basic access protection for setting the group mask. The code SHALL be a UTF-8 string and SHOULD be configurable.

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). The field is specified unterminated to ensure that reading code doesn’t rely on a termination character that may be missing because of broken implementations.

3.2.2 Client Group Response

The response to a client group request. The packet contains the result code for the operation and the current client group mask. This packet type is for clients, servers SHALL not receive it.

For this response, the IDN-Hello packet header is followed by the client group response header:

Octet	0	1	2	3
0	Struct Size	OP Code	Group Mask	

Struct Size

This field contains the size of the header in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

OP Code

The result code for the request that generated the response:

Operation	
0x00	Successful operation
0xFD	Error: Authentication
0xFE	Error: Operation not supported
0xFF	Error: Invalid request

Group Mask

This field always returns the mask for the currently enabled client groups.

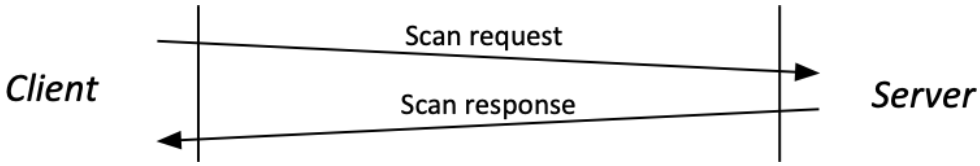
A logical ‘1’ allows processing for a group, a logical ‘0’ excludes the group from processing. The numbering follows the bit significance. Group 0 refers to the least significant bit (bit 0), Group 15 refers to most significant bit (bit 15).

4 Device Discovery

IDN-Hello device discovery allows client applications to find servers, determine server status, unit status and link options and discover about services offered by a server.

4.1 Network Scan

A network scan finds IDN-Hello servers on a network. Depending on the purpose, underlying protocol and routing capabilities, it can be send as a broadcast, network broadcast, multicast or unicast. Further, it can be used for reachability checks and the implementation of plug and play features. A unique unit ID allows for algorithms that can monitor the availibility and absence of hosts.



4.1.1 Scan Request

A scan request SHALL be answered with a scan response. This packet type is for servers, clients SHALL not receive it.

Scan request processing SHALL not be affected by the current client group mask.

For this request, the IDN-Hello packet header is not followed by data.

4.1.2 Scan Response

The response to a scan request. It tells a client application about the protocol version and the server status. For identification, it provides a unique unitID and a host name. This packet type is for clients, servers SHALL not receive it.

For this response, the IDN-Hello packet header is followed by the scan response header:

Octet	0	1	2	3
0	Struct Size	Protocol Version	Status	0x00
4	Unit ID			
8	Unit ID			
12	Unit ID			
16	Unit ID			
20	Host Name			
24	Host Name			
28	Host Name			
32	Host Name			
36	Host Name			

Struct Size

This field contains the size of the header in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

Protocol Version

This field contains the protocol version. The upper 4 bits SHALL contain the major version number and the lower 4 bits SHALL contain the minor version number.

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	Major				Minor			

The protocol version is introduced for compatibility reasons. 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.

Status

Unit status flags and link options.

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	MALFN	Offline	XCLD	OCPD	0	0	0	RT

Malfunction (MALFN)

In case this bit is set, the unit has a permanent malfunction. This could be the detection of blown fuses or broken circuitry. Details could be retrieved with a parameter request.

Offline

In case this bit is set, the unit is currently unavailable. This could be due to bootup, overheat or E-Stop. Details could be retrieved with a parameter request.

Excluded (XCLD)

In case this bit is set, the client's group, is excluded from processing. Traffic other than management or discovery will be answered with errors or be ignored.

Occupied (OCPD)

In case this bit is set, all sessions are occupied by client links. The server won't be able to accept further client links (and allocate sessions for them).

Realtime (RT)

In case this bit is set, the server offers realtime streaming through IDN-RT as part of the IDN-Hello implementation.

Unit ID

The unique ID of the unit. The content of this field SHALL uniquely identify a specific IDN-capable unit. The unitID itself is of variable length. To allow for binary comparisons, regardless of the length of the unitID, the field SHALL be padded with binary zeros (0x00).

The first octet is the length octet. This is the amount of unitID octets that follow (not including the length itself). The second octet is the unitID category. This allows for the distinction between multiple sources of unique identifiers. The category is followed by the identifier octets:

Offset	
0	Number of unitID octets (not including this octet)
1	Identifier category
2..n	Identifier octets

Identifier Categories

Category	
0x01	6 Octets: EUI-48 unique address (known as MAC-Address)
0x10	8 Octets: Xilinx 56 bit DNA with added CRC (x^8 + x^2 + x^1 + 1)

Standardized Representation

The standardized string representations for UnitIDs SHALL start with the identifier category octet, followed by a hyphen '-', followed by the unique identifier octets. Each octet shall be printed as a two digit hexadecimal number (without the leading '0x'). The length octet is implicit in this case and not printed.

The unitID (EUI-48 category) of:

```
0x07 0x01 0x12 0x34 0x56 0x78 0x9A 0xBC
```

shall be printed as: 01-123456789ABC

Host Name

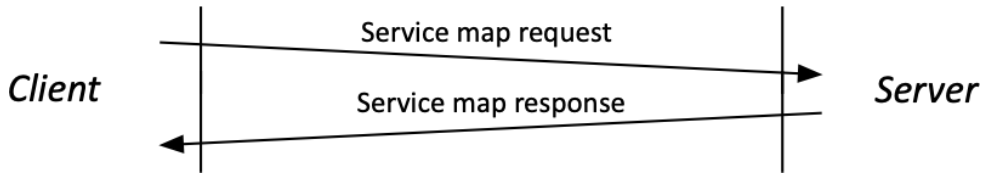
A user friendly name that SHALL identify the unit. The name SHOULD be user configurable. For independent devices, the name could include a brand or model number. This name must not be confused with service names that SHALL name eventually replaceable services (in case of swaps).

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). The field is specified unterminated to ensure that reading code doesn't rely on a termination character that may be missing because of broken implementations.

4.2 Service Map

A service map tells client applications about services that are offered by a server. Generally, these are endpoints of channel message streams (please refer to the IDN-Stream standard).

Further, in case of IDN-Hello servers that act as routers, switches or bridges, the service map tells client applications about the topology such that parameter requests could be directed to specific servers behind that server. In this case, the first IDN-Hello server would act as a relay and route requests to the servers it hides.



4.2.1 Service Map Request

A service map request SHALL be answered with a service map response. This packet type is for servers, clients SHALL not receive it.

Service map request processing SHALL not be affected by the current client group mask. It could however be affected by the client group itself. Servers might be configured such that they offer different services to the individual client groups.

For this request, the IDN-Hello packet header is not followed by data.

4.2.2 Service Map Response

The response to a service map request. It tells a client application about the services offered by a server. This packet type is for clients, servers SHALL not receive it.

For this response, the IDN-Hello packet header is followed by the service map response header:

Octet	0	1	2	3
0	Struct Size	Entry Size	Relay Count	Service Count

The service map response header is followed by the relay table (first) and the service table (second). The relay table has relay count rows of service map entry structs and the service table has service count rows of service map entry structs. Please note that relay entries and service entries share the same struct.

Struct Size

This field contains the size of the response header in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

Entry Size

This field contains the size of the service map entry in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

Relay Count

The number of entries in the relay table.

Service Count

The number of entries in the service table.

4.2.3 Service Map Entry

This struct describes relay entries and service entries. The use of the fields does not generally differ, the purpose is complementing though.

Octet	0	1	2	3
0	Service ID	Service Type	Flags	Relay Number
4	Name			
8	Name			
12	Name			
16	Name			
20	Name			

Service ID

The ID of the service. This field is not used for relay entries.

Relay entry: Must be 0.

Service entry: Must be in range 1..255. Note that a value of 0 is not valid as it would identify the default service.

Service Type

The type of the service. This field can be used by client applications to find appropriate services like for example needed when listing available services for a specific IDN-Stream service mode or IDN-Stream chunk type.

Relay entry: Must be 0 to identify a relay.

Service entry: Should be in range 1..255. A value of 0 marks a void enty that can be used in case the service exists but the type is not yet known, in case of misconfiguration, or for testing. Please refer to the IDN-Stream standard for the individual service types.

Flags

Status flags and informational flags.

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	0	0	0	0	0	0	0	DSID

Default Service ID (DSID)

This flag marks a default service. Only one default service SHALL be reported per service type. This mark has informational purposes only. This could be for example a service list

that is presented to a user.

Please note that the IDN-Stream standard refines the service type by adding a service mode. A default service (the service in case a service ID of 0 is passed) is located mode-based. It doesn't make much sense though to configure different services for different modes of a single service type.

Relay Number

The number of the relay for topological information.

Relay entry: Must be in range 1..255. Note that since this number identifies a relay instance that forwards to another server, a value of 0 doesn't make sense.

Service entry: A value of 0 identifies a root service. These are services provided by the server itself. A value in range 1..255 identifies the relay with that number in the relay table. A relay number that can't be found in the relay table is invalid.

Name

A user friendly name. The name SHOULD be user configurable.

Relay entry: The host name of the relayed server.

Service entry: The name of the service. This name must not be confused with host names that might include a brand or model number. The service name SHOULD be a universal, changeable name like "left4", "center" or "graphics" such that general setup configuration doesn't have to change in case of swaps.

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). The field is specified unterminated to ensure that reading code doesn't rely on a termination character that may be missing because of broken implementations.

5 Parameters

6 IDN-RT (Realtime Streaming)

Realtime streaming of IDN-Stream channel messages is actually a small protocol on it's own. This sub-protocol could be implemented individually, without the rest of the IDN-Hello protocol. Defining IDN-RT within the IDN-Hello standard makes most sense though because device discovery is usually needed for clients to find a server that a stream could be sent to. Further, IDN-RT shares the packet definition with IDN-Hello.

IDN-RT provides a basic unidirectional method of sending IDN-Stream channel messages, which can be used with broadcasts or multicasts and a complementary method for acknowledgement, which gives feedback about reception and link events. This is especially useful for IDN-Stream messages that route, configure or close channels. Beyond this simple acknowledgement, protocols would go into lossless copies of data like TCP - but with tradeoffs in realtime performance.

Servers SHALL implement full support of this protocol such that clients can choose whether to implement the basic stream (as unicast or broadcast) or the acknowledged stream.

For clarification throughout this chapter, the term link is used for the raw exchange of datagrams while connection is used when the session layer is involved. So, datagrams are part of links between hosts and these datagrams can lead to or can be part of established connections.

IDN-RT doesn't provide session management. Incoming links allocate and connect to sessions with default parameters. Configuration of these parameters is up to the device. The session then processes the IDN-Stream messages that have been received over the established connection. Limited link monitoring is provided by the acknowledge mechanism.

The packet sequence number is used for packet tracking. For the first packet of a connection, the number is arbitrary and stored by the server. For each succeding packet, the number must then be counted up to be able to track for lost packets and link quality. Further, the server SHOULD discard duplicates and could check for the packets being in the correct sequence. However, it should be noted at this point, that realtime streaming requires proper link quality. In case of problems either link quality must be improved or tradeoffs regarding realtime response (and a change of protocol) must be considered.

The strictly monotonically increasing packet sequence number implies that timestamps in IDN-Stream channel messages, transported by the packets, SHALL increase monotonically (they might be equal). Further, with IDN-RT being a real time protocol, timestamps SHALL match the point in time of sending. While producers of discrete streams may just assign the current time, continuous stream producers may have to wait until the next timestamp is reached. In any case, producers may send packets past the timestamps of their channel messages but SHALL not send ahead of these timestamps. Consumers may not be able to synchronize otherwise.

6.1 Processing of IDN-Stream Messages

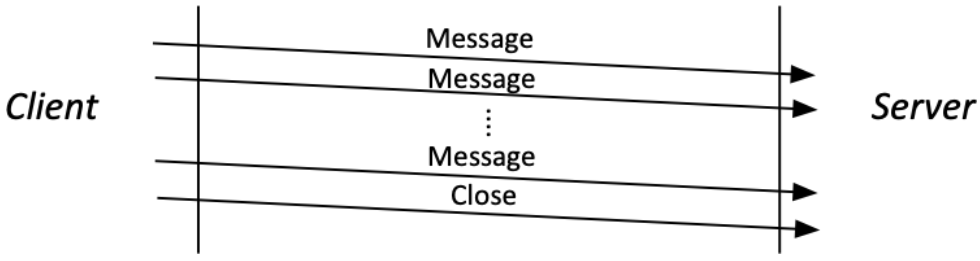
Servers SHALL check incoming packets for their size. This SHALL include a check of the received packet size versus the contained IDN-Stream message length. Malformed packets SHALL be silently discarded and SHOULD therefore not open a connection or reach a session. Empty packets (packets not containing a message) are generally OK. In this case, semantics are specified with the commands.

In case the client sends from a client group, that is currently excluded from processing, packets from that link SHALL not be processed by a session.

Please note that the IDN-RT links and IDN sessions may have individual timeouts for reasons including safety. A session timeout SHALL stop session processing and data output while a link timeout SHALL close the connection and release the session (which would stop processing as well). This means that sending empty packets will keep the link connected and the session allocated but may lose the session output due to a session timeout. A packet containing an IDN-Stream void channel message though is considered part of the stream. The message has a valid timestamp and is processed by the session, which keeps the connection and the session alive.

6.2 Basic Simplex Stream (IDN-BRT)

Simplex, unidirectional streams of IDN channel messages are the most basic way of realtime communication. Just a packet header with a single command is needed. In case the client plays nice or doesn't want to wait for a link timeout, a second command, the connection close can be used. The packets can be transmitted via unicast, multicast or broadcast since no acknowledgement is involved. The sub-standard identifier IDN-BRT (for basic/broadcast realtime streaming) is used for such connections.



In case IDN-RT is implemented on a server as part of IDN-Hello implementation, clients that use the IDN-BRT subset can implement plug and play features by using the ping and scan requests of the IDN-Hello protocol.

6.2.1 Channel Message

Pass an IDN-Stream channel message to an IDN session.

The packet payload (which follows the packet header) can be empty or can contain an IDN-Stream channel message. In case the link is new, the client is not excluded and sessions are available, a server SHALL accept the link and attach a new session to it. For established connections, the link timeout SHALL be reset.

In case the packet payload contains a message, this message SHALL be passed to the IDN session for processing. An empty packet (no packet payload) SHALL just establish a connection or reset the link timeout.

6.2.2 Connection Close

Pass an IDN-Stream channel message to an IDN session and close the connection.

The packet payload (which follows the packet header) can be empty or can contain an IDN-Stream channel message. In case the link is new, the client is not excluded and sessions are available, a server SHALL accept the link and attach a new session to it.

In case the packet payload contains a message, this message SHALL be passed to the IDN session for processing. An empty packet (no payload) is only allowed in case the connection is established.

After processing, the session SHALL be closed gracefully by finishing output or moving to a home position and closing open channels. The connection SHALL be closed. Before closing, note that session status can be used to monitor output lag and resource availablity for new connections.

6.2.3 Abort

Close the connection and reset the attached session immediately.

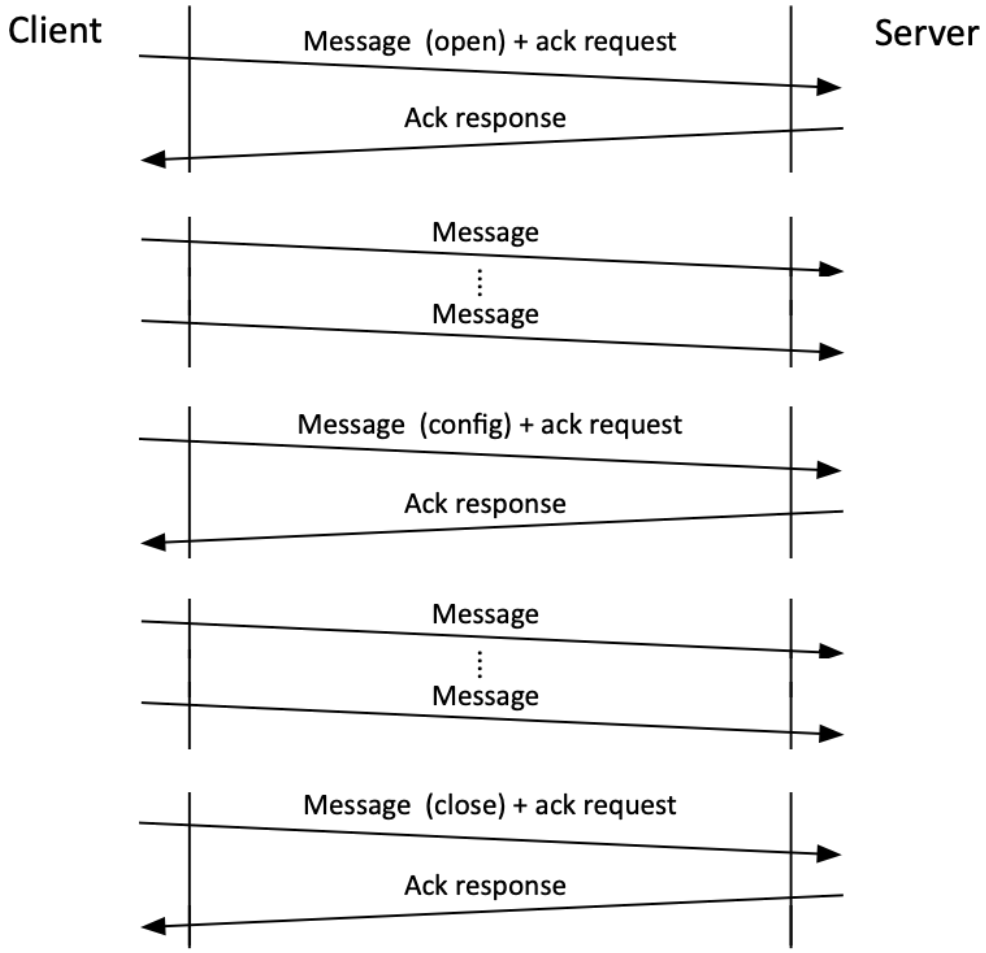
The request has no packet payload.

The session referred to by the link SHALL be reset immediately by aborting output or moving to a home position and resetting open channels. The connection SHALL be closed. Before aborting, session status can be used to monitor output lag.

Please note that IDN-RT is a transport protocol and therefore, this command is not intended to be used for safety purposes.

6.3 Acknowledged Stream

Acknowledgement adds link monitoring and packet reception codes to basic simplex streaming without changing the realtime characteristics. Because of bidirectional traffic, it is suited for unicast communication but could generally be used for multicast/broadcast on transport protocols that allow addressing. In this case, the client would have to keep track of server addresses (for the replies). Please note that for example WIFI links could be degraded by bidirectional traffic.



Acknowledgement makes most sense for the transport of IDN-Stream channel messages that contain a configuration header, especially when this header changes the configuration. With the acknowledgement, the client then knows that the message was received correctly and was passed to the IDN session. Through event flags, the client can find out about what happened on link and session between acknowledgements.

6.3.1 Channel Message with Acknowledge Request

Pass an IDN-Stream channel message to an IDN session, ask for an acknowledgement.

The packet payload (which follows the packet header) can be empty or can contain an IDN-Stream channel message. In case the link is new, the client is not excluded and sessions are available, a server SHALL accept the link and attach a new session to it. For established connections, the link timeout SHALL be reset.

In case the packet payload contains a message, this message SHALL be passed to the IDN session for processing. An empty packet (no payload) SHALL just establish a connection or reset the link timeout.

Finally, an acknowledgement containing the sequence number of the packet that requested the acknowledge SHALL be sent.

6.3.2 Connection Close with Acknowledge Request

Pass an IDN-Stream channel message to an IDN session and close the connection, ask for an acknowledgement.

The packet payload (which follows the packet header) can be empty or can contain an IDN-Stream channel message. In case the link is new, the client is not excluded and sessions are available, a server SHALL accept the link and attach a new session to it.

In case the packet payload contains a message, this message SHALL be passed to the IDN session for processing. An empty packet (no payload) is only allowed in case the connection is established.

After processing, the session SHALL be closed gracefully by finishing output or moving to a home position and closing open channels. The connection SHALL be closed. Before closing, note that session status can be used to monitor output lag and resource availablity for new connections.

Finally, an acknowledgement containing the sequence number of the packet that requested the acknowledge SHALL be sent.

6.3.3 Message Acknowledgement

The reply to an acknowledge request. The reply tells the client about the reception of the packet to be acknowledged, events that happened since the last acknowledgement, and about status of session and channels. The reply could be used for RTT (round trip time) measurements (and with that for cascading latency measurements) and SHALL be sent immediately.

For fields that involve a channel, values SHALL refer to the channel of the message in the packet to be acknowledged. In case there was no message, these fields SHALL read a

reset value or be marked unknown. Fields involving a channel may not reflect the current status of a service or a device connected to that channel. Depending on implementation and use case, session input buffering for jitter compensation and timeline checks may delay channel operations. Therefore, care must be taken, when using advanced techniques like service rerouting or immediate reopening of a channel as their status may lag behind. Producers could monitor the Pipeline Event flags CnlRted and CnlClsd along with sending void messages on the channel in case needed.

For this command, the IDN-Hello packet header is followed by the acknowlegement header:

Octet	0	1	2	3
0	Struct Size	Result Code	Input Event Flags	
4	Pipeline Event Flags		Status Flags	Link Quality
8	Latency			

Struct Size

This field contains the size of the acknowledgement header in octets. The value SHALL be checked upon reception. Later versions may add fields and the size is used to distinguish.

Result Code

Message reception and connection establishment result.

Code	
0x00	Message successfully received and passed to the IDN session
0xEB	Empty (no message) close command without established connection
0xEC	All sessions are occupied by clients (new connection refused)
0xED	The client group is excluded from streaming
0xEE	Invalid packet payload
0xEF	Any other processing error

Input Event Flags

Events that happened on link and session input since the last acknowledge. After the acknowledge is sent, the passed events SHALL be cleared and monitoring SHALL start over. The events SHALL be related to the link, the packet was received on and the session connected to it.

Bit	15 (MSB)	14	13	12	11	10	9	8
	IRAErr	0	CCErr	0	BPErr	LAErr	OVErr	MVErr

Bit	7	6	5	4	3	2	1	0 (LSB)
	SeqErr				Order	0	0	New

Internal Resource or Assertion Error (IRAErr)

If set, this bit indicates an internal problem like memory allocation, structure integrity, data that can't be coalesced, or assertions that were not met. Messages or data got dropped.

Message on Closed Channel (CCErr)

If set, this bit indicates that messages (other than void messages) could not be processed because the referred channel was not routed to a service. Since messages must not be sent on closed channels, this error could indicate a lost channel routing or point to implementation problems. Because of input buffering, the indication could lag behind. IDN-Stream void messages SHALL not be counted as they do not contain data and are usually used for keepalive or monitoring purposes.

Before Process Time (BPErr)

If set, this bit indicates unresolved timestamp discontinuities. Messages that arrived later contained timestamps which are earlier than timestamps that have been processed already. Depending on algorithm or device this may be a push time for lower level processing or an internal tracking mark. Messages got dropped.

Late Arrival Error (LAErr)

If set, this bit indicates that messages arrived later than expected. This may indicate a resolved network congestion that could for example happen on WIFI routers when changing the radio channel. Although the timestamp was found continuous and messages were processed, a lower level device underrun may have occurred.

Overrun Error (OVErr)

If set, this bit indicates that more messages arrived than can be buffered. Since, for real time streaming, more recent messages are more important, older messages have been dropped. While this error can be caused by general resource shortage, other causes could

be excessive jitter which lead to large latency for underrun protection or clients sending faster than realtime beyond server buffer capacity.

Message Validation Error (MVErr)

If set, this bit indicates early message validation and acceptance errors. This could for example be an invalid contentID when checked during session input. Please note that this flag refers to the message itself, opposed to general packet payload checks (size for example), that are reported through the Result Code.

Sequence Error (SeqErr)

In case any of these bits are set, sequence number tracking detected an error since the last time queried. Messages might be lost and output might have been in error. While this could generally happen on packet-switched networks, it should not happen on local networks since throughput of any protocol is degraded. For example, network implementations like network stacks and switches are encouraged to keep datagrams in order because of upper layer protocol throughput. This is not generally true for routing though. In case sequence errors pile up, implementations might be broken or physical link quality might be poor and should be improved.

Bitmask	
0001	Type 1 error: Sequence not strictly monotonic increased (by 1)
0010	Type 2 error: Duplicate sequence number(s)
0100	Type 3 error: Missing sequence number(s)

While all errors relate to the sequence number not strictly monotonically increasing by one, subsequent checking can find the other error types, which may make error tracking easier. In a sequence window, duplicates can be detected immediately while missing sequences can be reported when leaving the window. Since flags are cleared with the acknowledge, the field may show all combinations of types.

Timeline not in Order (Order)

If set, this bit indicates a resolved timestamp discontinuity. Messages contained timestamps which are earlier than timestamps already received, but the implementation was able to reorder the messages silently at the input. In case this bit is set together with sequence errors, packets were reordered by the network. In case there was no sequence error, the producer caused the discontinuity.

New Connection (New)

If set, this bit indicates that a new connection was established since the last time queried. This is a confirmation for a packet that was supposed to establish a connection and would

indicate an anomaly in case the connection is supposed to be established. Connection timeouts from link interruptions could cause such a case.

Pipeline Event Flags

Events that happened on the processing pipeline since the last acknowledge. After the acknowledge is sent, the passed events SHALL be cleared and monitoring SHALL start over. The events SHALL be related to the channel of the message that the acknowledge was requested for and could lag behind. Please refer to the notes at the beginning of the chapter.

Bit	15 (MSB)	14	13	12	11	10	9	8
	IAPErr	DVIErr	PVLErr	RGUErr	MCLErr	CTYErr	DCMErr	CKTErr

Bit	7	6	5	4	3	2	1	0 (LSB)
	CFGErr	FRGErr	BSYErr	SMErr	0	0	CnlClsd	CnlRted

Internal Assertion or Processing Error (IAPErr)

If set, this bit indicates an internal problem like structure integrity, assertions, that were not met, or failures during pipeline processing.

Device Irregularity Error (DVIErr)

If set, this bit indicates a device error. This could be related to data (overrun, underrun), be an unexpected device/API status or be a general malfunction.

Message Payload Validation Error (PVLErr)

If set, this bit indicates that the payload of a message couldn't be processed, decoded, or contained inconsistencies or invalid data. A processing error could be an invalid data sequence, a decoding error could be leftover fractions of decoding units, and an inconsistency could be a mismatch between subsequent data chunks (or an invalid mix of chunk types). For data to run through a decoder, it must usually match integer multiples of decoding units defined with the configuration, used to build the decoder. Fractions of data units cannot generally be moved between data chunks.

Range Check or Unsupported Feature Error (RGUErr)

If set, this bit indicates that a data chunk contained values, that are out of range or asked for features/operations that are not supported. This could be implementation or device limits like frequencies, data rates, or unimplemented optional operations.

Minimum Chunk Length Error (MCLErr)

If set, this bit indicates that data chunks contained less than the required minimum data size or duration. The minimum size depends on service and mode and is defined in the IDN-Stream standard for sanity checks, processing overhead compensation or data framing.

Continuity Error (CTYErr)

If set, this bit indicates that device data processing detected a discontinuity. In most cases, this is related to continuous mode data not being seamless, when a message timestamp plus the duration of the data chunk doesn't match the next message timestamp or a IDN-Stream chunk sequence number plus the chunk payload doesn't match the next chunk sequence number. The error can also point to erroneous integer counter wraps.

Service Data and Configuration Mismatch (DCMErr)

If set, this bit indicates that Service Configuration Match (SCM) bits in the data chunk header did not match Service Data Match (SDM) bits in the configuration header, stored with the decoder. The error may happen for lost configurations.SCM/SDM match is a cross check feature for unreliable connections. Please refer to the IDN-Stream standard.

Chunk Type Error (CKTErr)

If set, this bit indicates that the service connected to the channel received data chunks with invalid types for the service and/or mode. Messages got dropped.

Service Mode Configuration Error (CFGErr)

If set, this bit indicates that the service connected to the channel received a configuration, that is invalid for the service/mode such that a decoder for channel data could not be built. The flag can also indicate that data was to be processed when there is no decoder for the service/mode.

Fragment Reassembly Error (FRGErr)

If set, this bit indicates that reassembling a data chunk from fragments, transmitted in multiple messages, failed. This is very likely because of a lost fragment (first, sequel strictly counting up or last) but could also be because of excess reassembly memory use or be related to implementation problems on both sides.

Service Busy (BSYErr)

If set, this bit indicates that a channel could not be routed because the service (or the service in the requested mode) was busy. This may happen in case two channels try to use the exclusive mode of the same service or in case devices (or implementations) do not allow for sharing in general.

Invalid ServiceID or ServiceMode (SMErr)

If set, this bit indicates that a channel could not be routed to a service because of an unknown or invalid service ID or service mode in the message config header.

Channel Closed (CnlClsd)

If set, this bit indicates that the channel is closed. This indication is static, which means that the flag is set whenever the channel is closed (opposed to being triggered by an event). Because of input buffer delays, the channel status may lag behind. This could lead to a packet containing a message with a channel routing being acknowledged with a closed channel (because the message hasn't been processed yet). IDN-Stream void messages can be used for status monitoring.

Channel Routed (CnlRted)

If set, this bit indicates that the channel was routed to a new service and/or a new service mode. This includes invalid or busy routings where channels get routed to a NULL service to avoid closed channel errors. A confirmation for a failed routing would have the SMErr or BSYErr flag set. A successful routing would have both flags reset.

Status Flags

Current status of link, session and output devices. The flags SHALL only be related to the resources bound by the connection.

Bit	**7 (MSB)**	**6**	**5**	**4**	**3**	**2**	**1**	**0 (LSB)**
	0	0	0	0	0	SNMsg	DOBuf	SOCnl

Session has Messages (SNMsg)

If set, this bit indicates that the session keeps queued messages. This queue is usually needed as a latency buffer for input jitter compensation. Together with the Devices Occupy Buffers flag DOBuf and the Open Channels flag SOCnl, this bit can be used for completion monitoring.

Devices Occupy Buffers (DOBuf)

If set, this bit indicates that devices retained by the session keep data belonging to the stream. This may be needed for FIFO queues that decouple processing from device output.

Pipelining may cause that devices keep data after all channels are closed. When all data is drained, the bit SHALL be reset. This indication SHALL not include data in reassembly queues since this data is not yet processed and fragments might be lost. Together with the Session Message flag SNMsg and the Open Channels flag SOCnl, this bit can be used for completion monitoring. Please note that this indicator may not be infinitely precise. This can be due to complexity or due to data already being vanished in the shallows of hardware.

However, the indicator SHALL mark a point of immutability, where aborting the link would not remove any stream data from the output (because the data is already followed by idle or other data) and the device could take a new channel.

Session has Open Channels (SOCnl)

If set, this bit indicates that the session has open channels. It is valid for a session to have open channels but not keeping messages or output data - or vice versa. For example, in standby situations or at home positions, channels may be open and in case of buffering, a session may not have opened channels yet.

Link Quality

A general quality indicator. The value is implementation dependent and SHOULD be derived from sequence errors, network jitter, congestions and latency. A value of 0 represents an unknown quality. A value of 1 represents a very poor quality (high losses, high jitter) and a value of 255 represents excellent quality. Acknowledge requests for the first couple of packets/messages of a connection SHALL be answered with a value of ‘unknown’ since the quality can’t be known yet. Please note that jitter evaluation needs time stamps and is usually based on those found in IDN Channel Messages contained in the payload of IDN-RT packets.

Latency

The estimated latency for the first octet of the message payload to be visible at its output destination in microseconds. A value of 0 represents an unknown latency. The latency SHALL be related to the channel of the message that the acknowledge was requested for. Please refer to the notes at the beginning of the chapter. Acknowledge requests for the first couple of messages of a newly routed channel can be answered with a value of ‘unknown’ since the latency might not be known yet.

Depending on system setup, network routing, stream copies, transformations, pipelining, and ouput device type, latency calculation can be complex but may be needed by user interfaces for display and operators for tuning or debugging of a setup. Therefore, implementations are encouraged to calculate the latency as precise as possible. Copying streams is a special case and can’t be handled in depth. Depending on implementation or configuration, servers may return the latency of the first path, the maximum latency, or an average of all paths.

Since the acknowledgement packet shall be sent immediately, clients can calculate the round trip time (RTT) by sequence number matching and add half of it to the reported latency to determine the total latency from a sender’s perspective.

7 IDN-Hello over UDP

The User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. With UDP, applications can send datagrams to hosts on an Internet Protocol (IP) network. It uses a simple connectionless communication model with a minimum of protocol mechanisms. It provides checksums for data integrity and port numbers for addressing different functions at the source and destination hosts of the datagram.

UDP has no handshaking and thus exposes applications to network unreliabilities. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

Physical layers may require smaller packets than the 64k UDP limit. The underlaying IP handles fragmentation and reassembly. While this is useful for IDN-Hello and IDN-RT with discrete streams, IP fragmentation SHOULD be avoided for continuous streams because of real time characteristics and larger discontinuities in case single fragments are lost.

7.1 Server Port

IDN-Hello servers SHALL listen on the well known UDP port 7255 (decimal) such that clients can find devices by sending scan broadcasts. This doesn't restrain hosts from running IDN-Hello servers on different ports as this might be needed or useful for custom applications. In order to mark a product "IDN-Hello compliant" though, a server SHALL listen on the well known port.

Servers SHALL respond on the same port, a request was received. Internet protocol (IP) connection identification is based on the 5-tuple: Protocol, source address/port and destination address/port. This scheme is used by the protocol stack (bind, connect, protocol control block), firewalls (temporarily open inbound on outbound traffic) and applications. Multithreaded server implementations SHALL take this into account.

Client applications can choose multiple random ports but must be aware that IDN-RT is connection based and therefore stores the address tuple. A second client port will therefore open a second connection to a second IDN session. For multithreaded applications though, using one client port for administration (ping, scan) and another one for streaming (IDN-RT), absolutely makes sense and is encouraged.

7.2 Link timeout

For safety reasons, by default, the link SHALL time out after 1 second and close the connection. This, in turn, SHALL close the session and stop any output. Links that need larger timeouts are likely to be very unreliable. Link activity can be produced by sending empty packets or IDN-Stream void channel messages (depending on the intention).

8 Revision History

8.1 Revision xxx, XXXX 20xx

The initial publication

Contributors (Revision 001)

Matthias Frank, University of Bonn
Theo Dari
Patrick Murphy
Dirk Apitz, DexLogic

