IDA System Interface (ISI)

Last modified: June 20, 2007

Introduction

The IDA System Interface is being developed as the standard for communication between IDA seismic stations, data collection hubs, and TCP/IP enabled digitizers. It uses the IDA Authenticated Communication Protocol (IACP) as the underlying transport. An ISI session consists of establishing an IACP connection with the ISI server and then sending ISI frames and reading server responses as ISI frames. The simplest session would be one where the client connects to the ISI server, issues a state of health, configuration, or disk loop summary report, receives the reply, processes it (e.g., displays it or writes it to disk) and then closes the connection. Waveform sessions are slightly more involved. They require that the client send a series of frames that constitutes the waveform request, then read a series of waveform frames in reply. This document discusses the format of the IACP payloads that are used to conduct all such sessions.

The ISI implementation presently supports state of health, configuration, and waveform transfer between IDA/NRTS computers. Client side source code is available for pickup at

```
ftp://idahub.ucsd.edu/pub/pickup/ISI_Toolkit.tgz
```

This software includes library routines for conducting dialogs with an ISI server and sample programs that demonstrate the use of those routines.

ISI Payload Types

IACP payload identifiers 1000 through 1999 have been assigned for use by ISI. To date, 14 types have been defined:

```
#define ISI_IACP_REQ_SOH      1001 /* state of health request */
#define ISI_IACP_REQ_CNF      1002 /* configuration request */
#define ISI_IACP_REQ_WFDISC   1003 /* wfdisc request */
#define ISI_IACP_REQ_FORMAT   1004 /* format part of data request */
#define ISI_IACP_REQ_COMPRESS 1005 /* compress part of data request */
#define ISI_IACP_REQ_POLICY   1006 /* policy part of data request */
#define ISI_IACP_REQ_TWIND    1007 /* time window part of data request */
#define ISI_IACP_SYSTEM_SOH   1008 /* system SOH data */
#define ISI_IACP_STREAM_SOH   1009 /* stream SOH data */
#define ISI_IACP_WFDISC       1010 /* wfdisc record */
#define ISI_IACP_STREAM_CNF   1011 /* configuration data */
#define ISI_IACP_GENERIC_TS   1012 /* ISI generic time series packet */
#define ISI_IACP_RAW_PKT      1013 /* raw digitizer packet of arbitrary type */
#define ISI_IACP_REQ_SEQNO    1014 /* sequence number part of a data request */
```

ISI Data Types

The discussion of the payload formats that follows will refer to the fundamental data types given in the following table.

Name	Length	Description
INT8	1	signed byte
UINT8	1	unsigned byte
INT16	2	signed 16-bit integer, network byte order
UINT16	2	unsigned 16-bit integer, network byte order
INT32	4	signed 32-bit integer, network byte order
UINT32	4	unsigned 32-bit integer, network byte order
INT64	8	signed 64-bit integer, network byte order
UINT64	8	unsigned 32-bit integer, network byte order
REAL32	4	32-bit IEEE floating point, network byte order (cast as UINT32)
REAL64	8	64-bit IEEE floating point, network byte order (cast as UINT64)
CHAR	1	ASCII character

These fundamental data types are used to build more complicated data types.

ISI_STREAM_NAME

An ISI_STREAM_NAME is a 12 byte data structure with the following format:

Offset	Data Type	Length	Description
0	CHAR	7	station name
7	CHAR	3	channel name
10	CHAR	2	location code

ISI_TIME_STAMP

An ISI_TIME_STAMP is a 10 byte data structure used to specify an epoch time:

Offset	Data Type	Length	Description
0	REAL64	8	Seconds since January 1, 1970
8	UINT16	2	Clock status

ISI_SRATE

An ISI_SRATE is a 4 byte data structure to describe a sample rate.

Offset	Data Type	Length	Description
0	INT16	2	Sample rate factor
2	INT16	2	Sample rate modifier

The sample rate factor and modifier fields follow the same convention as used in the SEED specification. If the sample rate factor is positive, it represents samples per second otherwise it represents seconds per sample. In either case, the value is modified by the sample rate modifier field. If the modifier is positive, the factor is multiplied by the modifier otherwise it is divided by the modifier.

ISI_COORDS

An ISI_COORDS is a 16 byte data structure to specify a location.

Offset	Data Type	Length	Description
0	REAL32	4	latitude, decimal degrees
4	REAL32	4	longitude, decimal degrees
8	REAL32	4	surface elevation, meters
12	REAL32	4	depth of burial, meters

ISI_INST

An ISI_INST is a 23 byte data structure with basic information about a particular data stream.

Offset	Data Type	Length	Description
0	REAL32	4	CSS 3.0 “calib”
4	REAL32	4	CSS 3.0 “calper”
8	REAL32	4	CSS 3.0 “hang”
12	REAL32	4	CSS 3.0 “vang”
16	CHAR	7	CSS 3.0 “instype”

ISI_CD_STATUS

An ISI_CD_STATUS s a 16 byte data structure with CD1.1 channel status (format 1) parameters.

Offset	Data type	Length	Description
0	UNIT8	1	CD1.1 data status
1	UINT8	1	CD1.1 security
2	UINT8	1	CD1.1 miscellaneous status
3	UINT8	1	CD1.1 voltage status
4	REAL64	8	time of last GPS synchronization
12	UINT32	4	clock differential in microseconds

ISI_DATUM_DESC

An ISI_DATUM_DESC is a 4 byte data structure that describes the contents of the data portion of a waveform packet..

Offset	Data type	Length	Description
0	UNIT8	1	Compression 1 = none 2 = IDA first difference 3 = Steim1 4 = Steim2 5 = gzip
1	UINT8	1	Uncompressed data type 1 = INT8 2 = INT16 3 = INT32 4 = INT64 5 = REAL32 6 = REAL64 11 = native IDA rev 8 raw digitizer packet 12 = native IDA rev 10 raw digitizer packet 13 – native IDA rev 6 raw digitizer packet 14 – native IDA rev 7 raw digitizer packet 15 – native IDA rev 9 raw digitizer packet 16 – native IDA rev 5 raw digitizer packet 17 – QDPLUS (wrapped Quanterra Q330 QDP packet) 18 – miniSEED
2	UINT8	1	Byte Order (types 1-6) 0 = little endian byte order 1 = big endian byte order
3	UINT8	1	Uncompressed sample size in bytes

The byte order field is not applicable for those packets whose type is greater than 10. For those packets the compression should be 1 or 5 and the uncompressed sample size should be 1. In other words, they are an opaque collection of bytes as far as the ISI protocols are concerned.

ISI State of Health Request

An ISI state of health report describes the status of the disk loop at the server. This report is obtained by sending a zero length IACP frame with a payload identifier of 1001. The server will reply with a series of frames with IACP payload identifier of 1009. Each type 1009 frame contains state of health information for one data stream. After the final type 1009 frame has been sent, the server will send an IACP NULL frame (payload identifier 0). A type 1009 payload consists of 38 bytes with the following layout:

Offset	Data Type	Length	Description
0	ISI_STREAM_NAME	2	stream name
12	ISI_TIME_STAMP	10	time of oldest datum available on server
22	ISI_TIME_STAMP	10	time of youngest datum available on server
32	REAL64	8	elapsed time since server received data for this stream
40	UINT32	4	number of data segments for this stream
44	UINT32	4	number of data records received by server for this stream

ISI Configuration Request

An ISI configuration report describes the data streams that are available from the server. This report is obtained by sending a zero length IACP frame with payload identifier 1002. The server will reply with a series of frames with IACP payload identifier of 1011. Each type 1011 frame contains configuration information for one data stream. After the final type 1011 frame has been sent, the server will send an IACP NULL frame (payload identifier 0). A type 1011 payload consists of 55 bytes with the following layout:

Offset	Data Type	Length	Description
0	SI_STREAM_NAME	12	stream name
12	ISI_SRATE	4	sample rate
16	ISI_COORDS	16	coordinates
32	ISI_INST	23	instrument parameters

ISI Disk Loop Summary Request

An ISI disk loop summary is a collection of CSS 3.0 wfdiscs that describes the current state of data in the disk loop. This report is obtained by sending a zero length IACP frame with payload identifier 1003. The server will reply with a series of frames with IACP payload identifier of 1010. Each type 1010 frame contains configuration information for one data stream. After the final type 1010 frame has been sent, the server will send an IACP NULL frame (payload identifier 0). A type 1010 payload consists of 283 bytes with the following layout:

Offset	Data Type	Length	Description
0	CHAR	283	CSS 3.0 wfdisc string

ISI Data Requests

Two types of ISI data requests are supported. In one, the client defines the boundaries of the request by specifying start and end times. In the other, the boundaries are specified using sequence numbers. In either case, the waveform request is made by sending a sequence of of frames which specify the details of the request, terminated by an IACP NULL frame.

ISI Waveform (Time Window) Data Request

A time based ISI waveform request is initiated by sending a sequence of three types of frames

- 1004 (ISI_IACP_TYPE_REQ_FORMAT) for specifying the delivery format of the data packets
- 1005 (ISI_IACP_TYPE_REQ_COMPRESS) for specifying the data compression to apply, if any
- 1007 (ISI_IACP_TYPE_REQ_STREAM) to specify the desired time window for a particular data stream or streams. As many instances of the type 1007 frame should be sent as necessary in order to completely specify the list of desired streams.

ISI_IACP_REQ_FORMAT (type 1004)

This is an IACP frame with payload identifier 1004. The payload is a single UINT32.

Offset	Data Type	Length	Description
0	UINT32	4	0 for generic format 1 for native digitizer packet format

Generic format eliminates any digitizer specific encoding while native format is the raw digitizer packet. In either case, the data are returned in the data field of the generic time series packet (type 1012) described later in this document.

ISI_IACP_REQ_COMPRESS (type 1005)

This is an IACP frame with payload identifier 1005. The payload is a single UINT32.

Offset	Data Type	Length	Description
0	UINT32	4	1 for uncompressed 2 for IDA first difference 3 for Steim 1 4 for Steim 2 5 for gzip

This value specifies how the server will compress the waveform data before sending it over the network (at present only uncompressed or IDA first difference and gzip are implemented. If the client desires a native packet format which is already compressed (eg, MiniSEED) then it should specify uncompressed here.

ISI_IACP_REQ_STREAM (type 1007)

This is an IACP frame with payload identifier 1007. The payload is a 28 byte structure with the following layout:

Offset	Data Type	Length	Description
0	ISI_STREAM_NAME	12	stream name, possibly with wild cards
12	REAL64	8	time of first datum
20	REAL64	8	time of final datum

The stream name is the (sta, chan, loc) triple used to name the particular data stream. Limited wildcarding is permitted, as follows. If any of the 3 components of the name is given as an asterisk, the server interprets it to mean all values applicable to that server. For example, a triple ("*", "*", "*") would be interpreted to mean all streams available on the server, ("ABC", "*", "*") would mean all streams from station ABC, while ("*", "BHZ", "00") would mean all BHZ00 streams from all stations.

The time of first datum is the epoch time (seconds since January 1, 1970) for the desired first sample and the time of final datum is the epoch time of the desired final sample. The server will select the waveform packets that contain the desired times, but will not split or otherwise modify the original time span of the packets as they reside on the server. This means that, in practice, the client can expect the first waveform packet to begin slightly before the requested start time and the final packet can extend to slightly beyond the requested time. Note that since epoch time is specified as seconds since January 1, 1970 and it is now past that date, all epoch times will be positive numbers. We take advantage of this fact to define three wildcard times that are the negative values given below.

Value	Interpretation
-2.0	oldest packet available
-3.0	most recent packet available
-4.0	continuous data feed

The values of -2 and -3 can be used to specify either or both of the time boundaries. The -4 value is only valid for specifying the time of final datum. Any waveform request that does not specify -4 as the time of final datum is a segment request. Such a request spans a specific time interval. Any request which specifies -4 as the time of final datum is a continuous data request. In such a case the server will keep the connection alive indefinitely (with IACP heartbeats if necessary). As new data are acquired the server will forward them to the client.

For example a (begin, end) time pair of (-2, -3) would mean that the client wants to see all the data currently available at the server, while (-2, -4) would mean that the client wants all the data currently available on the server and all additional data the server would obtain in the future. Likewise, (-3, -4) would mean the client has no interest in any old data which are available but wants to get any data acquired since the request was made.

ISI Time Based Waveform Protocol

The client initiates a waveform request by sending an IACP NULL terminated series of request frames as described above. The server acknowledges the waveform request by returning the same IACP NULL terminated series of request frames, however it will expand any wild carded ISI_IACP_REQ_STREAM frames into multiple frames with explicit values for the (sta, chan, loc).

After the server has acknowledged the waveform request, it starts sending out generic time series frames (type 1012) that contain the requested data. If the waveform request was a segment request then once the final waveform frame has been sent the server will send an IACP Alert frame (payload identifier 100) with a cause code of 2, indicating that the request has been fully satisfied. Note that this does not mean that all the requested data were actually delivered, it only means that all the data that the server is capable of providing has been delivered.

In the case of a continuous waveform request, the connection will persist indefinitely. A continuous connection will terminate only from I/O errors, internal implementation errors (i.e. bugs in the client or server code) or external intervention (e.g. server reboot, etc).

While it is not uncommon for a continuous connection to persist for months or longer, it will eventually fail from one of these causes. When it does fail, the server will close its connection and destroy any state associated with the connection. It is up to the client to decide to reconnect, and if so, what streams and start times should be requested. The ISI Toolkit referred to at the start of this document includes an API that takes care of the bookkeeping and other client side drudgery associated with reestablishing a failed connection. This allows the high level application to loop around calls to a function which reads the next packet from the server and not concern itself with handling transient I/O errors.

ISI_GENERIC_TS (type 1012)

The waveform data are delivered as ISI_GENERIC_TS frames, which are IACP frames with payload identifier 1012. This is a variable length frame with a fixed length 64- byte header followed by a variable amount of data, as show below.

Offset	Data type	Length	Description
0	ISI_STREAM_NAME	12	stream name
12	ISI_SRATE	4	nominal sample rate
16	ISI_TSTAMP	10	time of first datum
26	ISI_TSTAMP	10	time of last datum
36	ISI_CD_STATUS	16	CD 1.1 channel status
52	UNIT32	4	number of samples
56	UINT3	4	number of bytes, N
60	ISI_DATUM_DESC	4	datum descriptor
64	UINT8	N	data

If the type field of the datum descriptor is greater than 10, then the data field is the entire raw digitizer packet (with its own header and data fields), otherwise it is the waveform data.

ISI Sequence Number Based Packet Request

For strict data communication purposes, requesting data based on time stamps is not always the best means of specifying what data should be sent. For instance, if the clock is bad then it may not even be possible to determine the time of a particular packet. IDA addresses this problem by treating the packets as an opaque collection of bytes, each with a unique sequence number. The client specifies the range of its request using these sequence numbes and, as with the time based requests, we allow special "wildcard" values that allow the user to request the oldest or youngest data available or to maintain a continuous data feed.

A critical requirement is that the data source assigns *unique* numbers to the packets it produces. This sequence number is a "creation sequence number", meaning that it reflects the order the packets were created by the data source. No relationship between a particular data stream and sequence number can be inferred. Furthermore, no relationship between the numerical value of a set of sequence numbers and data continuity should be inferred. This is because the same sequence number generator is used for all data streams. Since packets for the various streams appear at different intervals it will almost always be the case that the sequence numbers for two consecutive packets from the same stream will not increment by one. The only thing that can be inferred from sequence number is that the last sample in a packet with a higher sequence number contains data that is at a time later than (or possibly equal to) that of the last sample in a packet with an earlier sequence number. One may also conclude that a collection of packets with no gaps in the sequence number is complete.

Sequence numbers, once assigned, never change as they move from the digitizer to the station computer, to the regional hub, to the data center. This non-volatile nature of sequence numbers means that it is necessary to ensure that they *never* get reset at the data source. The way IDA have implemented this is by defining a 12 byte sequence number as follows.

ISI_SEQNO

Offset	Data Type	Length	Description
0	UINT32	4	signature of sequence number generator
4	UINT64	8	counter

The signature is simply a number that is guaranteed to be larger than all previous signatures ever used by that particular data source. By convention, we make the signature to be the current UTC time in seconds since 1/1/1970 when the sequence number generator was initialized. By initialized we mean that the 64-bit counter is reset to 0, either because we have never counted before or because we lost the original counter for one of many possible reasons. As long as the system has a way of determining current UTC time with some reasonable accuracy, this will ensure that even replacement systems will never issue sequence numbers that are smaller than what has been issued before at a particular site. This practice is guaranteed to fail in 2106 when the unsigned 32 bit signature wraps around to zero. It also requires that the computer assigning the sequence numbers has a way to determine current time with "reasonable" accuracy at the time the sequence number generator is initialized, but that is not difficult for anyone wearing a watch.

A sequence number request begins with the same type 1004 (ISI_IACP_TYPE_REQ_FORMAT) and 1005 (ISI_IACP_TYPE_REQ_COMPRESS) frames described in the previous section, however instead of type 1007 time window request frames the client should send 1014 (ISI_IACP_TYPE_REQ_SEQNO) frames to specify the sequence number boundaries of the data request. As many instances of the type 1014 frame should be sent as necessary in order to completely specify the list of desired streams, followed by a IACP_NULL frame.

ISI_IACP_REQ_SEQNO (type 1014)

This is an IACP frame with payload identifier 1014. The payload is a 31 byte structure with the following layout:

Offset	Data Type	Length	Description
0	CHAR	7	site name, possibly with wild cards
7	ISI_SEQNO	12	sequence number of first packet
19	ISI_SEQNO	12	sequence number of final packet

The site name is the name associated with the disk loop that contains the sequence numbered packets, which by IDA convention is the name of the station. One may use an asterisk to signify all data available to the server. The beginning and ending sequence numbers can be the absolute numbers refering to specific packets, or they can be one of three "wildcard" numbers. The wildcard sequence numbers are used to refer to the oldest packet currently in the disk loop, the youngest packet currently in the disk loop, and to define a continuous data feed. The wildcards are specified in the signature field. If one of the following signatures is used then the counter field is ignored and the following interpretation is applied:

Value (hex)	Interpretation
FFFFFFFF	oldest packet available
FFFFFFFE	most recent packet available
FFFFFFFD	continuous data feed

ISI Sequence Number Based Packet Protocol

The client initiates a packet request by sending an IACP NULL terminated series of request frames as described above. The server acknowledges the waveform request by returning the same IACP NULL terminated series of request frames, however it will expand any wild carded ISI_IACP_REQ_SEQNO frames into multiple frames with explicit values for the site field.

The protcol allows for ISI_IACP_REQ_COMPRESS (type 1005) frames to be included in the request. Although various compression algorithms can be specified in this frame, only types 1 (no compression) and 5 (gzip compression) are applicable. This is because the server treats these packets as fully opaque. If a client requests a compression other than gzip, then it is ignored and gzip compression is used instead.

After the server has acknowledged the packet request, it starts sending out raw packet frames (type 1013) that contain the requested data. If the packet request was a segment request then once the final packet has been sent the server will send an IACP Alert frame (payload identifier 100) with a cause code of 2, indicating that the request has been fully satisfied. Note that this does not mean that all the requested packets were actually delivered, it only means that all the data that the server is capable of providing has been delivered.

In the case of a continuous request, the connection will persist indefinitely. A continuous connection will terminate only from I/O errors, internal implementation errors (i.e. bugs in the client or server code) or external intervention (e.g. server reboot, etc).

While it is not uncommon for a continuous connection to persist for months or longer, it will eventually fail from one of these causes. When it does fail, the server will close its connection and destroy any state associated with the connection. It is up to the client to decide to reconnect, and if so, what sites and sequence numbers should be requested. The ISI Toolkit referred to at the start of this document includes an API that takes care of the bookkeeping and other client side drudgery associated with reestablishing a failed connection. This allows the high level application to loop around calls to a function which reads the next packet from the server and not concern itself with handling transient I/O errors.

ISI_RAW_PACKET (type 1013)

The ISI_RAW_PACKET frame has a variable structure, with each field tagged with an 8-byte preamble that describes the contents, as follows:

Offset	Data Type	Length	Description
0	UINT32	4	field identifier
4	UINT32	4	field length in bytes

This was done so as to allow extensions to the contents of this packet without affecting any exsiting clients. If a client does not recognize a particular field identifier it can just skip over the indicated number of bytes and continue.

The following field identifiers are currently defined:

```
#define ISI_TAG_EOF        0 /* used to denote the end of the list */
#define ISI_TAG_SITE_NAME  1 /* site name */
#define ISI_TAG_SEQNO      2 /* packet sequence number */
#define ISI_TAG_DATA_DESC  3 /* packet description */
#define ISI_TAG_LEN_USED   4 /* number of bytes in packet, as sent */
#define ISI_TAG_LEN_NATIVE 5 /* number of bytes in packet, after decompression */
#define ISI_TAG_PAYLOAD    6 /* the raw packet, described above */
#define ISI_TAG_RAW_STATUS 7 /* status flag applied to packet by data source */
```

The type 1013 packet consists of a concatenation of sequences of preamble followed by data. The sequence is terminated by a field identifier of 0 (ISI_TAG_EOF). Note that this inidicates the end of the sequence and is not a preamble with identifier 0 (in other words, there is *no* 4-byte field length following the ISI_TAG_EOF). The next section describes the layout of the payloads described by the preambles with one of the indentifiers from above.

ISI_TAG_SITE_NAME (1)

Offset	Data Type	Length	Description
0	CHAR	7	site name

This is the name of the data source. By IDA convention, this is usually the station name, however it need not be. The actual station name would normally be found in the raw digitizer packet that would be included elsewhere in the ISI_RAW_PACKET.

ISI_TAG_SEQNO (2)

Offset	Data Type	Length	Description
0	ISI_SEQNO	31	sequence number that identifies this digitizer packet

ISI_TAG_DATA_DESC (3)

Offset	Data Type	Length	Description
0	ISI_DATUM_DESC	4	digitizer packet description

Note that while this item has the same format as the data description that is in the type 1012 packets, it refers to the digitizer packet and not the waveform data which the packet may or may not contain.

ISI_TAG_LEN_USED (4)

Offset	Data Type	Length	Description
0	UINT32	4	Number of bytes in the payload field of this type 1013 packet (see ISI_TAG_PAYLOAD). This is the number of bytes of the packet as sent over the network. If compression was enabled then this will be less than the original size (which is given as the ISI_TAG_LEN_NATIVE) item, below.

ISI_TAG_LEN_NATIVE (5)

Offset	Data Type	Length	Description
0	UINT32	4	Number of bytes in the digitizer packet after decompression.

ISI_TAG_PAYLOAD (6)

Offset	Data Type	Length	Description
0	UINT8	variable	This is the digitizer packet, possibly compressed (as described by the ISI_TAG_DATA_DESC item). It consists of the number of bytes given by the ISI_TAG_LEN_USED item and decompressions to the number of bytes given by ISI_TAG_NATIVE.

ISI_TAG_STATUS (7)

Offset	Data Type	Length	Description
0	UINT32	4	0 = packet deemed normal by data source 1 = packet deemed suspect by data source