M. Spencer
Digium, Inc.
B. Capouch
Saint Joseph's College
E. Guy, Ed.
TruPhone
F. Miller
Cornfed Systems, Inc.
K. Shumard

Internet-Draft
Intended status: Informational
Expires: October 1, 2008
March 30, 2008

IAX: Inter-Asterisk eXchange Version 2
draft-guy-iax-04

Status of this Memo

Abstract

This document describes IAX, the Inter-Asterisk eXchange protocol, an application-layer control and media protocol for creating, modifying, and terminating multimedia sessions over Internet Protocol (IP) networks. IAX was developed by the open source community for the Asterisk PBX and is targeted primarily at Voice over Internet Protocol (VoIP) call control, but it can be used with streaming video or any other type of multimedia.

IAX is an "all in one" protocol for handling multimedia in IP networks. It combines both control and media services in the same protocol. In addition, IAX uses a single UDP data stream on a static port greatly simplifying Network Address Translation (NAT) gateway traversal, eliminating the need for other protocols to work around NAT, and simplifying network and firewall management. IAX employs a compact encoding which decreases bandwidth usage and is well suited for Internet telephony service. In addition, its open nature permits new payload types additions needed to support additional services.

Table of Contents

1. Introduction

Numerous protocols have been specified by the Internet community to support control or signaling of multimedia sessions, for instance, SIP [RFC3261], MGCP [RFC3435], and MEGACO/H.248 [RFC3525]. In general, these protocols are designed to offer full support for many types of media transmission. This flexible approach adds some overhead to the protocol headers, but allows for the protocol use well beyond the current application. Typically, these protocols reference, but do not specify, the media transmission protocol used to carry the actual stream. SIP commonly uses Session Description Protocol (SDP) [RFC4566] to specify Real-Time Transport Protocol (RTP) [RFC3550] streams. This method allows for great flexibility, but again leads to more overhead. Furthermore, multimedia solutions which use different, perhaps dynamic, network addresses for signaling and media transmission frequently suffer from Network Address Translation (NAT) traversal and security challenges.

IAX is the Inter-Asterisk eXchange protocol which facilitates VoIP connections between servers, and between servers and clients that also use the IAX protocol. IAX was created through an open source methodology rather than through a traditional standards based methodology. It is an open protocol originally used by Asterisk, a dual licensed open source and commercial PBX server from Digium. Independent IAX implementations may be open, proprietary, or licensed in anyway the author seems fit without royalty to the protocol creators.

1.1. Basic Properties

IAX is a robust and full featured, yet, simple protocol. It is general enough that it can handle most common types of media streams. However, the protocol is highly optimized for VoIP calls where low overhead and low bandwidth consumption are priorities. This pragmatic aspect makes IAX more efficient for VoIP than protocols which consider possibilities far beyond current needs and specify many more details than are strictly necessary to describe or transport a point-to-point call. Furthermore, because IAX is designed to be lightweight and VoIP-friendly, it consumes less bandwidth than more general approaches. IAX is a binary protocol, designed to reduce overhead, especially in regards to voice streams. Bandwidth efficiency, in some places, is sacrificed in exchange for bandwidth efficiency for individual voice calls. For example, when transmitting a voice stream compressed to 8kbs with a 20ms packetization, each data packet consists of 20 bytes. IAX adds 20% overhead, 4 bytes, on the majority of voice packets while RTP adds 60% overhead with 12 additional bytes per voice packet.

In addition to efficiency, IAX's single static UDP port approach makes IAX traffic easy for network managers to shape, prioritize, and pass through firewalls. IAX's basic structure is that it multiplexes signaling and multiple media streams over a single UDP stream between two computers. IAX also uses the same UDP port for both its signaling and media messages, and because all communications regarding a call are done over a the same point-to-point path, NAT traversal is much simpler for IAX than for other commonly deployed protocols.

1.2. Drawbacks

While IAX is very effective, addressing many of today's communications needs, it does have a few limitations. For instance, IAX uses a point-to-point codec negotiation mechanism that limits extensibility because every IAX node in a call path must support every used codec to some degree. In addition, the codec definition is controlled by an internally defined 32-bit mask, so the codecs must be defined in the protocol, and the maximum number of simultaneous codecs is, therefore, limited.

One of IAX's design strengths also presents a potential problem. The use of a single, well-known, port makes the protocol an easier target for denial of service attacks. Real time systems like VoIP are particularly sensitive to these attacks.

The protocol is typically deployed with all signaling and media going to a centralized server. While this combined path approach provides a great deal of control, it limits the overall system scalability. IAX now provides the ability to split the media from the signaling stream which overcomes this limitation of earlier IAX versions.

Most IAX drawbacks are due to implementation issues rather than protocol issues. Threading presents a series of problems. Many implementations have a limited number of threads available to process IAX traffic and can become overwhelmed by high use or denial of service attacks. Newer implementations have additional controls to minimize the impact of these challenges.

2. IAX Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Additionally, this document uses the following terminology:

Peer: A host or device that implements the IAX protocol.

Call: A call is a relationship between two or more parties (i.e., resources such as devices, user agents, or programs) that exists for some time for the purpose of exchanging real-time media. In the context of this document, a call is an end to end relationship where at least the one leg of call path is implemented using the IAX protocol.

Calling Party: A device or program that initiates a call.

Called Party: A device or program to which a call is directed.

Context: A context is a named partition of a Dialplan.

Dialplan: A Dialplan is a set of rules for associating provided names and numbers with a particular called party.

Frame: The atomic communication unit between two IAX peers. All IAX messages are carried within frames.

Information Element (IE): A discrete data unit appended to an IAX frame which specifies user or call-specific data.

Registrant: A registrant is a peer that makes REGISTER requests in order to advertise the address of a resource, i.e., a device or program to which a call may be directed.

Registrar: A registrar is a peer that processes REGISTER requests and places the information it receives in those requests into the location service. [RFC3261].

3. Overview of IAX Protocol

IAX is a peer-to-peer, VoIP-oriented, protocol. IAX includes both control and media functions. It can register locations, create, modify, terminate multimedia sessions, and carry the actual media streams specified by the sessions it manages. The protocol is designed and optimized for describing and transporting multimedia calls using Internet Protocol. This document describes version 2 of IAX; Version 1, although somewhat similar in design, utilized a different port and was not widely deployed.

The basic design approach for IAX multiplexes signaling and multiple media streams over a single UDP association between two hosts. This is accomplished by using the same "well-known" UDP port, 4569, for all types of IAX traffic. IAX's unified signaling and media paths achieve NAT transparency, which is an advantage of IAX over alternative media transport protocols such as SIP [RFC3261].

IAX is coded as a binary protocol. One major benefit of using a binary protocol is bandwidth efficiency because the quality of voice calls is frequently related to the amount of bandwidth consumed. This is one way the protocol is specifically optimized to make efficient use of bandwidth for individual voice calls. The bandwidth efficiency for other stream types is sacrificed for the sake of individual voice calls. Other benefits of a binary protocol are robustness against buffer overrun attacks, and compact implementation capability, which reduces interoperability issues related to parsing.

The atomic communication unit in IAX is the "Frame". There are multiple classes of Frames, each of which is described below. In general, "Full Frames" carry signaling/control data, while "Mini Frames" carry media stream data. Full Frames enclose optional 'Information Elements' (IEs). IEs describe various types of user- or call-specific data. "Meta Frames" are used for call trunking or video stream transmission.

An IAX-based call may consist of many call legs, or segments. Each call leg may be implemented using different protocols, e.g., SIP to IAX to ISDN (Integrated Services Digital Network). IAX is responsible for setting up one or more legs of a complete call path, not necessarily the end-to-end call.

IAX is an optimized peer-to-peer protocol. If two adjacent call legs utilize the IAX protocol and if the intermediate peer determines that it does not need to remain in the call path, it can supervise a calling path change such that it removes itself from the path. This supervision is complete, a call path is not changed until all peers in the optimized call path confirm they can properly communicate.

IAX supports security features by allowing multiple methods of user authentication and authorization, as well as during peer registration. IAX also specifies a generic framework for native encryption.

4. Naming Conventions

Call Identifier: A call leg is marked with two unique integers, one assigned by each peer involved in creating the call leg.

Number: The Calling and Called Numbers are a set of digits and letters identifying a call originator and the desired terminating resource. The term 'Number' is historic and has been expanded to include letters. A peer is responsible for defining its own dialplan. A peer MAY define its dialplan according to ITU-T Recommendation E.164. [E164] However, this is not required.

Username: A username is a string used for identification purposes.

5. IAX Uniform Resource Identifiers

5.1. IAX URI Scheme Registration

This section registers IAX according to the guidelines in [RFC4395].

URI scheme name:

iax.

Status:

Permanent.

URI scheme syntax:

The "iax:" scheme follows the guidelines in [RFC3986].

The general form is as follows:

```
iax:[username@]host[:port][/number[?context]]
```

where these tokens have the following meanings:

iax: The literal 'iax:'.

username: A string used for identification purposes.

host: The domain of the resource. The host part contains either a fully-qualified domain name or numeric IPv4 or IPv6 address. An IPv6 address must be enclosed within brackets (i.e., '[2001:db8::1]') as defined in [RFC3986]. Using the fully-qualified domain name form is RECOMMENDED whenever possible.

port: The numeric UDP port number.

number: The name or number identifying the resource on that host.

context: The name of the host partition in which the service is identified or processed.

Examples

```
iax:example.com/alice
iax:example.com:4569/alice
iax:example.com:4570/alice?friends
iax:192.0.2.4:4569/alice?friends
iax:[2001:db8::1]:4569/alice?friends
iax:example.com/12022561414
iax:johnQ@example.com/12022561414
```

ABNF Certain values are included by reference from [RFC3986]:

```
iax-uri = "iax:" [userinfo "@" ]host[":" port][ "/" number[ "?"
context]]

userinfo = <as specified in RFC 3986>

host = <as specified in RFC 3986>

port = <as specified in RFC 3986>

number = *(unreserved | sub-delims | pct-encoded )

context = *(unreserved | sub-delims | pct-encoded )

unreserved = <as specified in RFC 3986>

sub-delims = <as specified in RFC 3986>

pct-encoded = <as specified in RFC 3986>
```

URI Scheme Semantics:

An IAX URI identifies a communications resource capable of communicating using the IAX Version 2 protocol defined in this document. Within this document, we refer to IAX Version 2 protocol URI as IAX. An IAX URI contains enough information to initiate an IAX-based call with that resource.

IAX URIs are associated with server resources to which calls may be routed. For instance, an IAX URI may represent an appearance on a phone, a voice-mail box on a messaging service, an interactive program, a PSTN address or gateway, or any group of the above.

The iax uri scheme translates into a location that may be used by the iax protocol to establish a new call using the uri scheme components described in the previous section. This new call function is the only defined operation.

Encoding considerations:

IAX URI scheme encoding conforms to the encoding rules established for URIs in [RFC3986].

Applications/protocols that use this URI scheme name:

The scheme is used by ENUM Dynamic Delegation Discovery System (DDDS) services to specify resources that support the IAX protocol. The IAX protocol provides application-layer control and media protocol for creating, modifying, and terminating multimedia sessions over Internet Protocol (IP) networks.

Interoperability considerations:

None.

Security considerations:

The IAX URI Scheme does not introduce any new security concerns except that it provides a uniform syntax for describing IAX resources and that, when published, these addresses are subject to various denial of service attacks.

Contact:

Ed Guy, edguy@emcsw.com, +1.973.437.4519.

Author/Change controller

Not Applicable.

References:

Spencer, M., Shumard, K., Capouch, B., and E. Guy, 'IAX: Inter-Asterisk eXchange Version 2,' (This Document), March 2008.

5.2. URI Comparison

Some operations in this specification require determining whether two IAX URIs are equivalent. IAX URIs are compared for equality according to the following rules:

All components of the URI MUST be identical except:

The port, if omitted, is considered to be the same as the default, 4569.

All URI components, except the username field, are case insensitive, and MUST be normalized to lower case as per Section 6.2.2.1 of [RFC3986] before comparison.

The URIs within each of the following sets are equivalent:

```
iax:atlanta.com/alice
iax:AtLaNtA.com/ALicE
iax:atlanta.com:4569/alice

iax:alice@atlanta.com/alice
iax:alice@AtLaNtA.com:4569/ALicE
```

The URIs within the following set are not equivalent:

```
iax:ALICE@atlanta.com/alice
iax:alice@atlanta.com/alice
```

NOTE: A host in domain form and in IP address form are NOT considered identical even if the host name resolves to an address record that matches the given IP address.

6. Peer Behavior and Related Messages

Messages are divided into two categories: reliable and non-guaranteed. The reliable messages are referred to as "Full Frames." In addition to a message type indicator and facilities to ensure reliability, see Section 7, they include the full call identifier. It consists of each of peer's identifiers for the call. Additional attributes, "Information Elements" or "IEs", may be associated with the Full Frame messages.

The non-guaranteed messages are referred to as "Mini-Frames" and "Meta Frames" and these more compact messages only have the originating peer's call identifier and MUST NOT have any "Information Elements."

Peer behavior is presented in several partitions divided by the following functional areas:

Registration (OPTIONAL)

Call Link Management

Call Path Optimization (OPTIONAL)

Mid-Call Behavior

Call Tear Down

Network Monitoring

Digit Dialing (OPTIONAL)

Miscellaneous

Media Messages

Each of these behavior topics and the messages involved are described in the sections which follow.

6.1. Registration (OPTIONAL)

6.1.1. Overview

In order for one IAX peer to be reachable by another IAX peer, the calling peer needs the network address of the receiving peer. This address may be manually provisioned, determined through a shared directory, e.g. an ENUM-like service, [RFC3761] or configured using the IAX protocol. IAX provides a facility for one peer to register

its address and credentials with another so that callers can reach the registrant. The IAX registration facility is optional. If implemented, the IAX registration protocol MAY be done in parts, e.g., an analog telephone adapter MAY only implement the registrant portion of the protocol.

IAX allows user authentication via multiple methods. MD5 Message-Digest authentication [RFC1321] uses a md5 sum arrangement, but still requires that both ends have plaintext access to the secret. Rivest, Shamir and Adleman's (RSA) algorithm [RFC3447] allows unidirectional secret knowledge through public/private key pairs. IAX Private keys SHOULD always be Triple Data Encryption Standard (3DES) encrypted [RFC1851].

```
                         ________________
                        |                |
                        |  Unregistered  |<--------------------------\
                        |________________|                           |
                                |                                    |
                  /Init         |                                    |
                  ------------  |                                    |
                  snd REGAUTH   |    +--------+                      |
                                |    |        | rec REGAUTH          |
                         _______V____V___     | -----------          |
                        |                |    | snd REGREQ           |
                        |   Reg Sent     +----+                      |
                        |________________+----------+                |
                                |    ^              | rec REGAUTH    |
                   rec REGACK   |    |              | /No Credentials|
                  ------------  |    | REG timeout  | -------------- |
                   snd ack      |    | -------      | snd ack        |
                                |    | REGREQ     __V___             |
                         _______V____|___        |      |            |
                        |                |       |  No  |            |
                        |   Registered   |       | Auth |            |
                        |________________|       |______|            |
                                |                   ^                |
                                |                   | rec REGAUTH    |
                                | release           | /No Credentials|
                                | -------           | -------------- |
                  +-------+     | snd REGREL        | snd ack        |
     rec REGAUTH  |       |     |                   |                |
     -----------  |      _V_____V________           |                |
     snd REGREL   |     |                |----------+                |
                  +-----+   Releasing    |---------------------------+
                        |________________|      rec ACK
                                                -------
                                                   x


                     __________
    rec  REGREJ     |          |
    ----------   *->| Rejected |
    snd   ack       |__________|
```

Figure 1: Registrant State Diagram

Registration, illustrated in Figure 1, is performed by a registrant that sends a username and a registration 'refresh' period to the registrar. This is accomplished with a REGREQ message. If

authentication is required, the registrar responds with the REGAUTH message which indicates the types of authentication supported by the registrar. In response, the registrant resends a REGREQ with one of the supported authentications. If the registrant can not authenticate, no further action is necessary. If accepted, the registrar sends a REGACK message which MUST indicate the 'apparent address' and SHOULD indicate the 'refresh'/expire time. If no 'refresh' is sent a default registration expiration of 60 seconds MUST be assumed by both peers. At any time during this exchange, the registrar may send a REGREJ message to indicate a failure.

A registration has a specified time period associated with it for which it is valid. This time period begins when the registrar sends a REGACK message. A registrant may extend that time period be repeating the registration process. A registrant MAY also force an expiration in the registrar by sending the REGREL message. This message may be challenged with REGAUTH or if sufficient credentials were included, it will be accepted with REGACK. In response to a REGAUTH, a REGREL message SHOULD be resent using the specified credentials.

See Section 9.3 and Section 9.4 for example call flows.

6.1.2. REGREQ Registration Request Message

The REGREQ occurs independently of any media-carrying call. A REGREQ MUST include the 'username' IE and SHOULD include the 'refresh' IE. A REGREQ is used both for an initial registration request as well as for a reply to a REGAUTH. As a reply to a REGAUTH message, it MUST include credentials such as a response to a REGAUTH's challenge.

Upon receipt of a REGREQ message which has credentials, a registrar MUST determine their validity. If valid, it MUST respond with a REGACK message indicating the time period for which this registration is valid. If the provided credentials are not valid or the registrar cannot validate the credentials, the registrar MUST respond with a REGREJ message. If credentials are not provided, the registrar MUST respond with a REGAUTH message that indicates the available authentication methods.

Registrants MUST implement this message and registrars MUST be able to process it.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Username	Section 8.4.6	Required	
MD5 Result	Section 8.4.15	Conditional	per REGAUTH
RSA Result	Section 8.4.16	Conditional	per REGAUTH
Refresh	Section 8.4.18	Optional	

6.1.3. REGAUTH Registration Authentication Response Message

A REGAUTH is a response to a REGREQ or REGREL. It is sent when a registrar requires authentication to permit registration. A REGAUTH message MUST include the 'authentication methods' and 'username' IEs, and the 'MD5 challenge' or 'RSA challenge' IE if the authentication methods include MD5 or RSA.

Upon receipt of a REGAUTH message, the registrant MUST resend the REGREQ or REGREL message with one of the requested credentials, if it has the specified credentials.

Registrars MUST implement this message and registrants MUST be able to process it.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Username	Section 8.4.6	Required	
Auth Methods	Section 8.4.13	Required	
Challenge	Section 8.4.14	Conditional	If RSA or MD5

6.1.4. REGACK Registration Acknowledgment Message

A REGACK is sent in response to a REGREQ. A REGACK typically includes the 'refresh' IE specifying the number of seconds before the registration will expire. If the 'refresh' IE is not included with a REGACK, a default registration expiration of 60 seconds MUST be assumed. A REGACK MAY also include the 'username' and 'apparent

address' IEs to indicate how the peer identifies the registrant. IEs related to caller identification or the time the registration occurred MAY be sent as well.

Receipt of a REGACK message requires an ACK in response.

Registrars MUST be able to send this message and registrants MUST be able to process it.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Username	Section 8.4.6	Required	
Date Time	Section 8.4.28	Required	
Apparent Address	Section 8.4.17	Required	
Message Count	Section 8.4.23	Optional	
Calling Number	Section 8.4.2	Optional	
Calling Name	Section 8.4.4	Optional	
Refresh	Section 8.4.18	Optional	

6.1.5. REGREJ Registration Rejection Message

A REGREJ indicates that a registration request has been rejected. This rejection can occur for several reasons. A REGREJ MUST include the 'causecode' and 'cause' IEs to specify why registration was rejected.

Upon receipt of a REGREJ message, the registrant MUST consider registration process unsuccessful and no further interaction is required. A peer MAY reinitiate the process at later time accounting for potential configuration changes on the registrar or registrant.

Both registrants and registrars MUST be capable of sending and processing this message.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Cause	Section 8.4.21	Required	
Cause Code	Section 8.4.33	Required	

6.1.6. REGREL Registration Release Request Message

A REGREL is used by a registrant for a forced release of a prior registration. It MUST include the 'username' IE to identify the registrant to be released, and MAY include the 'causecode' and 'cause' IEs to specify why registration is being released.

Upon receipt of this message, a peer MUST authenticate the sender using the provided credentials or send a REGAUTH message requesting them. If authenticated it MUST immediately purge its registration of the specified registrant or send a REGREJ message if the registration is not found.

Registrants SHOULD be capable of sending this message and registrars MUST be able to process it.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Username	Section 8.4.6	Required	
MD5 Result	Section 8.4.15	Conditional	MD5 or RSA Result is required
RSA Result	Section 8.4.16	Conditional	
Cause	Section 8.4.21	Optional	
Cause Code	Section 8.4.33	Optional	

6.2. Call Leg Management

```
                                          +--------+  HANGUP/ack
                                          |        |
                             _____________|__      |
                            |                |     |
                 +--------->|    Initial     |<----+
                 |          |________________|<---------------------+
                 |                  |                               ^
                 |       start call |                               |
                 |       ---------- |                               |
                 |       send NEW   |  +-------+                    |
                 |                  |  |       |  rec AUTHREQ       |
                 |             _____V__V__     |  -----------       |
                 |            |           |    |  snd AUTHREP       |
                 +------------|  Waiting  |----+                    |
         rec REJECT           |___________|------------------------>+
         ----------                  |                              |
           ack                       |              rec HANGUP      |
                                     |              ---------       |
                                     |              snd ack         |
                                     |                              |
                       rec ACCEPT    |                              |
                       ----------    |   +------+                   |
                       snd ack       |   |      | PROCEEDING / ack  |
                            _________V___V      | RINGING / ack     |
                           |              |     |                   |
                           |     Linked   |-----+                   |
                           |______________|------------------------>+
                                    |               rec HANGUP      |
                       rec ANSWER   |               ----------      |
                      ------------  |               snd ack         |
                       snd ack      |                               |
                                    |                               |
                                    |               rec HANGUP      |
                             _______V________       ---------       |
                            |                |       snd ack        |
                            |      UP        |--------------------->+
                            |________________|--------------------->+
                                                    finish
                                                    ------
                                                    snd HANGUP
```

                 Figure 2: Call Origination State Diagram

```
                                 +--------+ rec HANGUP/ack
                                 |        |
                    _____________V__      | rec NEW(no Auth)/snd AUTHREQ
                   |                |     |
                   |    Initial     |-----+ rec NEW(not Auth)/snd REJECT
                   |                |
                   |________________|<--------------------+
                           |                              |
             rec NEW       |                              |
        (valid credentials)|                              |
             ----------    |   +------+                   |
             snd ACCEPT    |   |      | snd PROCEEDING    |
                  _________V___V      | snd RINGING       |
                 |              |     |                   |
                 |     Linked   |-----+                   |
                 |              |
                 |______________|------------------------>+
                          |               rec HANGUP      |
              /answered   |               ----------      |
             -----------  |               snd ack         |
             snd ANSWER   |                               |
                          |               rec HANGUP      |
                   _______V________       ---------       |
                  |                |       snd ack        |
                  |      UP        |--------------------->+
                  |________________|--------------------->+
                                          finish
                                          ------
                                          snd HANGUP
```

Figure 3: Call Termination State Diagram

6.2.1. Overview

The IAX protocol can be used to setup 'links' or 'call legs' between two peers for the purposes of placing a call. The process, illustrated in Figure 2 and Figure 3, starts when a peer sends a NEW message indicating the destination 'number' (or name) of a Called Party on the remote peer. The remote peer can respond with either a credentials challenge (AUTHREQ), a REJECT message, or an ACCEPT message. The AUTHREQ message indicates the permitted authentication schemes and SHOULD result in the sending of an AUTHREP message with the requested credentials. The REJECT message indicates the call cannot be established at this time. And ACCEPT indicates that the call leg between these two peers is established and that Higher level call signaling (Section 6.3) MAY proceed. After sending or receiving the ACCEPT message, the Call Leg is in the 'Linked' state and is used

to pass call control message until the call is completed. Further detail on messages used for this process can be found in Section 6.3.

Call Legs are labeled with a pair of identifiers. Each end of the call leg assigns the source or destination identifier during the call leg creation process.

6.2.2. NEW Request Message

A NEW message is sent to initiate a call. It is the first call-specific message sent to initiate an actual media exchange between two peers. 'NEW' messages are unique compared to other Call Supervision messages in that they do not require a destination call identifier in their header. This absence is because the remote peer's source call identifier is not created until after receipt of this frame. Before sending a NEW message, the local IAX peer MUST assign a source call identifier that is not currently being used for another call. A time-stamp MUST also be assigned for the call, beginning at zero and incrementing by one each millisecond. Sequence numbers for a NEW message, described in the transport section, (Section 7) are both set to 0.

A NEW message MUST include the 'version' IE, and it MUST be the first IE; the order of other IEs is unspecified. A NEW SHOULD generally include IEs to indicate routing on the remote peer, e.g., via the 'called number' IE or to indicate a peer partition or ruleset, the 'called context' IE. Caller identification and CODEC negotiation IEs MAY also be included.

Upon receipt of a NEW message, the receiving peer examines the destination and MUST perform one of the following actions:

Send a REJECT response,

Challenge the caller with an AUTHREQ response,

Accept the call using an ACCEPT message, or

Abort the connection using a HANGUP message, although the REJECT message is preferred at this point in call.

If the call is accepted, the peer MUST progress the call and further respond with one of PROCEEDING, RINGING, BUSY or ANSWER depending on the status of the called party on the peer. See Section 6.3 for further detail.

The following table specifies IEs for the NEW message:

IE	Section	Status	Comments
Version	Section 8.4.10	Required	
Called Number	Section 8.4.1	Required	
Auto Answer	Section 8.4.24	Optional	
Codecs Prefs	Section 8.4.35	Required	
Calling Presentation	Section 8.4.29	Required	
Calling Number	Section 8.4.2	Optional	
Calling TON	Section 8.4.30	Required	
Calling TNS	Section 8.4.31	Required	
Calling Name	Section 8.4.4	Optional	
ANI	Section 8.4.3	Optional	
Language	Section 8.4.9	Optional	
DNID	Section 8.4.12	Optional	
Called Context	Section 8.4.5	Conditional	'Default' assumed if IE excluded
Username	Section 8.4.6	Optional	
RSA Result	Section 8.4.16	Conditional	If challenged with RSA.
MD5 Result	Section 8.4.15	Conditional	If challenged with MD5
Format	Section 8.4.8	Required	
Capability	Section 8.4.7	Conditional	
ADSICPE	Section 8.4.11	Optional	

Date Time	Section 8.4.28	Optional	Suggested
Encryption	Section 8.4.34	Optional	

6.2.3. ACCEPT Response Message

An ACCEPT response is issued when a NEW message is received, and authentication has taken place (if required). It acknowledges receipt of a NEW message and indicates that the call leg has been setup on the terminating side, including assigning a CODEC. An ACCEPT message MUST include the 'format' IE to indicate its desired CODEC to the originating peer. The CODEC format MUST be one of the formats sent in the associated NEW command.

Upon receipt of an ACCEPT, an ACK MUST be sent and the CODEC for the call MAY be configured using the 'format' IE from the received ACCEPT. The call then waits for an ANSWER, HANGUP or other call control signal. (See Section 6.3.) If a subsequent ACCEPT message is received for a call which has already started, or has not sent a NEW message, the message MUST be ignored.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Format	Section 8.4.8	Required	

6.2.4. REJECT Response Message

A REJECT response is sent to indicate that a NEW, AUTHREP, DIAL, or ACCEPT request has been denied. It MAY be due to an authentication failure, an invalid username, or if a peer cannot provide a valid password or response to an issued challenge. It MAY also be used to notify a peer of a call setup failure, e.g., when IAX peers cannot negotiate a CODEC to use. Upon receipt of a REJECT message, the call leg is destroyed and no further action is required. (Note: REJECT messages require an explicit ACK.)

REJECT messages MAY include the 'causecode' and 'cause' IEs to indicate the rejection reason.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Cause	Section 8.4.21	Optional	
Cause Code	Section 8.4.33	Optional	

6.2.5. HANGUP Request Message

A HANGUP message is sent by either peer and indicates a call tear-down. It MAY include the 'causecode' and 'cause' IEs to indicate the reason for terminating the call. Upon receipt of a HANGUP message, an IAX peer MUST immediately respond with an ACK, and then destroy the call leg at its end. After a HANGUP message has been received for a call leg, any messages received which reference that call leg (i.e., have the same source/destination call identifiers) MUST be answered with an INVAL message. This indicates that the received message is invalid because the call no longer exists.

After sending a HANGUP message, the sender MUST destroy the call and respond to subsequent messages regarding this call with an INVAL message.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Cause	Section 8.4.21	Optional	
Cause Code	Section 8.4.33	Optional	

6.2.6. AUTHREP Authentication Reply Message

An AUTHREP MUST include the appropriate challenge response or password IE, and is only sent in response to an AUTHREQ. An AUTHREP requires a response of either an ACCEPT or a REJECT.

Typical reasons for rejecting an AUTHREP include 'destination does not exist' and 'suitable bearer not found'.

The following table specifies IEs for this message:

IE	Section	Status	Comments
RSA Result	Section 8.4.16	Conditional	If RSA
MD5 Result	Section 8.4.15	Conditional	If MD5

6.2.7. AUTHREQ Authentication Request Message

The AUTHREQ message is sent in response to a NEW message if authentication is required for the call to be accepted. It MUST include the 'authentication methods' and 'username' IEs, and the 'challenge' IE if MD5 or RSA authentication is specified.

Upon receiving an AUTHREQ message, the receiver MUST respond with an AUTHREP or HANGUP message.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Username	Section 8.4.6	Required	
Auth Methods	Section 8.4.13	Required	
Challenge	Section 8.4.14	Required	

6.3. Call Control

6.3.1. Overview

IAX's call control messages provide end-to-end signaling functions common to other telephony control protocols. The messages include RINGING, ANSWER, BUSY, and PROCEEDING. These messages MUST only be sent after an IAX call leg has been ACCEPTed.

In response to an exchange starting with a NEW message, typically, the first call control message is RINGING, however, a PROCEEDING message MAY precede it or the call MAY proceed directly to the ANSWER message. If the call is answered, an ANSWER message will be sent. Other possibilities include a "BUSY" indication, or if the called party's service cannot be reached, the call will be torn down using the link-level HANGUP and an appropriate cause code.

If the link was started with a DIAL message, the sequence is an optional PROCEEDING, then optional RINGING, then ANSWER or BUSY. Of course, a link level HANGUP MAY occur at any time.

Various private extensions to IAX Control messages have been deployed for passing application-specific data over IAX control link. One such extension is an application that controls ham radio transceivers. An IAX peer that receives a control message that is not understood MUST respond with the UNSUPPORT message.

The mandatory IAX control messages are explained below.

6.3.2. PROCEEDING Response Message

The PROCEEDING message SHOULD be sent to a calling party when their call request is being processed by a further network element but has not yet reached the called party.

Upon receipt of a PROCEEDING message, the peer SHOULD perform protocol-specific actions to indicate this fact to the calling party, e.g., tones, an ISUP Proceeding message, etc. If the prior call leg is utilizing the IAX protocol, a PROCEEDING message MUST be sent to that peer. The processing of this message at an originating or transcoding peer is not specified, however, if possible, the status may be displayed to the calling party.

The PROCEEDING message does not require any IEs.

6.3.3. RINGING Response Message

This message is sent from a terminating party to indicate that that the called party's service has processed the call request and is being alerted to the call. A IAX RINGING message MUST be sent to an IAX-based calling party when the peer determines that the called party is being alerted, e.g., when their phone is ringing.

Upon receipt of an IAX RINGING message, the peer MUST pass this indication to the calling party, unless the calling party has already received such indication. For an initiating peer, this is typically done by starting the ring-back tone, however, many implementations start ringback before ringing in order to meet user expectations. If the calling party is using the IAX protocol, a RINGING message MUST be passed to this caller.

The RINGING message does not require any IEs.

6.3.4. ANSWER Response Message

This message is sent from the called party to indicate that the party has accepted the call request and is communicating with the calling party. Upon receipt of this message, any ring-back or other progress tones MUST be terminated and the communications channel MUST be opened.

The ANSWER message does not require any IEs.

6.4. Mid-Call Link Operations

6.4.1. FLASH Request Message

The FLASH message is sent to indicate a mid call feature. Its interpretation is system dependent and if it is not expected, it SHOULD be ignored. Typically, this message is only sent from Analog Telephone adapters when a brief circuit interruption is made during an answered call.

The FLASH message does not require any IEs.

6.4.2. HOLD Request Message

The HOLD message is sent to cause the remote system to stop transmitting audio on this channel, and optionally replace the audio with music or other sounds. If the remote system cannot perform this request, it SHOULD be ignored.

The HOLD message SHOULD only be sent in IAX calls which are started using the DIAL message.

The HOLD message does not require any IEs.

6.4.3. UNHOLD Request Message

The UNHOLD message is sent to cause the remote system to resume transmitting audio on this channel. If the remote system cannot perform this request, it SHOULD be ignored.

The UNHOLD message SHOULD only be sent in IAX calls after the HOLD message.

The UNHOLD message does not require any IEs.

6.4.4. QUELCH Request Message

The QUELCH message is sent to cause the remote peer to squelch or stop transmitting audio on this channel. It MAY replace the audio sent to the further party with music or other sounds. If the remote system cannot perform this request, it SHOULD be ignored.

The QUELCH message MUST only be sent in IAX calls after an ACCEPT is sent or received; it SHOULD only be used on calls which are started using the NEW message.

The QUELCH message does not require any IEs.

6.4.5. UNQUELCH Request Message

The UNQUELCH message is sent to cause the remote system to resume transmitting audio on this channel. If it previously replaced the audio with music or other sounds, it MUST discontinue it immediately. If the remote system cannot perform this request, it SHOULD be ignored.

The UNQUELCH message SHOULD only be sent in IAX calls after the QUELCH message.

The UNQUELCH message does not require any IEs.

6.4.6. TRANSFER Request Message

The TRANSFER message causes the receiving peer to restart the call using another specified number. The receiving peer MUST be on the calling side of this call leg and the new call behavior is unspecified. After processing this message, a HANGUP message SHOULD be sent and the call leg torn down.

When sending a TRANSFER message, the new number to which the call is being transferred MUST be included in the CALLED_NUMBER IE and a CALLED_CONTEXT IE MAY be included. The call leg MUST NOT be used for anything else and MAY be torn down.

The following table specifies IEs for this message:

IE	Section	Status	Comments
Called Number	Section 8.4.1	Required	

Called Context	Section 8.4.5	Optional	Use this IE if context is other than default.

6.5. Call Path Optimization

If a peer is handling a call between two other IAX peers and the peer no longer has any need to monitor the progress, content, or duration of the call, it MAY remove itself from the call by directing the other two peers to communicate directly. This call path optimization, or "supervised transfer," is done in a manner that ensures the call will not be lost in the process; the initiating peer does not give up control of the process until it has confirmed the other two peers are communicating. Note: the parties involved in the call are not aware of this operation; it is purely a network operation.

```
                                 ________________
        rec  TXREJ              |                |     rec TXREL
        ----------   *--------->|      None      |<-----------------+
        snd  TXREJ              |________________|        ack       ^
        to other                  |           |                     |
                                  |           V                     |
                                  |                                 |
                                  |           *   (From All)        |
                   /Init Transfer |           | rec TXREQ           |
                    ------------  |           | ---------           |
                      snd TXREQ   |           | snd TXCNT           |
                      to both     |           |                     |
                                 _v___________v__                   |
                                |                |                  |
                                |     Begin      |----------------->+
                                |________________|                  |
                                  |           |                     |
                        rec TXACC |           | rec TXREADY         |
                        --------- |           | ---------           |
                      snd TXREADY |           |     x               |
                                  |           |                     |
                                 _v___________v__                   |
                                |                |----------------->+
                      ----------|     Ready      |----------        |
                     |          |________________|          |       |
                     |                   |                  |       |
     /Both Legs Ready|   /Both Legs Ready|       rec TXMEDIA|       |
   and not media-only|    and media-only |                  |       |
       ------------  |    ------------   |       -----------|       |
       snd TXREL     |     snd TXMEDIA   |            x     |       |
                     |                   |                  |       |
                 ____V____          _____V___            ___V_____  |
                |         |        |         |          |         | |
                | Release |        |  Media  |          | Media   | |
                |_________|        |_________|          |  Pass   | |
                                         |              |_________| |
                                         |                  |       |
                                         V                  V       |
    rec  TXCNT                           +------------------------->+
    ----------  (In any state)
    snd  TXACC
```

Figure 4: Call Path Optimization State Diagram

When a peer initiates this procedure, both call legs MUST be in the

UP state, i.e., they MUST have sent or received the ACCEPT message for that call leg. To start, it sends a TXREQ message with the addresses and information from the other remote peers to each its neighbors. If capable of performing this procedure, they begin transmitting all channel information to both the initiating peer and the new remote peer. They also send a TXCNT message indicating packet counts for the call leg to the new remote peer. Each TXCNT message is acknowledged with a TXACC message. The peers respond by sending a TXREADY message to the initiator indicating that they have confirmed the new communications path. When all remote peers have sent the initiator a TXREADY message, the transfer is successful and the initiator responds with a TXREL and has finished its involvement with the call. If during the transfer process, the two remote peers cannot communicate, they send a TXREJ message to the initiator. An example is shown in Section 9.5.

These messages are described in the sections which follow:

6.5.1. TXREQ Transfer Request Message

The TXREQ message is sent by a peer to initiate the transfer process. When sent, It MUST be sent to both adjacent peers involved in the call.

It MUST include the following Information Elements:

IE	Section	Status	Comments
Apparent Address	Section 8.4.17	Required	
Call Number	Section 8.4.20	Required	
Transfer ID	Section 8.4.26	Required	

The Apparent Address is the IP address data structure address for the other remote peer. The Call Number IE is The callid used by the other remote peer and the Transfer ID is a unique number assigned by the initiator.

Upon receipt of a TXREQ message for a valid call from the corresponding remote peer, a peer MUST respond by attempting to communicate with the newly specified remote peer. This task is accomplished by sending a TXCNT message directly to the peer at the address specified in the Apparent Address parameter.

6.5.2. TXCNT Transfer Connectivity Response Message

The TXCNT message is used to verify connectivity with a potential replacement peer for a call. It MUST include the TRANSFERID IE. Upon receipt on a message of this type, and if the peer has previously received a TXREQ for this call leg, the peer MUST respond with a TXACC message.

If the TXCNT Message is not successfully transmitted or if a TXACC message is not received in response to it, the transfer process MUST be aborted by sending a TXREJ message to the initiating host.

It MUST include the following Information Element:

IE	Section	Status	Comments
Transfer ID	Section 8.4.26	Required	A unique number assigned by the initiator.

6.5.3. TXACC Response Message

Like the TXCNT message, the TXACC message is used to verify connectivity with a potential replacement peer. It MUST include the TRANSFERID IE. Upon receipt on a message of this type if the peer is attempting to transfer this call leg, the peer stops sending call related media to the initiating peer and sends a TXREADY message to it.

It MUST include the following Information Element:

IE	Section	Status	Comments
Transfer ID	Section 8.4.26	Required	A unique number assigned by the initiator.

6.5.4. TXREADY Transfer Ready Response Message

The TXREADY message indicates that the sending peer has verified connectivity with the peer which it was instructed to transfer the call. It MUST include the TRANSFERID IE. When TXREADY messages are received from both remote peers, it MUST discontinue media transport and send a TXREL message to each peer.

It MUST include the following Information Element:

IE	Section	Status	Comments
Transfer ID	Section 8.4.26	Required	A unique number assigned by the initiator.

6.5.5. TXREL Transfer Release Response Message

The TXREL message indicates that the transfer process has successfully completed. After sending and upon receipt of this message, no further interaction (other than an ACK, of course) is needed between the peers on this call-leg. The TXREL is also used to revert a split-media call (one where the media and signaling follow different paths) to a call where the media and signaling follow the same path.

It MUST include the following Information Element:

IE	Section	Status	Comments
Call Number	Section 8.4.20	Required	

6.5.6. TXMEDIA Transfer Media Message

The TXREL message indicates that the MEDIA transfer process has successfully completed. After sending and upon processing of this message, full frames MUST continue to follow the original signaling path and media frames MUST follow the newly negotiated path. This split-path process continues until the call ends with a HANGUP or peer receives a TXREL message for the call leg. A peer MAY force the paths to rejoin by sending a TXREL message.

It MUST include the following Information Element:

IE	Section	Status	Comments
Call Number	Section 8.4.20	Required	

6.5.7. TXREJ Transfer Rejection Response Message

The TXREJ MAY be sent at anytime during the transfer process to indicate that the transfer cannot proceed. Upon receiving a TXREJ message, if the receiver is the initiating peer, it MUST form a TXREJ message and send it to the other remote peer.

The TXREJ message does not require any IEs.

6.6. Call Tear Down

The messages used to finish a call vary depending on the particular process the call is in at the time. The terminal messages for a call are:

HANGUP. See Section 6.2.5.

REJECT. See Section 6.2.4.

TRANSFER. See Section 6.4.6.

TXREADY. See Section 6.5.4.

These messages are discussed in their respective sections. Also, if the reliable transport procedures determines that messaging cannot be maintained, the call leg MUST be torn down without any other indications over the errant IAX call leg.

6.7. Network Monitoring

The IAX protocol has various tools to determine the network load. It uses the POKE message to monitor reachability of remote peer and the LAGRQ message to measure the quality of a current call leg including the jitter buffer delay.

6.7.1. POKE Request Message

A POKE message is sent to test connectivity of a remote IAX peer. It is similar to a PING message, except that it MUST be sent when there is no existing call to the remote endpoint. It MAY also be used to "qualify" a user to a remote peer, so that the remote peer can maintain awareness of the state of the user. A POKE MUST have 0 as its destination call number.

Upon receiving a POKE message, the peer MUST respond with a PONG message.

This message does not require any IEs.

6.7.2. PING Request Message

A PING message is sent to test connectivity of the remote IAX endpoint on an existing call. Transmission of a PING MAY occur when a peer-defined number of seconds have passed without receiving an incoming media frame on a call, or by default every 20 seconds. Receipt of a PING requires an acknowledging PONG be sent.

This message does not require any IEs.

6.7.3. PONG Response Message

A PONG message is a response to a PING or a POKE. It acknowledges the connection. The receiver uses the time-stamp of the received PING or POKE and its times to determine the Round Trip Time of the connection. Several receiver report IEs MAY be included with a PONG, including received jitter, received frames, delay, and dropped frames. Receipt of a PONG requires an ACK.

This message does not require any IEs.

6.7.4. LAGRQ Lag Request Message

A LAGRQ is a lag request. It is sent to determine the lag between two IAX endpoints, including the amount of time used to process a frame through a jitter buffer (if any). It requires a clock-based time-stamp, and MUST be answered with a LAGRP, which MUST echo the LAGRQ's time-stamp. The lag between the two peers can be computed on the peer sending the LAGRQ by comparing the time-stamp of the LAGRQ and the time the LAGRP was received.

This message does not require any IEs.

6.7.5. LAGRP Lag Response Message

A LAGRP is a lag reply, sent in response to a LAGRQ message. It MUST send the same time-stamp it received in the LAGRQ after passing the received frame through any jitter buffer the peer has configured.

This message does not require any IEs.

6.8. Digit Dialing

Digit Dialing support is an optional portion of the IAX protocol designed to support devices that do not maintain their own dial plans, for instance, analog telephone adapters, or ATAs. The dialing portion of the IAX protocol MAY be implemented for the client/ phone-side, server side or not all. The exchanges work as a series

of Dialing Plan requests (DPREQ) each followed by a response (DPREP) indicating if additional digits SHOULD be collected before sending the call. The sections that follow describe these messages and the rules associated with them.

6.8.1. DPREQ Dial Plan Request Message

A DPREQ is a request for the server to analyze the passed called number and determine if there is a valid dialing pattern on the remote peer. It MUST include the 'called number' IE to specify what extension is being queried. This command is used in the case where a local peer does not handle its own dialplan/extension switching. The local peer can inquire (as a user dials) how the remote peer perceives the 'called number'. If a DPREP is received indicating that the number is valid, a DIAL MAY be sent.

This message MAY be sent by the client and MUST be implemented on servers which provide IAX dialing support.

It MUST include the following Information Element:

IE	Section	Status	Comments
Call Number	Section 8.4.20	Required	

6.8.2. DPREP Dial Plan Response Message

A DPREP is a reply to a DPREQ, containing the status of the dialplan entry requested in the 'called number' IE of the DPREQ. It MUST include the 'called number', 'dpstatus', and 'refresh' IEs. The called number is the same one received in the 'called number' IE of the DPREQ. The 'dpstatus' IE contains the status of the dialplan entry referenced by the received called number. The status indicates whether the called number exists, can exist, needs more digits, or is invalid. More information can be found in Section 8.4 under the DPSTATUS information element. The 'refresh' IE specifies the number of minutes the 'dpstatus' is valid. If the 'refresh' IE is not present, a default 10 minutes period is assumed.

The sending of this message MUST be implemented by servers which support IAX dialing. Clients which support IAX dialing MUST be capable of receiving such messages.

It MUST include the following Information Elements:

IE	Section	Status	Comments
Call Number	Section 8.4.20	Required	
Dial Plan Status	Section 8.4.20	Required	Indicates if number exists, is a partial match, etc.
Dial Plan Refresh	Section 8.4.20	Optional	Inclusion is strongly suggested. The default is 10 minutes.

6.8.3. DIAL Request Message

The DIAL message is used with IAX peers that do not maintain their own dialplan/extension routing. Once an extension is validated by one or more DPREQ/DPREP exchanges, the number MAY be dialed in a DIAL message, using the 'called number' IE to specify the extension it is attempting to reach. The remote peer then handles the remaining aspects of call setup, including ringing the extension and notifying the local peer when it has been answered following the same requirements as the NEW command (Section 6.2.2).

The following table specifies the IEs used by this message:

IE	Section	Status	Comments
Called Number	Section 8.4.1	Required	
Called Context	Section 8.4.5	Optional	Use this IE if context is other than default.

6.9. Miscellaneous

6.9.1. ACK acknowledgement Message

An ACK acknowledges the receipt of an IAX message. An ACK is sent upon receipt of a full frame which does not have any other protocol-defined response. An ACK MUST have both a source call number and destination call number. It MUST also not change the sequence number

counters, and MUST return the same time-stamp it received. This time-stamp allows the originating peer to determine to which message the ACK is responding. Receipt of an ACK requires no action.

An ACK MAY also be sent as an initial acknowledgment of an IAX message which requires some other protocol-defined message acknowledgment, as long as the required message is also sent within some peer-defined amount of time. This allows the acknowledging peer to delay transmission of the proper IAX message, which may add security against brute-force password attacks during authentication exchanges.

When the following messages are received, an ACK MUST be sent in return: NEW, HANGUP, REJECT, ACCEPT, PONG, AUTHREP, REGREL, REGACK, REGREJ, TXREL. ACKs SHOULD not be expected by any peer and their purpose is purely to force the transport layer to be up to date.

The ACK message does not requires any IEs.

6.9.2. INVAL Invalid Response Message

An INVAL is sent as a response to a received message that is not valid. This occurs when an IAX peer sends a message on a call after the remote peer has hungup its end. Upon receipt of an INVAL, a peer MUST destroy its side of a call.

The INVAL message does not requires any IEs.

6.9.3. VNAK Voice Negative Acknowledgement Message

A VNAK is sent when a message is received out of order, particularly when a mini frame is received before the first full voice frame on a call. It is a request for retransmission of dropped messages. A message is considered out of sequence if the received iseqno is different than the expected iseqno. On receipt of a VNAK, a peer MUST retransmit all frames with a higher sequence number than the VNAK message's iseqno.

The VNAK message does not requires any IEs.

6.9.4. MWI Message Waiting Indicator Request Message

An MWI message is used to indicate to a remote peer that it has one or more messages waiting. It MAY include the 'msgcount' IE to specify how many messages are waiting.

The following table specifies IEs used by this message

IE	Section	Status	Comments
MSGCOUNT	Section 8.4.23	Optional	Suggested.

6.9.5. UNSUPPORT Unsupported Response Message

An UNSUPPORT message is sent in response to a message that is not supported by an IAX peer. This occurs when an IAX command with an unrecognized or unsupported subclass is received. No action is required upon receipt of this message, though the peer SHOULD be aware that the message referred to in the optionally included 'IAX unknown' IE is not supported by the remote peer.

The following table specifies IEs used by this message

IE	Section	Status	Comments
UNKNOWN	Section 8.4.22	Optional	Suggested.

6.10. Media Messages

The IAX protocol supports many types of media and these are transported through the same UDP port as other IAX messages. Voice and video are unique in that they utilize two different encodings each with a different support procedures. Abbreviated 'Mini frames' are normally used for audio and video, however, each time the time-stamp is a multiple of 32,768 (0x8000 hex), a standard or 'Full Frame' MUST be sent. This approach facilitates efficiency and reliability by sending compressed packets, without guaranteed delivery, most of the time while periodically forcing reliable exchanges with the peer. If communication fails, call tear-down procedures are invoked.

Upon receiving any media message, except the abbreviated audio and video mini frames, an ACK message MUST be sent. The content SHOULD be passed to an associated application, device, or call leg. The data MAY be buffered before it is presented to the user.

6.10.1. DTMF Media Message

The message carries a single digit of DTMF (Dual Tone Multiple Frequency). Useful background information about DTMF can be found in

[RFC4733] and [RFC4734], but, note that IAX does not use the RTP protocol.

6.10.2. Voice Media Message

The message carries voice data and indicates the CODEC used.

6.10.3. Video Media Message

The frame carries video data and indicates the video format of the data.

6.10.4. Text Media Message

The frame carries a text message in UTF-8 [RFC3629] format.

6.10.5. Image Media Message

This message carries a single image. The image MUST fit in one message in this version of the protocol.

6.10.6. HTML Media Message

The HTML message class carries HTML and related data as well as status about the display of that HTML page. The subclass parameter indicates the HTML content type. It MAY be a URL, the start, middle or end of a data block. HTML data MUST be in the format described in [html401].

If a peer receives an HTML message for a channel that does not support HTML, it MUST respond with an HTML message that has the HTML NOT SUPPORTED indication.

When a devices that supports HTML completes loading the page, it SHOULD send a LOAD COMPLETE message

6.10.7. Comfort Noise Media Message

This message indicates that comfort noise SHOULD be played. It has a parameter that indicates the level. The noise is to be locally generated.

7. Message Transport

IAX is sent over UDP and uses an application level protocol to provide reliable transport where needed.

With respect to transport, there are two message formats: reliable or 'Full Frames' and unacknowledged 'Mini' or 'Meta' frames. All messages except certain voice and video messages are reliable. Reliable messages are transported by a scheme which maintains message counts and time stamps for both peers involved in the call. The counts are per call. Each peer maintains a timer for all reliable messages and MUST periodically retransmit those messages until they acknowledge or the retry limit is exceeded.

When starting a call, the outgoing and incoming message sequence numbers MUST both be set to zero. Each reliable message that is sent increments the message count by one except the ACK, INVAL, TXCNT, TXACC, and VNAK messages which do not change the message count. The message includes the outgoing message count and the highest numbered incoming message which has been received. In addition, it contains a time-stamp which represents the number of milliseconds since the call started. Or, in the case of certain network timing messages, it contains a copy of the time-stamp sent to it. Time-stamps MAY be approximate, but, MUST be in order.

When any message is received, the time-stamps MUST be checked to make sure that they are in order. If a message is received out of order, it MUST be ignored and a VNAK message sent to resynchronize the peers. And if the message is a reliable message, the incoming message counter MUST be used to acknowledge all the messages up to that sequence number which have been sent.

If no acknowledgment is received after a locally configured number of retries (default 4) the call leg SHOULD be considered unusable and the call MUST be torn down without any further interaction on this call leg.

7.1. Trunking

IAX allows multiple media exchanges between the same two peers to be multiplexed into a single trunk call coalescing media payload into a combined packet. This decreases bandwidth usage as there are fewer total packets being transmitted. Trunking MAY occur in one or both directions of an IAX exchange. A trunk consists of a trunk header and one or more trunked IAX calls. The trunk message contains a time-stamp specifying the time of transmission of the trunk frame. The audio data from the trunked calls are encapsulated in the trunk frame following the header. Each trunked call consists of two octets

specifying the call's source number, two octets specifying the length in octets of the media data, and the media data itself. IAX permits transmitting the time-stamps of each encapsulated mini frame as well, so that accurate timing information can be used for jitter buffers, etc. A flag in the meta command header specifies whether the encapsulated mini frames retain their original time-stamps. If they do not retain them, they MUST assume the time-stamp in the trunk header upon being received by the trunk peer.

7.2. Timers

There are various timers in the IAX protocol. There are other application level timers, such as the call timer and ring timer, which are beyond the scope of this document. This section describes the IAX timers and specifies their default values and behavior.

7.2.1. Retransmission Timer

The message retransmission procedures are described in Section 7. On each call, there is a timer for how long to wait for an acknowledgment of a message. This timer starts at twice the measured round trip time from the last PING/PONG command. If a retransmission is needed, it is exponentially increased until it meets a boundary value. The maximum retry time period boundary is 10 seconds.

7.2.2. Registration Period Timer

Registrations are valid for a specified time period. It is the client's responsibility to renew this registration before the time period expires. The registrations SHOULD be renewed at random intervals to prevent network congestion. A registrar MUST monitor this time period and invalidate the registration if the client/ registrant has not renewed their registration before the timer elapses.

7.3. NAT Considerations

IAX is very well suited to operating behind NAT due to its single port approach. This approach eliminates any start of call media stream delays while the NAT gateway establishes a bidirectional port association. Deploying a single IAX server behind a NAT gateway requires little effort. If the server acts as a registrar, the IAX UDP port on the NAT gateway must be forwarded to the server. If the server acts as a registrant, the default, 60 second, REGREQ refresh timer should be sufficient to maintain a port association in the NAT gateway, however, a static port mapping is preferred.

If multiple servers are to be deployed behind a single NAT gateway,

most NAT gateways require each IAX server to use different UDP ports. Of course, there may be NAT implementations which recognize when multiple devices utilize the same private port and and manage it appropriately.

7.4. Encryption

IAX supports call encryption using the symmetric key, Rijndael [AES] block cipher (also called AES---Advanced Encryption Standard). Rijndael is a 128-bit block cipher utilizing a shared secret. IAX encrypts on a call-by-call basis starting with a plain-text NEW message indicating, in addition to the other message parameters, that the call should be encrypted. This indication is given by sending the ENCRYPTION IE (Section 8.4.34) in the NEW request message. If the called host supports encryption, it will respond with a plain-text AUTHREQ message which also includes the ENCRYPTION IE. All subsequent messages in the call MUST be encrypted. If the called host does not support encryption, the AUTHREQ sent in response to the NEW must not include the ENCRYPTION IE and the calling host MUST either HANGUP the request or continue with the unencrypted call.

The key to use in encrypting the messages is computed by taking the CHALLENGE IE Section 8.4.14 from the AUTHREQ and concatenating any one of the shared passwords then computing the 128-bit MD5 digest of this combination. To decrypt, if there is more than one password for the peer, each must be tried until the message is successfully decoded. The key remains constant for the duration of the call. Only the data portion of the messages are encoded.

8. Message Encoding

8.1. Frame Structure

This section contains the specification for each type of frame that IAX defines.

8.1.1. Full Frames

Full frames can send signaling or media data. Generally full frames are used to control initiation, setup, and termination of an IAX call, but they can also be used to carry stream data (though this is generally not optimal).

Full frames are sent reliably, so all full frames require an immediate acknowledgment upon receipt. This acknowledgment can be explicit via an 'ACK' message (see Section 8.2.12) or implicit based upon receipt of an appropriate response to the full frame issued.

The standard full frame header length is 12 octets.

Field descriptions:

'F' bit

This bit specifies whether the frame is a full frame or not. If the 'F' bit is set to 1 the frame is a full frame. If it is set to 0 it is not a full frame.

Source call number

This 15-bit value specifies the call number the transmitting client uses to identify this call. The source call number for an active call MUST NOT be in use by another call on the same client. Call numbers MAY be reused once a call is no longer active, i.e. when either there is positive acknowledgment that the call has been destroyed or when all possible timeouts for the call have expired.

'R' bit

This bit specifies whether the frame is being retransmitted or not. If the 'R' bit is set to 0 the frame is being transmitted for the first time. If it is set to 1 the frame is being retransmitted. IAX does not specify a retransmit timeout; this is left to the implementor.

Destination call number

This 15-bit value specifies the call number the transmitting client uses to reference the call at the remote peer. This number is the same as the remote peer's source call number. The destination call number uniquely identifies a call on the remote peer. The source call number uniquely identifies the call on the local peer.

Time-stamp

The time-stamp field contains a 32-bit time-stamp maintained by an IAX peer for a given call. The time-stamp is an incrementally increasing representation of the number of milliseconds since the first transmission of the call.

OSeqno

The 8-bit OSeqno field is the outbound stream sequence number. Upon initialization of a call, its value is 0. It increases incrementally as full frames are sent. When the counter overflows, it silently resets to 0.

ISeqno

The 8-bit ISeqno field is the inbound stream sequence number. Upon initialization of a call its value is 0. It increases incrementally as full frames are received. At any time the ISeqno of a call represents the next expected inbound stream sequence number. When the counter overflows, it silently resets to 0.

Frametype

The Frametype field identifies the type of message carried by the frame. See Section 8.2 for more information.

'C' bit

This bit determines how the remaining 7 bits of the Subclass field are coded. If the 'C' bit is set to 1, the Subclass value is interpreted as a power of 2. If it is not set, the Subclass value is interpreted as a simple seven-bit unsigned integer.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|     Source Call Number      |R|   Destination Call Number   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            time-stamp                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    OSeqno     |    ISeqno     |   Frame Type  |C|  Subclass   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :                             Data                              :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Full Frame Binary Format

8.1.2. Mini frames

Mini Frames are so named because their header is a minimal 4 octets. Mini frames carry no control or signaling data; their sole purpose is to carry a media stream on an already-established IAX call. They are sent unreliably. This decision was made because VoIP calls typically can miss several frames without significant degradation in call quality while the incurred overhead in ensuring reliability increases bandwidth requirements and decreases throughput. Further, because voice calls are typically sent in real time, lost frames are too old to be reintegrated into the audio stream by the time they can be retransmitted.

Field descriptions:

'F' bit

Mini frames MUST have the 'F' bit set to 0 to specify that they are not full frames.

Source call number

The source call number is the number that is used by the transmitting peer to identify the current call.

time-stamp

Mini frames carry a 16-bit time-stamp, which is the lower 16 bits of the transmitting peer's full 32-bit time-stamp for the call. The time-stamp allows synchronization of incoming frames so that they MAY be processed in chronological order instead of the (possibly different) order in which they are received. The 16-bit

time-stamp wraps after 65.536 seconds, at which point a full frame SHOULD be sent to notify the remote peer that its time-stamp has been reset. A call MUST continue to send mini frames starting with time-stamp 0 even if acknowledgment of the resynchronization is not received.

The F bit, source call number, and 16-bit time-stamp comprise the entire four octet header for a full frame. Following this header is the actual stream data, of arbitrary length, up to the maximum supported by the network.

Mini frames are implicitly defined to be of type 'voice frame' (frametype 2; see Section 8.2). The subclass is implicitly defined by the most recent full voice frame of a call (i.e. the subclass for a voice frame specifies the CODEC used with the stream). The first voice frame of a call SHOULD be sent using the CODEC agreed upon in the initial CODEC negotiation. On-the-fly CODEC negotiation is permitted by sending a full voice frame specifying the new CODEC to use in the subclass field.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     Source call number      |             time-stamp        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                               Data                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Mini Frame Binary Format

8.1.3. Meta frames

Meta frames serve one of two purposes. Meta video frames allow the transmission of video streams with an optimized header. They are similar in purpose to mini voice frames. Meta trunk frames are used for trunking multiple IAX media streams between two peers into one header, to further minimize bandwidth consumption.

8.1.3.1. Meta Video Frames

Field descriptions:

'F' bit

Meta video frames MUST have the 'F' bit set to 0 to indicate that they are not full frames.

Meta Indicator

The meta indicator is a 15-bit field of all zeroes, used to indicate that the frame is a meta frame. Meta frames are identifiable because the first 16 bits will always be zero in any meta frame, whereas full or mini frames will have either the 'F' bit set or some (nonzero) value for the source call number (or both).

'V' bit

The 'V' bit in a meta video frame is set to 1 to specify that the frame is a meta video frame.

Source call number

The call number that is used by the transmitting peer to identify this video call.

time-stamp

Meta video frames carry a 16-bit time-stamp, which is the lower 16 bits of the transmitting peer's full 32-bit time-stamp for the call. When this time-stamp wraps, a full frame SHOULD be sent to notify the remote peer that the time-stamp has been reset to 0.

Following the time-stamp is the actual video stream data. Meta video frames are implicitly defined to be of type 'video frame' (frametype 3; see Section 8.2). The video CODEC used is implicitly defined by the subclass of the most recent full video frame of a call.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|         Meta Indicator      |V|      Source Call Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|?|          time-stamp         |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                         Data                  |
:                                                               :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Meta Video Frame Binary Format

8.1.3.2. Meta Trunk Frames

IAX natively supports two methods of trunking multiple media streams between two peers into a single association. The first method sends a standard meta header, along with a single 32-bit time-stamp describing the transmission time of the trunk frame. Following the time-stamp are one or more media frames consisting of the call number and the length in octets of the stream data included in the frame.

The second method of trunking is very similar to the first. It sends a standard meta header, including the 32-bit time-stamp describing the time of transmission of the trunk frame. But the media frames included in the trunk are actually complete mini frames, including the 16-bit time-stamp for each call. The first method uses slightly less bandwidth (2 fewer octets per call in the trunk), while the second method maintains the individual time-stamps for each call so that jitter buffering can use the actual time-stamps associated with a call instead of the (less accurate) time-stamp representing the entire trunk. Either method is permissible for trunking.

Field descriptions:

'F' bit

Meta trunk frames MUST have the 'F' bit set to 0 to indicate that they are not full frames.

Meta Indicator

The meta indicator is a 15-bit field of all zeroes, used to indicate that the frame is a meta frame. Meta frames are identifiable because the first 16 bits will always be zero in any meta frame, whereas full or mini frames will have either the 'F' bit set or some (nonzero) value for the source call number (or both).

'V' bit

The 'V' bit in a meta trunk frame is set to 0 to specify that the frame is not a meta video frame.

Meta Command

This seven bit field identifies whether the meta frame is a trunk or not. A value of '1' indicates that the frame is a meta trunk frame. All other values are reserved for future use.

Command Data

This 8-bit field specifies flags for options which apply to a trunked call. The least significant bit of the field is the 'trunk time-stamps' flag. A value of 0 indicates that the calls in the trunk do not include their individual time-stamps. A value of 1 indicates that the calls do each include their own time-stamp. All other bits are reserved for future use.

Time-stamp

Meta trunk frames carry a 32-bit time-stamp, which represents the actual time of transmission of the trunk frame. This is distinct from the time-stamps of the calls included in the trunk.

Following the 32-bit time-stamp is one or more trunked calls. If the 'trunk time-stamps' flag is set to 0, each entry consists of 2 octets specifying the source call number of the call, 2 octets specifying the length in octets of the media data, and then the media data. If the 'trunk time-stamps' flag is set to 1, each entry consists of 2 octets specifying the length in octets of the media data, and then a mini frame (2 octets specifying source call number, 2 octets specifying 16-bit time-stamp, and the media data). The following two diagrams help illustrate pictorially this structure.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|         Meta Indicator      |V|Meta Command | Cmd Data (0)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            time-stamp                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |R|      Source Call Number     |     Data Length (in octets)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :                             Data                              :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   .
                                   .
                                   .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |R|      Source Call Number     |     Data Length (in octets)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :                             Data                              :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Meta Trunk Frame Binary Format (trunk time-stamps 0)

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|         Meta Indicator      |V|Meta Command | Cmd Data (1)  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            time-stamp                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Data Length (in octets)   |R|     Source Call Number      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           time-stamp          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                       Data                    |
   :                                                               :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   .
                                   .
                                   .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Data Length (in octets)   |R|     Source Call Number      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           time-stamp          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                       Data                    |
   :                                                               :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Meta Trunk Frame Binary Format (trunk time-stamps 1)

8.1.4. Encrypted Frames

All of the above frames may be encrypted. The header call numbers are passed through in the clear, first 4 bytes for a full frame or 2 bytes for a mini frame. The remainder of the frame is padded with between 16 and 32 bytes of random data, then encrypted with AES each block being xor'd with the previous block. The padding is added at the front of the data.

Figure 10 shows a padded full frame before encryption. and Figure 11 shows the frame after encryption. Other frame types follows the same procedure, except the clear text portion is shorter, as described above.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|     Source Call Number      |R|   Destination Call Number   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         12 Random bytes                       |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               28  Random bits                        |padding|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   : between 0 and 15 (as indicated by the padding field above)    :
   :                         Random bytes                          :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   :                    Remainder of Actual Frame                  :
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Full Frame before encryption

Since AES requires a 16 byte block size, some padding is essential. This padding has been placed at the beginning of the payload because it makes it more difficult to take advantage of the predictability of the IAX frame header. For example, the first encrypted Frame an IAX client sends within an incoming IAX call is entirely predictable: It is always an ACK - where even the time-stamp is guessable as it is the time the AUTHREP packet was sent.

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|     Source Call Number      |R|   Destination Call Number   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Encrypted data                        |
   |                Multiple of 16 bytes                           |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Frame after encryption

The same encryption rules apply to the miniframes, except that the initial unencrypted portion is only 2 bytes.

8.2. Frame Types

The IAX protocol specifies 10 types of possible frames for the "frametype" field of a full frame. They are:

8.2.1. DTMF Frame

The frame carries a single digit of DTMF (Dual Tone Multiple Frequency). More information about DTMF can be found in RFC 4733[RFC4733] and [RFC4734].

For DTMF frames, the subclass is the actual DTMF digit carried by the frame.

8.2.2. Voice Frame

The frame carries voice data.

The subclass specifies the audio format of the data. Predefined voice formats can be found in Section 8.5 below.

8.2.3. Video Frame

The frame carries video data.

The subclass specifies the video format of the data. Predefined video formats can be found in Section 8.5 below.

8.2.4. Control Frame

The frame carries session control data, i.e. it refers to control of a device connected to an IAX endpoint.

The subclass is a value from Section 8.2.11 describing the device control signal.

8.2.5. Null Frame

Frames with the Null value MUST NOT be transmitted.

8.2.6. IAX Frame

The frame carries control data that provides IAX protocol specific endpoint management. This frametype is used to manage IAX protocol interactions that are generally independent of the type of endpoints.

The subclass is a value from Section 8.2.12 describing an IAX event.

8.2.7. Text Frame

The frame carries a non-control text message in UTF-8 [RFC3629] format.

All text frames have a subclass of 0.

8.2.8. Image Frame

The frame carries a single image.

The subclass describes the format of the image from Section 8.5 below.

8.2.9. HTML Frame

The frame carries HTML data.

The subclass is a value from the HTML Subclasses table in Section 8.3.

8.2.10. Comfort Noise Frame

The frame carries comfort noise.

The subclass is the level of comfort noise in -dBov.

The following table specifies all valid Frame Type Values:

TYPE	Description	Subclass Description	Data Description
0x01	DTMF	0-9, A-D, *, #	Undefined
0x02	Voice	Audio Compression Format	Data
0x03	Video	Video Compression Format	Data
0x04	Control	See Control Frame Types	Varies with subclass
0x05	Null	Undefined	Undefined
0x06	IAX Control	See IAX Protocol Messages	Information Elements
0x07	Text	Always 0	Raw Text
0x08	Image	Image Compression Format	Raw image
0x09	HTML	See HTML Frame Types	Message Specific
0x0A	Comfort Noise	Level in -dBov of comfort noise	None

8.2.11. Control Frames

The following table specifies all valid Control Frame Subclasses:

VALUE	Name	Description
0x01	Hangup	The call has been hungup at the remote end.
0x02	Reserved	Reserved for future use
0x03	Ringing	Remote end is ringing (ringback)
0x04	Answer	Remote end has answered
0x05	Busy	Remote end is busy
0x06	Reserved	Reserved for future use
0x07	Reserved	Reserved for future use
0x08	Congestion	The call is congested.
0x09	Flash Hook	Flash hook
0x0a	Reserved	Reserved for future use
0x0b	Option	Device-specific options are being transmitted
0x0c	Key Radio	Key Radio
0x0d	Unkey Radio	Unkey Radio
0x0e	Call Progress	Call is in progress
0x0f	Call Proceeding	Call is proceeding
0x10	Hold	Call is placed on hold
0x11	Unhold	Call is taken off hold

8.2.12. IAX Frames

Frames of type 'IAX' are used to provide management of IAX endpoints. They handle IAX signaling (e.g. call setup, maintenance, and tear-down). They MAY also handle direct transmission of media data, but this is not optimal for VoIP calls. They do not carry session-specific control (e.g., device state), as this is the purpose of Control Frames. The IAX commands are listed and described below.

The following table specifies all valid IAX Frame Values:

Hex	Name	Description
0x01	NEW	Initiate a new call
0x02	PING	Ping request
0x03	PONG	Ping or poke reply
0x04	ACK	Explicit acknowledgment
0x05	HANGUP	Initiate call tear-down
0x06	REJECT	Reject a call
0x07	ACCEPT	Accept a call
0x08	AUTHREQ	Authentication request
0x09	AUTHREP	Authentication reply
0x0a	INVAL	Invalid message
0x0b	LAGRQ	Lag request
0x0c	LAGRP	Lag reply
0x0d	REGREQ	Registration request
0x0e	REGAUTH	Registration authentication
0x0f	REGACK	Registration acknowledgement
0x10	REGREJ	Registration reject
0x11	REGREL	Registration release

0x12	VNAK	Video/Voice retransmit request
0x13	DPREQ	Dialplan request
0x14	DPREP	Dialplan reply
0x15	DIAL	Dial
0x16	TXREQ	Transfer request
0x17	TXCNT	Transfer connect
0x18	TXACC	Transfer accept
0x19	TXREADY	Transfer ready
0x1a	TXREL	Transfer release
0x1b	TXREJ	Transfer reject
0x1c	QUELCH	Halt audio/video [media] transmission
0x1d	UNQUELCH	Resume audio/video [media] transmission
0x1e	POKE	Poke request
0x1f	Reserved	Reserved for future use
0x20	MWI	Message waiting indication
0x21	UNSUPPORT	Unsupported message
0x22	TRANSFER	Remote transfer request
0x23	Reserved	Reserved for future use
0x24	Reserved	Reserved for future use
0x25	Reserved	Reserved for future use

8.3. HTML Command Subclasses

IAX HTML Command Subclasses:

NUMBER	DESCRIPTION
1	Sending a URL
2	Data frame
4	Beginning frame
8	End frame
16	Load is complete
17	Peer does not support HTML
18	Link URL
19	Unlink URL
20	Reject Link URL

8.4. Information Elements

IAX messages sent as full frames MAY carry information elements to specify user- or call-specific data. Information elements are appended to a frame header in its data field. Zero, one, or multiple information elements MAY be included with any IAX message.

Information elements are coded as follows:

The first octet of any information element consists of the "IE" field. The IE field is an identification number which defines the particular information element. Table 1 lists the defined information elements and each information element is defined below the table.

The second octet of any information element is the "data length" field. It specifies the length in octets of the information element's data field.

The remaining octet(s) of an information element contain the actual data being transmitted. The representation of the data is dependent on the particular information element as identified by

its "IE" field. Some information elements carry binary data, some carry UTF-8 [RFC3629] data, and some have no data field at all. Elements which carry UTF-8 MUST prepare strings as per [RFC3454] and [RFC3491], so that illegal characters, case folding and other characters properties are handled and compared properly. The data representation for each information element is described below.

The following table specifies the Information Element Binary Format:

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      IE       |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:             DATA              :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following is a table of the information elements IAX defines, and a brief description of each information element's purpose. More information about each IE may be found below the table.

HEX	NAME	DESCRIPTION
HEX	NAME	DESCRIPTION
0x01	CALLED NUMBER	Number/extension being called
0x02	CALLING NUMBER	Calling number
0x03	CALLING ANI	Calling number ANI for billing
0x04	CALLING NAME	Name of caller
0x05	CALLED CONTEXT	Context for number
0x06	USERNAME	Username (peer or user) for authentication
0x07	PASSWORD	Password for authentication
0x08	CAPABILITY	Actual CODEC capability

0x09	FORMAT	Desired CODEC format
0x0a	LANGUAGE	Desired language
0x0b	VERSION	Protocol version
0x0c	ADSICPE	CPE ADSI capability
0x0d	DNID	Originally dialed DNID
0x0e	AUTHMETHODS	Authentication method(s)
0x0f	CHALLENGE	Challenge data for MD5/RSA
0x10	MD5 RESULT	MD5 challenge result
0x11	RSA RESULT	RSA challenge result
0x12	APPARENT ADDR	Apparent address of peer
0x13	REFRESH	When to refresh registration
0x14	DPSTATUS	Dialplan status
0x15	CALLNO	Call number of peer
0x16	CAUSE	Cause
0x17	IAX UNKNOWN	Unknown IAX command
0x18	MSGCOUNT	How many messages waiting
0x19	AUTOANSWER	Request auto-answering
0x1a	MUSICONHOLD	Request musiconhold with QUELCH
0x1b	TRANSFERID	Transfer Request Identifier
0x1c	RDNIS	Referring DNIS
0x1d	Reserved	Reserved for future use
0x1e	Reserved	Reserved for future use
0x1f	DATETIME	Date/Time
0x20	Reserved	Reserved for future use

0x21	Reserved	Reserved for future use
0x22	Reserved	Reserved for future use
0x23	Reserved	Reserved for future use
0x24	Reserved	Reserved for future use
0x25	Reserved	Reserved for future use
0x26	CALLINGPRES	Calling presentation
0x27	CALLINGTON	Calling type of number
0x28	CALLINGTNS	Calling transit network select
0x29	SAMPLINGRATE	Supported sampling rates
0x2a	CAUSECODE	Hangup cause
0x2b	ENCRYPTION	Encryption format
0x2c	ENCKEY	Reserved for future Use
0x2d	CODEC PREFS	CODEC Negotiation
0x2e	RR JITTER	Received jitter, as in rfc3550
0x2f	RR LOSS	Received loss, as in rfc3550
0x30	RR PKTS	Received frames
0x31	RR DELAY	Max playout delay for received frames in ms
0x32	RR DROPPED	Dropped frames (presumably by jitter buffer)
0x33	RR OOO	Frames received Out of Order

Table 1: Information Element Definitions

8.4.1. CALLED NUMBER

The purpose of the CALLED NUMBER information element is to indicate the number or extension being called. It carries UTF-8-encoded data. The CALLED NUMBER information element MUST use UTF-8 encoding and not

numeric data because destinations are not limited to E.164 numbers ([E164]), national numbers, or even digits. It is possible for a number or extension to include non-numeric characters. The CALLED NUMBER IE MAY contain a SIP URI, [RFC3261] or a URI in any other format. The ability to serve a CALLED NUMBER is server dependent.

The CALLED NUMBER information element is generally sent with IAX NEW, DPREQ, DPREP, DIAL, and TRANSFER messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x01     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:  UTF-8-encoded CALLED NUMBER  :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.2. CALLING NUMBER

The purpose of the CALLING NUMBER information element is to indicate the number or extension of the calling entity to the remote peer. It carries UTF-8-encoded data.

The CALLING NUMBER information element is usually sent with IAX NEW messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x02     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
: UTF-8-encoded CALLING NUMBER  :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.3. CALLING ANI

The purpose of the CALLING ANI information element is to indicate the calling number ANI (Automatic number identification) for billing. It carries UTF-8-encoded data.

The CALLING ANI information element MAY be sent with an IAX NEW message, but it is not required.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x03     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:   UTF-8-encoded CALLING ANI   :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.4. CALLING NAME

The purpose of the CALLING NAME information element is to indicate the calling name of the transmitting peer. It carries UTF-8-encoded data.

The CALLING NAME information element is usually sent with IAX NEW messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x04     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:   UTF-8-encoded CALLING NAME  :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.5. CALLED CONTEXT

The purpose of the CALLED CONTEXT information element is to indicate the context (or partition) of the remote peer's dialplan that the CALLED NUMBER is interpreted. It carries UTF-8-encoded data.

The CALLED CONTEXT information element MAY be sent with IAX NEW or TRANSFER messages, though it is not required.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x05     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
: UTF-8-encoded CALLED CONTEXT  :
|                               |
```

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.6. USERNAME

The purpose of the USERNAME information element is to specify the identity of the user participating in an IAX message exchange. It carries UTF-8-encoded data.

The USERNAME information element MAY be sent with IAX NEW, AUTHREQ, REGREQ, REGAUTH, or REGACK messages, or any time a peer needs to identify a user.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x06     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :     UTF-8-encoded USERNAME    :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.7. CAPABILITY

The purpose of the CAPABILITY information element is to indicate the media CODEC capabilities of an IAX peer. Its data is represented in a four octet bitmask according to Section 8.5. Multiple CODECs MAY be specified by logically OR'ing them into the CAPABILITY information element.

The CAPABILITY information element is sent with IAX NEW messages if appropriate for the CODEC negotiation method the peer is using.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x08     |      0x04     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | CAPABILITY according to Media |
   | Format Subclass Values Table  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.8. FORMAT

The purpose of the FORMAT information element is to indicate a single preferred media CODEC. When sent with a NEW message, the indicated CODEC is the desired CODEC an IAX peer wishes to use for a call. When sent with an ACCEPT message, it indicates the actual CODEC that has been selected for the call. Its data is represented in a four octet bitmask according to Section 8.5. Only one CODEC MUST be specified in the FORMAT information element.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x09     |      0x04     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   FORMAT according to Media   |
| Format Subclass Values Table  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.9. LANGUAGE

The purpose of the LANGUAGE information element is to indicate the language in which the transmitting peer would like the remote peer to send signaling information. It carries UTF-8-encoded data and tags should be selected per [RFC4646] and [RFC4647].

The LANGUAGE information element MAY be sent with an IAX NEW message.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x0a     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:     UTF-8-encoded LANGUAGE    :
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.10. VERSION

The purpose of the VERSION information element is to indicate the protocol version the peer is using. Peers at each end of a call MUST use the same protocol version. Currently the only supported version is 2. The data field of the VERSION information element is 2 octets long.

The VERSION information element MUST be sent with an IAX NEW message.

When sent, the VERSION information element MUST be the first IE in the message.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x0b     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0002             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.11. ADSICPE

The purpose of the ADSICPE information element is to indicate the CPE ADSI capability. The data field of the ADSICPE information element is 2 octets long.

The ADSICPE information element MAY be sent with an IAX NEW message.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x0c     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       ADSICPE Capability      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.12. DNID

The purpose of the DNID information element is to indicate the Dialed Number ID, which may differ from the 'called number'. It carries UTF-8-encoded data.

The DNID information element MAY be sent with an IAX NEW message.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x0d     |  Data Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
:    UTF-8-encoded DNID Data    :
|                               |
```

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.13. AUTHMETHODS

The purpose of the AUTHMETHODS information element is to indicate the authentication methods a peer accepts. It is sent as a bitmask 2 octets long. The table below lists the valid authentication methods.

The AUTHMETHODS information element MUST be sent with IAX AUTHREQ and REGAUTH messages.

```
                       1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x0e     |      0x02     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Valid Authentication Methods |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following table lists valid values for authentication:

METHOD	DESCRIPTION
0x0001	Reserved (was Plaintext)
0x0002	MD5
0x0004	RSA

8.4.14. CHALLENGE

The purpose of the CHALLENGE information element is to offer the MD5 or RSA challenge to be used for authentication. It carries the actual UTF-8-encoded challenge data.

The CHALLENGE information element MUST be sent with IAX AUTHREQ and REGAUTH messages.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x0f     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :  UTF-8-encoded Challenge Data :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.15. MD5 RESULT

The purpose of the MD5 RESULT information element is to offer an MD5 response to an authentication CHALLENGE. It carries the actual UTF-8-encoded challenge result.

The MD5 RESULT information element MAY be sent with IAX AUTHREP and REGREQ messages if an AUTHREQ or REGAUTH and appropriate CHALLENGE has been received. This information element MUST NOT be sent except in response to a CHALLENGE.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     0x10      |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :    UTF-8-encoded MD5 Result   :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.16. RSA RESULT

The purpose of the RSA RESULT information element is to offer an RSA response to an authentication CHALLENGE. It carries the actual UTF-8-encoded challenge result.

The RSA RESULT information element MAY be sent with IAX AUTHREP and REGREQ messages if an AUTHREQ or REGAUTH and appropriate CHALLENGE have been received. This information element MUST NOT be sent except in response to a CHALLENGE.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x11     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :    UTF-8-encoded RSA Result   :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.17. APPARENT ADDR

The purpose of the APPARENT ADDR information element is to indicate the perceived network connection information used to reach a peer, which may differ from the actual address when the peer is behind NAT. The APPARENT ADDR IE is populated using the source address values of the UDP and IP headers in the IAX message to which this response is generated. The data field of the APPARENT ADDR information element is the same as the POSIX sockaddr struct for the address family in use (i.e., sockaddr_in for IPv4, sockaddr_in6 for IPv6). The data length depends on the type of address being represented.

The APPARENT ADDR information element MUST be sent with IAX TXREQ and REGACK messages. When used with a TXREQ message, the APPARENT ADDR MUST specify the address of the peer the local peer is trying to transfer its end of the connection to. When used with a REGACK message, the APPARENT ADDR MUST specify the address it uses to reach the peer (which may be different than the address the peer perceives itself as in the case of NAT or multi-homed peer machines).

The data field of the APPARENT ADDR information element is the same as the linux struct sockaddr_in: two octets for the address family, two octets for the port number, four octets for the IPv4 address, and 8 octets of padding consisting of all bits set to 0. Thus the total length of the APPARENT ADDR information element is 18 octets.

The following diagram demonstrates the generic APPARENT ADDR format:

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x12     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        sockaddr struct        |
   :   for address family in use   :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following diagram demonstrates the APPARENT ADDR format for an IPv4 address:

```
                       1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x12     |      0x10     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            0x0200             | <- Address family (INET)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            0x11d9             | <- Portno (default 4569)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      32-bit IP address        |
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   |      8 octets of all 0s       |
   |   (padding in sockaddr_in)    |
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following diagram demonstrates the APPARENT ADDR format for an IPv6 address:

```
                       1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x12     |      0x1C     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            0x0A00             | <- Address family (INET6)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            0x11d9             | <- Portno (default 4569)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           32 bits             | <- Flow information
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      128-bit IP address       | <- Ip6 Address
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           32 bits             | <- Scope ID
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.18. REFRESH

The purpose of the REFRESH information element is to indicate the number of seconds before an event expires. Its data field is 2 octets long.

The REFRESH information element is used with IAX REGREQ, REGACK, and DPREP messages. When sent with a REGREQ it is a request that the peer maintaining the registration set the timeout to REFRESH seconds. When sent with a DPREP or REGACK, it is informational and tells a remote peer when the local peer will no longer consider the event valid. The REFRESH sent with a DPREP tells a peer how long it SHOULD store the received dialplan response.

If the REFRESH information element is not received with a DPREP, the expiration of the cache data is assumed to be 10 minutes. If the REFRESH information element is not received with a REGACK, registration expiration is assumed to occur after 60 seconds.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x13     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  2 octets specifying refresh  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.19. DPSTATUS

The purpose of the DPSTATUS information element is to indicate the status of a CALLED NUMBER in a remote dialplan. Its data field is a two octet bitmask specifying flags from the table below. Exactly one of the low 3 bits MUST be set, and zero, one, or two of the high two bits MAY be set.

The DPSTATUS information element MUST be sent with IAX DPREP messages, as it is the payload of the dialplan response.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x14     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|                     |N|C|E|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following table lists the dialplan status flags:

FLAG	DESCRIPTION
0x0001	Exists
0x0002	Can exist
0x0004	Non-existent
0x4000	Retain dialtone (ignorepat)
0x8000	More digits may match number

8.4.20. CALLNO

The purpose of the CALLNO information element is to indicate the call number a remote peer needs to use as a destination call number to identify a call being transferred. The peer managing a transfer sends the CALLNO for one transfer endpoint to the other transfer endpoint so that it knows what call number to specify for the transfer. The data field is 2 octets long and specifies a call number in the same manner as a source call number or destination call number is specified in a frame header.

The CALLNO information element MUST be sent with IAX TXREQ, TXREADY, and TXREL messages. Transferring cannot succeed if the CALLNO IE is not included with the appropriate transfer messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x15      |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Callno of transfer recipient |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.21. CAUSE

The purpose of the CAUSE information element is to indicate the reason an event occurred. It carries a description of the CAUSE of the event as UTF-8-encoded data. Notification of the event itself is handled at the message level.

The CAUSE information element SHOULD be sent with IAX HANGUP, REJECT, REGREJ, and TXREJ messages.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x16     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :  UTF-8-encoded CAUSE of event :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.22. IAX UNKNOWN

The purpose of the IAX UNKNOWN information element is to indicate that a received IAX command was unknown or unrecognized. The one octet data field contains the subclass of the received frame that was unrecognized.

The IAX UNKNOWN information element MUST be sent with IAX UNSUPPORT messages.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x17     |      0x01     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Rec'd Subclass|
   +-+-+-+-+-+-+-+-+
```

8.4.23. MSGCOUNT

The purpose of the MSGCOUNT information element is to indicate how many voicemail messages are waiting in a registered user's mailbox. The data field is 2 octets long. If it is set to all 1s, there is at least one message present. Any other value specifies the number of old messages in the high 8 bits and the number of new messages in the low 8 bits.

The IAX MSGCOUNT information element MAY be sent with IAX REGACK messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x18     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Old messages |  New messages |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.24. AUTOANSWER

The purpose of the AUTOANSWER information element is to request that a call be auto-answered upon receipt of a NEW message which includes the AUTOANSWER information element. Note that this is a request and may or may not be granted by the remote peer. There is no data field with this information element, as its presence alone indicates all necessary information.

The AUTOANSWER information element MAY be sent with IAX NEW messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x19     |      0x00     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.25. MUSICONHOLD

The purpose of the MUSICONHOLD information element is to request that music-on-hold be played while a call is in the QUELCH state. The optional data field specifies a music-on-hold class to be used, as UTF-8-encoded data. In the absence of a data field, no music-on-hold class is specified and the IE SHOULD be treated as a generic request for music-on-hold.

The MUSICONHOLD information element MAY be sent with IAX QUELCH messages.

If no MUSICONHOLD information element is received, music-on-hold is not requested.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x1a     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :  Optional Music On Hold Class :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.26. TRANSFERID

The purpose of the TRANSFERID information element is to identify a transfer across all three peers participating in a transfer event. It carries a number, four octets long, that SHOULD be unique for the duration of the transfer process.

The TRANSFERID information element SHOULD be sent with IAX TXREQ and TXCNT messages to aid the peers involved in a transfer in identifying the proper calls. It is not required as long as the transferring peers can positively identify the calls participating in the transfer without the TRANSFERID.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x1b     |      0x04     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       4 octet transfer        |
   |           identifier          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.27. RDNIS

The purpose of the RDNIS information element is to indicate the referring DNIS. It carries UTF-8-encoded data.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x1c     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :      UTF-8-encoded RDNIS      :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.28. DATETIME

The DATETIME information element indicates the time a message is sent. This differs from the header time-stamp because that time-stamp begins at 0 for each call, while the DATETIME is a call-independent value representing the actual real-world time. The data field of a DATETIME information element is four octets long and stores the time as follows: The five least significant bits are seconds, the next six least significant bits are minutes, the next least significant five bits are hours, the next least significant five bits are the day of the month, the next least significant four bits are the month, and the most significant seven bits are the year. The year is offset from 2000, and the month is a 1-based index (i.e., January == 1, February == 2, etc). The timezone of the clock MUST be UTC to avoid confusion between the peers.

The DATETIME information element SHOULD be sent with IAX NEW and REGACK messages. However, it is strictly informational.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x1f     |      0x04     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     year    | month |   day   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  hours  |  minutes  | seconds |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.29. CALLINGPRES

The purpose of the CALLINGPRES information element is to indicate the calling presentation of a caller. The data field is 1 octet long and contains a value from the table below.

The CALLINGPRES information element MUST be sent with IAX NEW messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x26     |      0x01     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Calling Pres. |
+-+-+-+-+-+-+-+-+
```

The following table lists valid calling presentation values:

FLAG	PRESENTATION
0x00	Allowed user/number not screened
0x01	Allowed user/number passed screen
0x02	Allowed user/number failed screen
0x03	Allowed network number
0x20	Prohibited user/number not screened
0x21	Prohibited user/number passed screen
0x22	Prohibited user/number failed screen
0x23	Prohibited network number
0x43	Number not available

8.4.30. CALLINGTON

The purpose of the CALLINGTON information element is to indicate the calling type of number of a caller, according to ITU-T Recommendation Q.931 specifications. The data field is 1 octet long and contains data from the table below.

The CALLINGTON information element MUST be sent with IAX NEW messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x27     |      0x01     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Calling TON  |
+-+-+-+-+-+-+-+-+
```

The following table lists valid calling type of number values from ITU-T Recommendation Q.931:

VALUE	DESCRIPTION
0x00	Unknown
0x10	International Number
0x20	National Number
0x30	Network Specific Number
0x40	Subscriber Number
0x60	Abbreviated Number
0x70	Reserved for extension

8.4.31. CALLINGTNS

The CALLINGTNS information element indicates the calling transit network selected for a call. Values are chosen according to ITU-T Recommendation Q.931 specifications. The data field is two octets long. The first octet stores the network identification plan in the least significant four bits according to the first table below, and the type of network in the next three least significant bits according to the second table below. The second octet stores the actual network identification in UTF-8-encoded data.

The CALLINGTNS information element MUST be sent with IAX NEW messages.

```
                    1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x28     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| | TON | Plan  | UTF-8 Net ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following tables list the valid values for the data field of the 'calling tns' IE.

Q.931 Network Identification Plan Values:

BITS	DESCRIPTION
0000	Unknown
0001	Caller Identification Code
0011	Data Network Identification Code

Q.931 Type of Network Values:

BITS	DESCRIPTION
000	User Specified
010	National Network Identification
011	International Network Identification

8.4.32. SAMPLINGRATE

The purpose of the SAMPLINGRATE information element is to specify to a remote IAX peer the sampling rate in hertz of the audio data being the peer will use when sending data. Its data field is 2 octets long.

If the SAMPLINGRATE information element is not specified, a default sampling rate of 8 kHz may be assumed.

```
                    1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x29     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Sampling Rate in Hertz    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.33. CAUSECODE

The purpose of the CAUSECODE information element is to indicate the reason a call was REJECTed or HANGUPed. It derives from ITU-T Recommendation Q.931. The data field is one octet long and contains

an entry from the table below.

The CAUSECODE information element SHOULD be sent with IAX HANGUP, REJECT, REGREJ, and TXREJ messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x2a     |      0x01     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Cause Code  |
+-+-+-+-+-+-+-+-+
```

NUMBER	CAUSE
1	Unassigned/unallocated number
2	No route to specified transit network
3	No route to destination
6	Channel unacceptable
7	Call awarded and delivered
16	Normal call clearing
17	User busy
18	No user response
19	No answer
21	Call rejected
22	Number changed
27	Destination out of order
28	Invalid number format/incomplete number
29	Facility rejected
30	Response to status enquiry
31	Normal, unspecified

34	No circuit/channel available
38	Network out of order
41	Temporary failure
42	Switch congestion
43	Access information discarded
44	Requested channel not available
45	Pre-empted (causes.h only)
47	Resource unavailable, unspecified (Q.931 only)
50	Facility not subscribed (causes.h only)
52	Outgoing call barred (causes.h only)
54	Incoming call barred (causes.h only)
57	Bearer capability not authorized
58	Bearer capability not available
63	Service or option not available (Q.931 only)
65	Bearer capability not implemented
66	Channel type not implemented
69	Facility not implemented
70	Only restricted digital information bearer capability is available (Q.931 only)
79	Service or option not available (Q.931 only)
81	Invalid call reference
82	Identified channel does not exist (Q.931 only)
83	A suspended call exists, but this call identity does not (Q.931 only)
84	Call identity in use (Q.931 only)

85	No call suspended (Q.931 only)
86	Call has been cleared (Q.931 only)
88	Incompatible destination
91	Invalid transit network selection (Q.931 only)
95	Invalid message, unspecified
96	Mandatory information element missing (Q.931 only)
97	Message type nonexistent/not implemented
98	Message not compatible with call state
99	Information element nonexistent
100	Invalid information element contents
101	Message not compatible with call state
102	Recovery on timer expiration
103	Mandatory information element length error (causes.h only)
111	Protocol error, unspecified
127	Internetworking, unspecified

8.4.34. ENCRYPTION

The purpose of the ENCRYPTION information element is to indicate what encryption methods are accepted for an IAX peer. The data field is a 2 octet bit mask specifying which encryption methods from the table below are accepted.

The ENCRYPTION information element MAY be sent with IAX NEW and AUTHREQ messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x2b     |      0x01     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   |      Encryption Methods       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following table lists valid native encryption methods:

METHOD	DESCRIPTION
0x0001	AES-128

8.4.35. CODEC PREFS

The purpose of the CODEC PREFS information element is to indicate the CODEC preferences of the calling peer. The data field consists of a list of CODECs in the peer's order of preference as UTF-8-encoded data.

The CODEC PREFS information element MAY be sent with IAX NEW messages.

If the CODEC PREFS information element is absent, CODEC negotiation takes place via the CAPABILITY and FORMAT information elements.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x2d     |  Data Length  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |
   :       CODEC Prefs Data        :
   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.36. RR JITTER

The purpose of the Receiver Report (RR) JITTER information element is to indicate the received jitter on a call, per [RFC3550]. The data field is 4 octets long and carries the current measured jitter.

The RR JITTER information element MAY be sent with IAX PONG messages.

```
                        1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      0x2e     |      0x04     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Received Jitter       |
    |                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.37. RR LOSS

The purpose of the RR LOSS information element is to indicate the number of lost frames on a call, per [RFC3550]. The data field is 4 octets long and carries the percentage of frames lost in the first octet, and the count of lost frames in the next 3 octets.

The RR LOSS information element MAY be sent with IAX PONG messages.

```
                        1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      0x2f     |      0x04     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Loss Percent |               |
    +-+-+-+-+-+-+-+-+  Loss Count   |
    |                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.38. RR PKTS

The purpose of the RR PKTS information element is to indicate the total number of frames received on a call, per [RFC3550]. The data field is 4 octets long and carries the count of frames received.

The RR PKTS information element MAY be sent with IAX PONG messages.

```
                        1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      0x30     |      0x04     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Frames Received Count      |
    |                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.39. RR DELAY

The purpose of the RR DELAY information element is to indicate the maximum playout delay for a call, per [RFC3550]. The data field is 2 octets long and specifies the number of milliseconds a frame may be delayed before it MUST be discarded.

The RR DELAY information element MAY be sent with IAX PONG messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x31     |      0x02     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Maximum Playout Delay      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.40. RR DROPPED

The purpose of the RR DROPPED information element is to indicate the total number of dropped frames for a call, per [RFC3550]. The data field is 4 octets long and carries the number of frames dropped.

The RR DROPPED information element MAY be sent with IAX PONG messages.

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0x32     |      0x04     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Total Frames Dropped     |
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.4.41. RR OOO

The purpose of the RR OOO information element is to indicate the number of frames received out of order for a call, per [RFC3550]. The data field is 4 octets long and carries the number of frames received out of order.

The RR OOO information element MAY be sent with IAX PONG messages.

```
                        1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      0x33     |      0x04     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Frames Received       |
   |          Out of Order         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

8.5. Media Formats

Media Format Subclass Values

SUBCLASS	DESCRIPTION	LENGTH CALCULATION
0x00000001	G.723.1	4, 20, and 24 byte frames of 240 samples
0x00000002	GSM Full Rate	33 byte chunks of 160 samples or 65 byte chunks of 320 samples
0x00000004	G.711 mu-law	1 byte per sample
0x00000008	G.711 a-law	1 byte per sample
0x00000010	G.726	
0x00000020	IMA ADPCM	1 byte per 2 samples
0x00000040	16-bit linear little-endian	2 bytes per sample
0x00000080	LPC10	Variable size frame of 172 samples
0x00000100	G.729	20 bytes chunks of 172 samples
0x00000200	Speex	Variable
0x00000400	ILBC	50 bytes per 240 samples
0x00000800	G.726 AAL2	
0x00001000	G.722	16kHz ADPCM
0x00002000	AMR	Variable
0x00010000	JPEG	
0x00020000	PNG	
0x00040000	H.261	
0x00080000	H.263	
0x00100000	H.263p	
0x00200000	H.264	

9. Example Message Flows

This section includes call flow diagrams for some of the various types of IAX calls that can be made. In each diagram, the '=' character represents a full frame and the '-' character represents a mini frame. Notes applicable to a generic call may be presented alongside each diagram.

9.1. Ping/Pong

PING->PONG

```
           Peer A                                  Peer B
            ________________________________________
           |                                        |
      T    |                                        |
      i    |  ===PING============================>  |
      m    |                                        |
      e    |  <============================PONG===  |Has same time-stamp
           |                                        | as received PING.
      |    |  ===ACK=============================>  |Has same time-stamp
      |    |                                        | as received PONG
     \ /   |________________________________________| and original PING
```

9.2. Lagrq/Lagrp

LAGRQ->LAGRP

```
           Peer A                                  Peer B
            ________________________________________
           |                                        |
      T    |                                        |
      i    |  ===LAGRQ===========================>  |
      m    |                                        |
      e    |  <===========================LAGRP===  |Same time-stamp as
           |                                        | received LAGRQ.
      |    |  ===ACK=============================>  |Same time-stamp as
      |    |                                        | received LAGRP and
     \ /   |________________________________________| original LAGRQ.
```

9.3. Registration

Registration of an IAX Peer

```
         Registrant  A                     Registrar B
              ________________________________________
             |                                        |
      T      |  ===REGREQ==========================>  |
      i      |                                        |
      m      |  <==========================REGAUTH===  |
      e      |                                        |
             |  ===REGREQ==========================>  |
      |      |                                        |
      |      |  <===========================REGACK===  |
    \ | /    |                                        |
     \|/     |  ===ACK==============================>  |
             |                                        |
             |________________________________________|
```

9.4. Registration Release

Registration Release

```
         Registrant A                        Registrar B
              ________________________________________
             |                                        |
      T      |  ===REGREL==========================>  |
      i      |                                        |
      m      |  <==========================REGAUTH===  |
      e      |                                        |
             |  ===REGREL==========================>  |
      |      |                                        |
      |      |  <===========================REGACK===  |
    \ | /    |                                        |
     \|/     |  ===ACK==============================>  |
             |                                        |
             |________________________________________|
```

9.5. Call Path Optimization

IAX Transfer

```
           Peer L         Peer C                 Peer R
            ________________________________________
           |                 |                       |
      T    |                 |                       |
           | <== TXREQ =====[*]== TXREQ =========>   |C requests transfer
      i    |                 |                       |
           | ========================== TXCNT  ==>   |L sends to R
      m    |                 |                       |
           | <========================== TXACC  ==== |R replies
      e    |                 |                       |R sends Media
           |                 |                       | to L
      |    |                 |                       |
      |    | = TXREADY ====> |                       |L tells C 'ready'
      |    |                 |                       | C stops media to L
      |    |                 |                       |
      |    | <== TXCNT ===========================   |L sends to R
      |    |                 |                       |
      |    | === TXACC ============================> |R replies
     \ /   |                 |                       |
           |                 | <== TXREADY ======    |R tells C 'ready'
           |                 |                       | C stops media to R
           |                 |                       |
           | <== TXREL =====[*]== TXREL =========>   |C Releases
           |                                         |
           |_________________________________________|
```

9.6. IAX Media Call

Complete end-to-end IAX media exchange

```
           Peer A                            Peer B
            ________________________________________
           |                                         |
           |  ====NEW============================>   |
      T    |  <=========================AUTHREQ===   |If authentication
           |                                         |   specified.
      i    |  ====AUTHREP========================>   |
      m    |  <===========================ACCEPT===   |
      e    |  ====ACK============================>   |
           |                                         |
      |    |  <=============Voice (full frame)===    |
```

```
      |    |  ====ACK===========================>   |
      |    |                                        |
      |    |  <---------Voice miniframe (ring)---   |
      |    |  <---------Voice miniframe (ring)---   |
      |    |                                        |
    \ | /  |  <==========================RINGING===  |
     \|/   |  ====ACK============================>  |
           |                                        |
           |  <---------Voice miniframe (ring)---   |
           |  <---------Voice miniframe (ring)---   |
           |                                        |
           |  <===========================ANSWER===  |
           |  ====ACK============================>  |
           |                                        |
           |  ====Voice (full frame)=============>  |
           |  <==============================ACK===  |
           |                                        |
           |                                        |
           |  <-----------Voice miniframes------->  |  exchange occurs
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <-----------Voice miniframes------->  |
           |                                        |
           |                                        |
           |  ====Voice (full frame)=============>  |  (note 1)
           |  <===ACK============================  |  (note 2)
           |                                        |  (every 65536 ms).
           |  <=============Voice (full frame)====  |  (note 3)
           |  ====ACK============================>  |
           |                                        |
           |                                        |
           |  <-----------Voice miniframes------->  |
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <-----------Voice miniframes------->  |
           |                                        |
           |                                        |
           |  ====HANGUP=========================>  |  Either can hangup
           |  <==============================ACK===  |
           |________________________________________|


   Note 1: Mini Frames carry the low 16 bits of the peer's
           32-bit time-stamp.
   Note 2: Full frames re-sync the 32 bit time-stamp when the 16 bit
           time-stamp overflows.
```

```
   Note 3:Each side has its own 32 bit time-stamp so each side needs
           to sync at 16 bit overflow.
```

9.7. IAX Media Call via an IAX Device

An IAX peer is not required to maintain a complete dialplan. In the event that a user wishes to dial from an IAX peer which does not switch its own calls, the following call flow diagram may represent the transaction:

```
           Peer A (IAX Device)                 Peer B (Dialplan Server)

            ________________________________________
           |                                        |
           |  ====NEW============================>  |
      T    |  <=========================AUTHREQ===  |  If auth specified
      i    |  ====AUTHREP========================>  |
      m    |  <==========================ACCEPT===  |
      e    |  ====ACK============================>  |
           |                                        |
           |  ====DPREQ==========================>  |  (Note 1)
      |    |  <===========================DPREP===  |
      |    |                                        |
      |    |  ====DIAL===========================>  |
      |    |  <========================PROGRESS===  |
      |    |  ====ACK============================>  |
    \ | /  |  <==========================ANSWER===  |
     \|/   |  ====ACK============================>  |
           |                                        |
           |  ====Voice (full frame)=============>  |
           |  <=============================ACK===  |
           |  <=============Voice (full frame)====  |
           |  ====ACK============================>  |
           |                                        |
           |                                        |
           |  <-----------Voice miniframes------->  |  Media exchange
           |  <---               .            --->  |
           |  <---               .            --->  |
           |  <---               .            --->  |
           |  <-----------Voice miniframes------->  |
           |                                        |
           |                                        |
           |  ====Voice (full frame)=============>  |  (note 2)
           |  <===ACK============================   |  (note 3)
           |                                        |  (every 65536 ms).
           |  <=============Voice (full frame)====  |  (Note 4)
           |  ====ACK============================>  |
```

```
           |                                         |
           |                                         |
           |  <-----------Voice miniframes------->  |
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <---               .           --->  |
           |  <-----------Voice miniframes------->  |
           |                                         |
           |                                         |
           |  ====HANGUP========================>  |  Either can hangup
           |  <=============================ACK===  |
           |________________________________________|
```

```
   Note 1: There will be multiple DPREQ/DPREPs per call unless
           dialed number is 1 digit long
   Note 2: Mini Frames carry the low 16 bits of the peer's
           32 bit time-stamp.
   Note 3: Full frames re-sync the 32 bit time-stamp when the 16 bit
           time-stamp overflows
   Note 4: Each side has its own 32 bit time-stamp so each side needs
           to sync at 16 bit overflow.
```

10. Security Considerations

IAX is a binary protocol for setting up point-to-point call legs which includes both media and signaling. As such, it is simpler to secure than other more general purpose VoIP protocols, however, security remains a difficult task and various aspects of the protocol must be examined to identify risks.

IAX registration is an area that requires careful attention. Previous protocol versions supported clear text passwords, this feature has been eliminated. The MD5 and RSA alternatives offer much higher security. Although not specified by the IAX protocol, some implementations limit the number of registrants per account to one. And a subsequent registrant with the same credentials would overwrite the prior and receive the calls destined for that user. Theft of service is trivial once a malicious caller has the ability to authenticate. In addition, since distinct cause codes are returned to erroneous registration attempts, an attacker can distinguish between existent and nonexistent users in a registration system, thus resulting in a possible directory harvest attack.

The IAX protocol protects against message replay by using a challenge response method. The IAX registrar or server challenges each call or registration with an arbitrary MD5 or RSA challenge. The response and subsequent authorization relies upon knowledge of a shared secret. Since the server typically chooses a challenges using a random number-based technique, the challenge set is large, making replay highly unlikely.

Although operation in the following manner is not recommended, the IAX protocol does permit servers to forego the challenge process described above. This open approach is inherently insecure and does nothing to prevent unauthorized or usage.

Call Encryption in IAX starts by utilizing static keys. Once negotiated, the key may be changed for the remainder of the call. Once the initial key is compromised, all subsequent calls are subject to interception. A more secure implementation would update the key frequently and as early as practical during each call.

The IAX protocol is also susceptible to eavesdropping. Call Detail, i.e., who is calling whom, is sent in unencrypted binary whether the call is to be encrypted or not. Without encryption, call content, i.e., audio and video, may be easily intercepted. However, this content is protected if the call is encrypted.

Man in the middle attacks are a threat to IAX if encryption is not used. This form of attack permits message insertion, deletion, and

modification such that a call may be redirected or the audio or video replaced in either or both directions for the complete or any portion of a call. If encryption is used, the call is protected end to end. Note: an initial NEW message in an encrypted call is unencrypted and could be changed; however, this is limited to a denial of service attack because subsequent messages containing the same address information are redelivered in an encrypted form.

Denial of service attacks can take at least two forms in IAX. One is simply overloading the peers with bogus requests. A carefully implemented IAX peer would identify this situation and raise an alarm or take other protective action.

Another form of denial of service (DoS) against an existing call is an engineered attack against an existing call. Injecting media, causing excess processing by inserting out of order packets, and sending commands such as hangup or transfer. These attacks require close monitoring of the binary channel to be successful as the message sequence numbers would need to be synchronized with the protocol exchange.

Of course, providing lower layer security with IPSEC, [RFC4301], would address many of these potential issues.

Unicode, [RFC3629] , and [RFC3454] security considerations also apply.

11. IANA Considerations

This document requests registration of the "iax" URI Scheme. See Section 5. IAX also requires a well-known UDP port to be assigned. The current port in use is 4569.

12. Implementation Notes

The original IAX implementation was in Asterisk, the open-source pbx, but [wikipedia] lists thirteen other publically available implementations at the time of this writing. Some of these implementations used draft versions of this specification. Many others were developed using the Asterisk source code as the only specification. While this approach is definitive, it is very difficult to determine the protocols higher level logic and optimize it for the particular programming language or application environment. Interoperability of these implementations can not be guaranteed.

Aside from the trials and tribulations of reverse engineering the source code to create a new implementation, the key lessons learned involve the use of threads, support of international character sets, security, and improved controls to limit interference during denial of service attacks.

The current Asterisk implementation has a limited number of IAX worker threads and, as a result, its scalability is limited, but it can run on low end machines where threads may not be freely available. Improving the threading model will undoubtedly improve performance.

Internationalization and localization are issues that were not originally addressed by most implementations. It was always on the IAX developers' road map, but never a priority. While creating this document, we formalized support for UTF-8 encoding to better support Internationalization and localization.

With regards to security, many IAX implementations permit clear text authentication. This method is not secure and should not be used.

Recently, some issues have been raised regarding server robustness when under a denial of service attack. IAX servers which support unauthenticated requests can receive the equivalent of a SYN attack. To mitigate the impact of these attacks, various controls to limit the number of unauthenticated calls and the number of calls per user may be added to the implementation. Other approaches, such as transferring the call to another, more protected port or using IP rate limiting when excessive failures are detected, are also suggested.

Lastly, given the open nature of the protocol and implementations, it is very easy to extend. This situation makes Postel's Robustness Principle, "Be conservative in what you do, be liberal in what you accept from others," essential to any successful IAX implementation.

13. Acknowledgments

This work was supported by Internet Foundation Austria. The authors would like to thank Birgit Arkesteijn, Marc Blanchet, Mohamed Boucadair, Steve Kann, Olle Johansson, Alexander Mayrhofer, Tim Panton, and Peter Saint-Andre for their extensive review and technical input. We would also like to thank Christopher DeMarco, Frank Ellermann, Daniel Medeiros, Dimitri E. Prado, Leif Madson, and Tilghman Lesher for their support and suggestions.

14. References

14.1. Normative References

[AES] U.S. Department of Commerce/N.I.S.T., "FIPS-197, Announcing the Advanced Encryption Standard", November 2001.

[E164] ITU-T, "The International Public Telecommunication Number Plan", Recommendation E.164, May 1997.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1851] Karn, P., Metzger, P., and W. Simpson, "The ESP Triple DES Transform", RFC 1851, September 1995.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[RFC4647] Phillips, A. and M. Davis, "Matching of Language Tags", BCP 47, RFC 4647, September 2006.

[html401] Jacobs, I., Hors, A., and D. Raggett, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

14.2. Informative References

[RFC3435] Andreasen, F. and B. Foster, "Media Gateway Control Protocol (MGCP) Version 1.0", RFC 3435, January 2003.

[RFC3525] Groves, C., Pantaleo, M., Anderson, T., and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[RFC3761] Faltstrom, P. and M. Mealling, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 3761, April 2004.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 115, RFC 4395, February 2006.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, December 2006.

[RFC4734] Schulzrinne, H. and T. Taylor, "Definition of Events for Modem, Fax, and Text Telephony Signals", RFC 4734, December 2006.

[wikipedia]
http://en.wikipedia.org/wiki/IAX, "Inter-Asterisk eXchange".

Authors' Addresses

Mark A. Spencer
Digium, Inc.
150 West Park Loop Suite 100
Huntsville, AL 35806
US

Phone: +1 256 428 6000
Email: markster@digium.com
URI: http://www.digium.com/

Brian Capouch
Saint Joseph's College
PO Box 909
Rensselaer, IN 47978
US

Phone: +1 219 866 6114
Email: brianc@saintjoe.edu

Ed Guy (editor)
TruPhone
235 Main Street, STE 253
Madison, NJ 07940
US

Phone: +1 973 437 4519
Email: edguy@emcsw.com
URI: http://www.TruPhone.com/

Frank Miller
Cornfed Systems, Inc.
103 Overhill Road
Baltimore, MD 21210
US

Phone: +1 410 404-8790
Email: fwmiller@cornfed.com
URI: http://www.digium.com/

Kenneth C. Shumard
3818 N Lakegrove Way
Boise, ID 83713
US

Phone: +1 208 724 7801
Email: kshumard@gmail.com