Channel Access Protocol Specification

<core-talk@aps.anl.gov>
version 1.5, 2014-08-27

Table of Contents

1. License

2. Document History

Revision	Date	Author	Section	Modification
1.0	2003-12-12	Klemen Žagar	all	Created.
1.1	2004-01-08	Aleš Pucelj	all	Finalized structure.
	2004-01-10	Matej Šekoranja	all	Review.
1.2	2004-04-19	Aleš Pucelj	all	Draft completed.
1.3	2004-05-31	Aleš Pucelj	all	Matej's comments considered (after Channel Access for Java implementation).
	2004-06-01	Matej Šekoranja	all	Review.
	2004-08-12	Klemen Žagar	all	Released
1.4	2008-02-07	Matej Šekoranja	all	Description of `CA_PROTO_READ` and `CA_PROTO_READ_SYNC` added.
	2008-02-07	Klemen Žagar	all	Released
1.4.1	2014-08-27	Daniel J. Lauk	all	Transformed to AsciiDoc format. Recreated graphics.
1.5	2014-09	Michael Davidsaver	all	Major revision to describe operation semantics

3. Introduction

This document describes the EPICS Channel Access (CA) protocol as it is, and has been, implemented. It is also intended to act as a specification to allow the creation of new client and server implements. The focus is on versions >= 4.11 of the CA protocol, which used by EPICS Base 3.14.0 and later. No changes from protocol versions before 4.8 (EPICS Base 3.13.0) will be included in this document.

For the benefit of those writing new clients and servers RFC 2119:Key words for use in RFCs to Indicate Requirement Levels are used.

4. Concepts

4.1. Process Variables

A Process Variable (PV) is the addressable unit of data accessible through the Channel Access protocol. Each PV has a unique name string and SHOULD be served by a single Channel Access server. Specifically, when searching for a PV, each client MUST NOT receive replies identifying more than one server.

4.2. Virtual Circuit

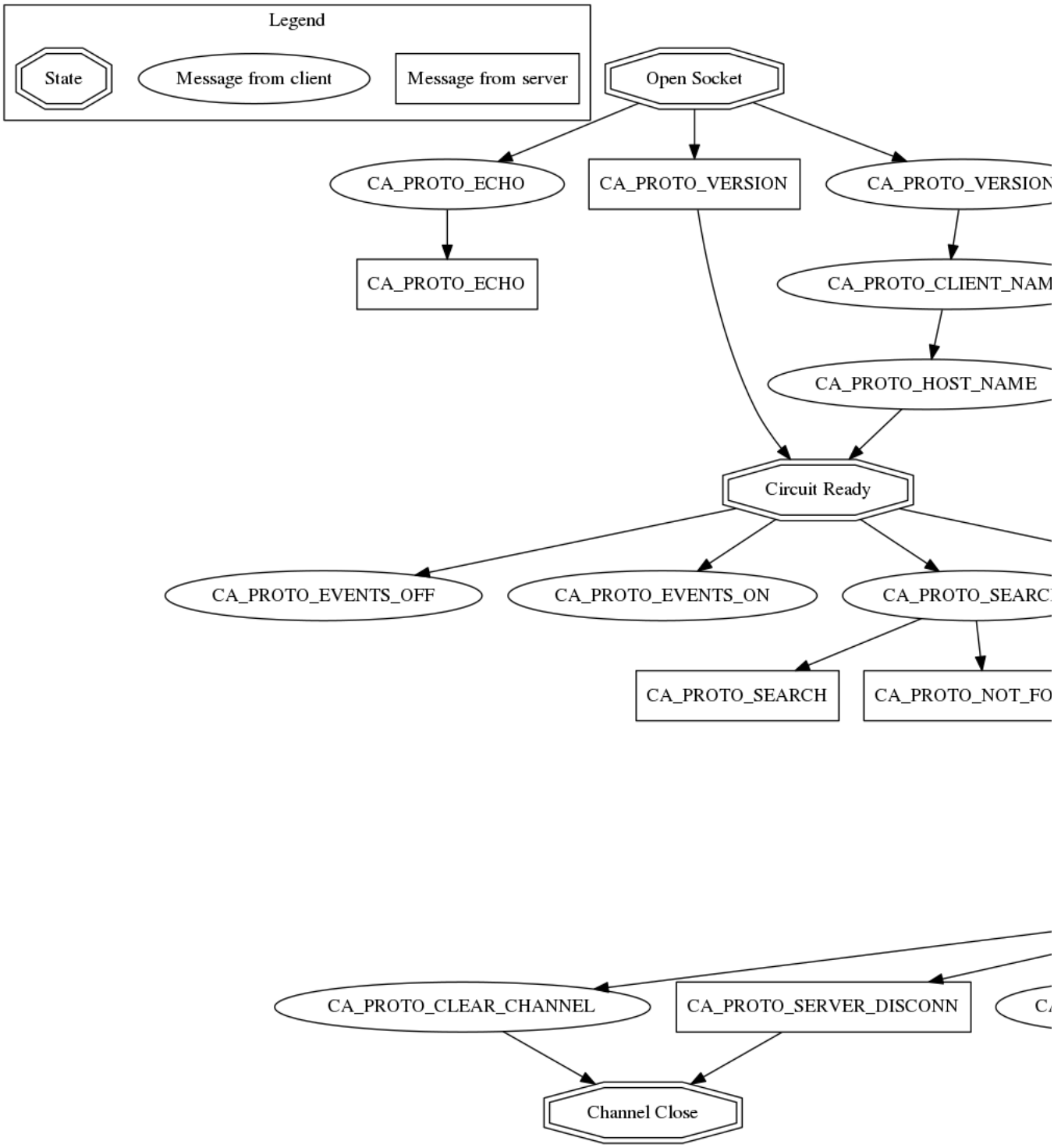
A TCP connection between a CA client and server is referred to as a Virtual Circuit.

Typically only one Circuit is opened between each client and server. However, a client MAY open more than one Circuit to the same server.

4.2.1. TCP Message Flow

The following tree diagram illustrates the order in which normal (not error) CA messages can be sent on a TCP connection. Nodes with box borders are messages sent be the server, and oval borders are messages sent be the client. Nodes with a double border (eg. "Open Socket") are not themselves messages. Instead they indicate pre-conditions which must be meet before certain messages can be sent.

The message CA_ERROR may be sent by a server in response to any client message.



4.3. Channels

A Channel is the association between a particular Circuit and PV name.

At core, a Channel is a runtime allocated pair of integer identifiers (CID and SID) used in place of the PV name to avoid the overhead of string operations. Both client and server MUST maintain a list of the identifiers of all open Channels associated with a Circuit.

The scope of these identifiers is a single Circuit. Identifiers from one Circuit MUST NOT be used on any other. Further more, the same identifier number may be used one two different Circuit in connection with two different PV names.

A Channel's identifiers are explained in section Message Identifiers.

4.4. Monitors

A monitor is created on a channel as a means of registering/subscribing for asynchronous change notifications (publications). Monitors may be filtered to receive only a subset of events (Event Mask), such as value or alarm changes. Several different monitors may be created for each channel.

Clients SHOULD NOT create two monitors on the same channel with the same Event Mask.

4.5. Server Beacons

Server beacons messages (CA_PROTO_RSRV_IS_UP) MUST be periodically broadcast. Beacon messages contain the IP address and TCP port on which the server listens A sequential beacon ID is also included.

When a server becomes active, it MUST immediately begin sending beacons with an increasing delay. An initial beacon interval of 0.02 seconds is RECOMMENDED. After each beacon is sent the interval SHOULD be increased up to a maximum interval. Doubling the interval is RECOMMENDED. The RECOMMENDED maximum interval is 15 seconds.

As a server sends beacons it MUST increment the BeaconID field for each message sent.

CA clients MAY use a server's first beacon as a trigger to re-send previously unanswered CA_PROTO_SEARCH messages.

While it was done historically, clients SHOULD NOT use Beacons to make timeout decisions for TCP Circuits. The CA_PROTO_ECHO message should be used instead.

Clients wishing to detect new servers should maintain a list of all servers along with the last BeaconID received, and the reception time. Servers SHOULD be removed from this list when no Beacon is received for some time (two beacon periods is RECOMMENDED).

4.6. Repeater

See Repeater Operation.

4.7. Timeout Behavior

CA clients typically SHOULD NOT automatically reconnect Circuits which have become unresponsive, instead CA clients SHOULD send a new CA_PROTO_SEARCH request.

CA clients SHOULD on occasion re-send PV name searches which are not answered.

Care must be taken to avoid excessive network load due to repeated lookups and connections. Clients are RECOMMENDED to implement an exponentially increasing (up to a maximum) interval when re-sending CA_PROTO_SEARCH messages for each PV.

Clients are RECOMMENDED to implement a timeout before re-starting a search when a Channel is closed due to an Exception, or Channel creation fails with CA_PROTO_CREATE_CH_FAIL reply.

4.8. Version compatibility

Certain aspects of Channel Access protocol have changed between releases. In this document, Channel Access versions are identified using `CA_VXYY`, where X represents single-digit major version number and YY represents a single- or double-digit minor version number. Stating that a feature is available in `CA_VXYY` implies that any client supporting version XYY must support the feature. Implementation must be backward compatible with all versions up to and including its declared supported minor version number.

Example 1. Channel Access version number

> `CA_V43`, denotes version 4.3 (major version 4, minor version 3).

Channel Access protocol carries an implicit major version of 4. Minor version begin with 1. Minor version 0 is not a valid version.

When a Virtual Circuit is created both client and server send their minor version numbers. The valid messages and semantics of the Circuit are determined by the lower of the two minor versions.

A partial history of CA minor version changes:

EPICS Base	CA Minor	Year	Reason
3.14.12	13	2010	Dynamic array size in monitors
3.14.12	12	2010	PV search over tcp
3.14.0-b2	11	2002	large array?, circuit priority?
3.14.0-b2	10	2002	Beacon counter???
3.14.0-b1	9	2001	Large packet header
3.13.0-b10	8	1997	??
3.13.0-a5	7	1996	Start of CVS history

4.9. Exceptions

Channel Access protocol error messages (CA_PROTO_ERROR) are referred to as Exceptions. Exceptions are sent by a CA server to indicate its failure to process a client message.

An Exception MAY be sent in response to any client message, including those which normally would not result in a reply.

Exception messages carry the header of the client message which triggered the error. It is therefore always possible to associate an Exception with the request which triggered it.

5. Operation

5.1. Overall Server Operation

A CA server will maintain at least two sockets.

A UDP socket bound to the CA port (def. 5064) MUST listen for PV name search request broadcasts. PV name search replies are sent as unicast messages to the source of the broadcast. This socket, or another UDP socket, SHOULD periodically send Beacons to the CA Beacon port (def. 5065).

A TCP socket listening on an arbitrary port. The exact port number is included in PV name search replies. This socket will be used to build Virtual Circuits.

A CA server SHOULD NOT answer PV name search requests for itself unless a CA_PROTO_CREATE_CHAN for that PV from the same client can be expected to succeed. To do otherwise risks excessive load in a tight retry loop.

5.2. Overall Client Operation

A CA client SHOULD maintain a registration with a Repeater on the local system, (re)starting it as necessary.

Clients will send PV name search messages and listen for replies. Typically a client will maintain a table of unanswered name searches and a cache of recent results in order avoid duplicate searches, and to process any replies.

Once an affirmative search reply is received, a Virtual Circuit to the responder is opened if needed. If the client already has a circuit open to this server, it SHOULD be reused. When a Circuit is available, a Channel is created on it, then various get/put/monitor operations are performed on this Channel.

5.3. Name Searching

The process of finding the server which advertises a PV to a particular client can be carried out over UDP, or with >= `CA_V412` over a TCP connection.

In either case each client SHOULD be pre-configured with a set of destinations to send queries. For UDP searching, this is a list of unicast or broadcast endpoints (IP and port). For TCP searching, this is a list of endpoints.

It is RECOMMENDED that a default set of UDP endpoints be populated with the broadcast addresses of all network interfaces except the loopback.

It is RECOMMENDED that, on client startup, Circuits be established to all endpoints in the TCP search list.

Search results are transitory. Subsequent searches MAY yield different results. Therefore queries SHOULD be re-tried unless an active Channel is already open.

5.3.1. UDP search datagrams

Several CA messages MAY be included in one UDP datagram.

A datagram which includes CA_PROTO_SEARCH messages MUST begin with a CA_PROTO_VERSION message.

For efficiency it is RECOMMENDED to include as many search requests as possible in each datagram, subject to datagram size limits.

A CA server MUST NOT send a CA_PROTO_NOT_FOUND in response to a UDP search request.

5.3.2. TCP search

CA_PROTO_SEARCH messages MUST NOT be sent on a Circuit unless a CA_PROTO_VERSION message has been received indicating >= `CA_V412`.

When supported, CA_PROTO_SEARCH messages may be sent at any time the circuit is open.

A CA server MAY send a CA_PROTO_NOT_FOUND in response to a UDP search request if the DO_REPLY bit is set.

Clients MAY ignore CA_PROTO_NOT_FOUND messages.

A CA_PROTO_NOT_FOUND message is not final. A subsequent search might yield a different result.

5.4. Virtual Circuits

5.4.1. Inactivity timeout

When a Circuit is created, both client and server MUST begin a countdown timer. When any traffic (including a CA_PROTO_ECHO message) is received on the Circuit, this counter is reset to its initial value. If the timer reaches zero, the Circuit is closed.

Clients MUST send a CA_PROTO_ECHO message before the countdown reaches zero. It is RECOMMENDED to send an echo message when the countdown reaches half its initial value.

When a CA_PROTO_ECHO message is received by the server, it MUST be immediately copied back to the client.

The RECOMMENDED value for the countdown timer is 30 seconds.

5.4.2. Circuit Setup

When a Circuit is created, both client and server MUST send CA_PROTO_VERSION as their first message. This message SHOULD be sent immediately.

Note for implementers. For EPICS Base before 3.14.12, RSRV did not immediately send a version message due to a buffering problem. Instead the version message was not sent until some other reply forced a flush of the send queue.

In addition the client SHOULD send CA_PROTO_HOST_NAME and CA_PROTO_CLIENT_NAME messages. Once this is done, the Circuit is ready to create channels.

Note that the host and client name messages SHOULD NOT be (re)sent after the first channel is created. If the client or host name strings change, the circuit SHOULD be closed.

If no host or client name messages are received a server MUST consider the client to be anonymous. It is RECOMMENDED that anonymous users not be granted rights for the Put operation.

5.4.3. Channel Creation

Channel creation starts with a CA_PROTO_CREATE_CHAN request from the client. This message includes the PV name string, and a client selected CID.

If the server can not provide the named PV it replies with CA_PROTO_CREATE_CH_FAIL using the same CID. The server MUST NOT remember the CID of failed creation requests as clients MAY re-used them immediately.

If the server can provide the named PV, it replies with CA_PROTO_ACCESS_RIGHTS followed by a CA_PROTO_CREATE_CHAN reply. Further CA_PROTO_ACCESS_RIGHTS messages MAY follow to reflect changes to access permissions.

Note that the CA_PROTO_CREATE_CHAN reply includes the Channel's native DBR datatype and the maximum number of elements which can be retrieved/set by a get, put, or monitor operation. These attributes are fixed for the lifetime of the channel.

The reply also contains the server selected SID identifier. Together with the CID, these two identifier will be used to refer to the Channel in subsequent operations.

The Channel remains active, and the identifiers valid, until a CA_PROTO_CLEAR_CHANNEL request is sent by a client and its reply received, until a CA_PROTO_SERVER_DISCONN message is received by a client, or if the circuit (TCP connection) is closed.

After a server sends a CA_PROTO_CLEAR_CHANNEL reply or a CA_PROTO_SERVER_DISCONN message it MAY reuse the SID immediately.

After a client receives a CA_PROTO_CLEAR_CHANNEL reply or a CA_PROTO_SERVER_DISCONN message it MAY reuse the CID immediately.

Therefore after a client sends a CA_PROTO_CLEAR_CHANNEL request, or a sever sends a CA_PROTO_SERVER_DISCONN request, no further messages (including CA_PROTO_ERROR) should be sent for the closed channel.

5.4.4. Put Operations

A Operation to write data to a Channel begins with a CA_PROTO_WRITE or CA_PROTO_WRITE_NOTIFY request. The difference between the two is that CA_PROTO_WRITE_NOTIFY gives a reply on success, while CA_PROTO_WRITE does not.

The CA_PROTO_WRITE SHOULD be used when it is not important that all Put operations are executed. A server SHOULD make best effort to ensure that, when a burst of CA_PROTO_WRITE requests is received, that the last request is processed (others could be dropped).

A CA_PROTO_WRITE_NOTIFY request indicates that the client intends to wait until the request is fulfilled before continuing. A server MUST reply to all CA_PROTO_WRITE_NOTIFY requests. A server SHOULD make best effort to fully process all CA_PROTO_WRITE_NOTIFY requests.

Both request messages include a SID to determine which Channel is being operated on.

In addition, a client selected IOID is included. This identifier will be included in a CA_PROTO_WRITE_NOTIFY reply, as well as any CA_PROTO_ERROR exception message resulting from a Put request.

5.4.5. Get Operation

The present value of a Channel is queried with a CA_PROTO_READ_NOTIFY request.

A server MUST reply to all CA_PROTO_READ_NOTIFY requests. A server SHOULD make best effort to fully process all CA_PROTO_READ_NOTIFY requests.

CA_PROTO_READ_NOTIFY messages include a SID to determine which Channel is being operated on, as well as a client selected IOID which will be included in the reply.

The IOID MUST be unique on the channel.

5.4.6. Monitor Operation

A Monitor operation is a persistent subscription which is initiated by a CA_PROTO_EVENT_ADD request and terminated with a CA_PROTO_EVENT_CANCEL request.

Both CA_PROTO_EVENT_ADD and CA_PROTO_EVENT_CANCEL messages include a channel SID as well as a client selected SubscriptionID.

The SubscriptionID MUST be unique on the channel.

When a subscription is created a server SHOULD immediately send a CA_PROTO_EVENT_ADD reply with the present value of the Channel if such a value is available.

After a CA_PROTO_EVENT_CANCEL request is received, a server MUST send one final CA_PROTO_EVENT_ADD reply with a zero payload size. Before a CA_PROTO_EVENT_CANCEL request is received, a server MUST NOT send a CA_PROTO_EVENT_ADD reply with a zero payload size.

5.4.7. Errors

Any client message MAY result in an CA_PROTO_ERROR reply from a server.

5.5. Data Count in Gets and Monitors

Prior to `CA_V413`, the element count in a CA_PROTO_EVENT_ADD or CA_PROTO_READ_NOTIFY reply MUST be the same as given in the corresponding CA_PROTO_EVENT_ADD or CA_PROTO_READ_NOTIFY request. A request for zero elements MUST result in

an ECA_BADCOUNT exception. If a server can not provide all of the elements requested, then it fills out the message body with null bytes.

Beginning in CA_V413, a request for zero elements is valid. The element count in a reply is then the number of elements the server could provide (perhaps zero).

The element count in a reply MUST NOT exceed the maximum element count on the channel.

This dynamic array size feature creates a potential ambiguity in the protocol if the number of bytes in a CA_PROTO_EVENT_ADD reply is zero.

Therefore it is RECOMMENDED that clients not create dynamic monitors for the plain DBR_* types. Clients needing to create such monitors are RECOMMENDED to promote the type to the corresponding DBR_STS_* (the extra meta-data can be ignored for internal processing). Then a zero element count has a non-zero body size.

Note to implementers. RSRV will always give at least one element in CA_PROTO_EVENT_ADD replies. libca will silently ignore CA_PROTO_EVENT_ADD replies with zero size before a CA_PROTO_EVENT_CANCEL request is received.

6. Data Types

This section defines all primitive data types employed by CA, as well as their C/C++ equivalents. These data types are referred to in the subsequent sections.

Type Name	C/C++	Description
BYTE	char	Signed 8-bit integer.
UBYTE	unsigned char	Unsigned 8-bit integer.
INT16	short	Signed 16-bit integer.
UINT16	unsigned short	Unsigned 16-bit integer.
INT32	int	Signed 32-bit integer.
UINT32	unsigned int	Unsigned 32-bit integer.
FLOAT	float	IEEE 32-bit float.
DOUBLE	double	IEEE 64-bit float.
STRING[n]	char[]	Array of UBYTE`s. If `[n] is specified, it indicates maximum allowed number of characters in this string including (if neccessary) termination character.
TIMESTAMP	None	Timestamp represented with two UINT32 values. First is number of seconds since 0000 Jan 1, 1990. Second is number of nanoseconds within second

All values are transmitted over the network in big-endian (network) order. For example: UINT32 3145 (0x00000C49) would be sent over the network represented as 00 00 0C 49.

7. Messages

7.1. Message Structure

All Channel Access messages are composed of a **header**, followed by the **payload**.

Header is always present. The command ID and payload size fields have a fixed meaning. Other header fields carry command-specific meaning. If a field is not used within a certain message, its value MUST be zeroed.

Total size of an individual message is limited. With CA versions older than CA_V49, the maximum message size is limited to 16384 (0x4000) bytes. Out of these, header has a fixed size of 16 (0x10) bytes, with the payload having a maximum size of 16368 (0x3ff0) bytes.

Versions CA_V49 and higher may use the **extended message form**, which allows for larger payloads. The extended message form is indicated by the header fields Payload Size and Data Count being set to 0xffff and 0, respectively. Real payload size and data count are then given as UINT32 type values immediately following the header. Maximum message size is limited by 32-bit unsigned integer representation, 4294967295 (0xffffffff). Maximum payload size is limited to 4294967255 (0xfffffffe7).

For compatibility, extended message form should only be used if payload size exceeds the pre-CA_V49 message size limit of 16368 bytes.

7.1.1. Header

Table 1. Standard Message Header

0	1	2	3	4	5	6	7
Command		Payload size		Data type		Data count	

0	1	2	3	4	5	6	7
Parameter 1				Parameter 2			

Table 2. Extended Message Header

0	1	2	3	4	5	6	7
Command		0xFFFF		Data type		0x0000	
Parameter 1				Parameter 2			
Payload size				Data count			

Names of header fields are based on their most common use. Certain messages will use individual fields for purposes other than those described here. These variations are documented for each message individually. All of values in header are unsigned integers.

Generic header fields:

Parameter	Type	Description
Command	UINT16	Identifier of the command this message requests. The meaning of other header fields and the payload depends on the command.
Payload Size	UINT16 or UINT32	Size of the payload (in bytes). MUST not exceed 0x4000 for UDP.
Data Type	UINT16	Identifier of the data type carried in the payload. Data types are defined in section Payload Data Types.
Data Count	UINT16 or UINT32	Number of elements in the payload.
Parameter 1	UINT32	Command dependent parameter.
Parameter 2	UINT32	Command dependent parameter.

7.1.2. Payload

The structure of the payload depends on the type of the message. The size of the payload matches the `Payload Size` header field.

Message payloads MUST be padded to a length which is a multiple of 8 bytes. Zero padding is RECOMMENDED.

7.2. Message Identifiers

Some fields in messages serve as identifiers. These fields serve as identification tokens in within the context of the a circuit (TCP connection). The RECOMMENDED scheme for allocating these values is to create them sequentially starting at 0. All IDs are represented with UINT32.

Overflow of all identifiers MUST be handled! A long running applications might use more than 2**32 of some identifier type type (typically IOID).

7.2.1. CID - Client ID

A CID is the client selected identifier for a channel. A CID MUST be unique for a single Circuit.

Clients MUST not send a request with a CID which is not associated with an active Channel.

Servers MUST ignore any request which does not include the CID of an active channel without closing the Circuit.

A CID is found in the Parameter 1 field of CA_PROTO_ERROR, CA_PROTO_CREATE_CHAN, CA_PROTO_ACCESS_RIGHTS, CA_PROTO_CREATE_CH_FAIL, and CA_PROTO_SERVER_DISCONN messages. And in the Parameter 2 field of CA_PROTO_CLEAR_CHANNEL message.

7.2.2. SID - Server ID

A SID is the server selected identifier for a channel. A SID MUST be unique for a single Circuit.

Servers MUST not send a request with a SID which is not associated with an active Channel.

Clients MUST ignore any request which does not include the SID of an active channel without closing the Circuit.

A SID is found in the Parameter 1 field of CA_PROTO_EVENT_ADD, CA_PROTO_EVENT_CANCEL, CA_PROTO_READ_NOTIFY, CA_PROTO_WRITE_NOTIFY, CA_PROTO_WRITE, CA_PROTO_CLEAR_CHANNEL, and CA_PROTO_CREATE_CHAN (reply only) messages,

7.2.3. Subscription ID

A SubscriptionID is the client selected identifier for a subscription. A CID MUST be unique for a single Circuit.

A SubscriptionID is found in the Parameter 2 field of CA_PROTO_EVENT_ADD and CA_PROTO_EVENT_CANCEL messages.

7.2.4. IOID

An IOID is the client selected identifier for a Get or Put operation. An IOID MUST be unique for a single message type on a single Circuit.

It is possible though NOT RECOMMENDED to use the same IOID concurrently in a CA_PROTO_WRITE, a CA_PROTO_READ_NOTIFY, and a CA_PROTO_WRITE_NOTIFY request.

An IOID is found in the Parameter 2 field of CA_PROTO_READ_NOTIFY, CA_PROTO_WRITE_NOTIFY, and CA_PROTO_WRITE messages.

7.2.5. Search ID

A SearchID is a client selected identifier for a PV name search. A SearchID must be unique for each client endpoint sending requests.

Due to the nature of UDP it is possible for datagrams to be duplicated. Several CA_PROTO_SEARCH messages with the same SearchID MAY be considered to be duplicates, and only one used.

8. Commands (TCP and UDP)

The following commands are sent as either UDP datagrams or TCP messages. Some of the messages are also used within the context of a Virtual Circuit (TCP connection).

8.1. CA_PROTO_VERSION

Command	CA_PROTO_VERSION
ID	0 (0x00)
Description	Exchanges client and server protocol versions and desired circuit priority. MUST be the first message sent, by both client and server, when a new TCP (Virtual Circuit) connection is established. It is also sent as the first message in UDP search messages.

8.1.1. Request

Table 3. Header

Field	Value	Description
Command	0	Command identifier for CA_PROTO_VERSION.
Payload size	0	Must be 0.
Priority	Desired priority	Virtual circuit priority.
Version	Version number	Minor protocol version number. Only used when sent over TCP.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

Table 4. Compatibility

Version	Comment
>= CA_V411	Server will send response immediately after establishing a virtual circuit.
< CA_V411	Message does not include minor version number (it is always 0) and is interpreted as an echo command that carries no data. Version exchange is performed immediately after CA_PROTO_CREATE_CHAN.

Comments

- Priority indicates the server's dispatch scheduling priority which might be implemented by a circuit dedicated thread's scheduling priority in a preemptive scheduled OS.
- Due to a buffering bug, RSRV implementing < CA_V411 did not send CA_PROTO_VERSION immediately on connection, but rather when some other other response triggers a buffer flush.

8.1.2. Response

Table 5. Header

Field	Value	Description
Command	0	Command identifier for CA_PROTO_VERSION.
Reserved	0	Must be 0.
Priority	0	Must be 0.

Field	Value	Description
Version	Version number	Minor protocol version number. Only used when sent over TCP.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

Table 6. Compatibility

Version	Comment
>= CA_V411	Server will not respond to request, but send response immediately after establishing a virtual circuit.
< CA_V411	Message does not include minor version number (it is always 0).

8.2. CA_PROTO_SEARCH

Command CA_PROTO_SEARCH

ID 6 (0x06)

Description Searches for a given channel name. Sent over UDP or TCP.

8.2.1. Request

Table 7. Header

Field	Value	Description
Command	6	Command identifier for CA_PROTO_SEARCH.
Payload Size	>= 0	Padded size of channel name.
Reply	Reply Flag	Search Reply Flag, indicating whether failed search response should be returned.
Version	Version Number	Client minor protocol version number.
SearchID		Client allocated Search identifer.
SearchID		Client allocated Search identifer.

Table 8. Payload

Name	Type	Value	Description
Channel name	STRING		Name of channel to search for.

Comments

- Sent as a UDP datagram.
- It is illegal to specify DO_REPLY flag whenever the message is sending as UDP datagram, regardless of whether broadcast or multicast is used.
- SearchID will be allocated by the client before this message is sent.
- SearchID field value is duplicated.
- Reply flag will be generally DONT_REPLY when searching using broadcast and DO_REPLY when searching using unicast. When DO_REPLY is set, server will send a CA_PROTO_NOT_FOUND message indicating it does not have the requested channel.

8.2.2. Response

Table 9. Header

Field	Value	Description
Command	6	Command identifier for CA_PROTO_SEARCH.
Payload Size	8	Payload size is constant.
Data Type	Port number	TCP Port number of server that responded.
Data Count	0	Must be 0.

Field	Value	Description
SID or IP	0xffffffff	Temporary SID (deprecated) or server IP address.
SearchID		Client allocated Search identifer.

Table 10. Payload

Name	Type	Value	Description
Server protocol version	UINT16		Server protocol version.

Comments

- Received as UDP datagram.
- Search ID field value (CID) is copied from the request.
- Before CA_V411 the SID/IP field will always have the value of 0xffffffff and the server IP address is assumed to be the senders IP.
- Starting with CA_V411 the server's IP address is encoded in the SID/IP field if it differs from the sender's IP, or 0xffffffff if it is the same.
- The port number included in the header is the **TCP** port of the server. Two servers on the same host can share a UDP port number, but not a TCP port number. Therefore, the port the client needs to connect to in that situation may not be the same as expected if this field in the response is not used.

8.3. CA_PROTO_NOT_FOUND

Command CA_PROTO_NOT_FOUND

ID 14 (0x0E)

Description Indicates that a channel with requested name does not exist. Sent in response to CA_PROTO_SEARCH, but only when its DO_REPLY flag was set. Sent over UDP.

8.3.1. Response

Table 11. Header

Field	Value	Description
Command	14	Command identifier for CA_PROTO_NOT_FOUND.
Reserved	0	Must be 0.
Reply Flag	DO_REPLY	Same reply flag as in request: always DO_REPLY.
Version	Same as request	Client minor protocol version number.
SearchID		Client allocated Search identifer.
SearchID		Client allocated Search identifer.

Comments

- Contents of the header are identical to the request.
- SearchID fields are diplicated.
- Original request payload is not returned with the response.

8.4. CA_PROTO_ECHO

Command CA_PROTO_ECHO

ID 23 (0x17)

Description Connection verify used by CA_V43. Sent over TCP.

8.4.1. Request

Table 12. Header

Field	Value	Description
Command	23	Command identifier for CA_PROTO_ECHO.

Field	Value	Description
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

8.4.2. Response

Table 13. Header

Field	Value	Description
Command	23	Command identifier for `CA_PROTO_ECHO`.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

9. Commands (UDP)

The following commands are sent as UDP datagrams.

9.1. `CA_PROTO_RSRV_IS_UP`

Command `CA_PROTO_RSRV_IS_UP`

ID 13 (`0x0D`)

Description Beacon sent by a server when it becomes available. Beacons are also sent out periodically to announce the server is still alive. Another function of beacons is to allow detection of changes in network topology. Sent over UDP.

9.1.1. Response

Table 14. Header

Field	Value	Description
Command	13	Command identifier for `CA_PROTO_RSRV_IS_UP`.
Reserved	0	Must be 0.
Version	Version number	CA protocol version
Server port	>= 0	TCP Port the server is listening on.
BeaconID	Sequential integers	Sequential Beacon ID.
Address	0 or IP	May contain IP address of the server.

Comments

- IP field may contain IP of the server. If IP is not present (field Address value is 0), then IP may be substituted by the receiver of the packet (usually repeater) if it is capable of identifying where this packet came from. Any non-zero address must be interpreted as server's IP address.
- BeaconIDs are useful in detecting network topology changes. In certain cases, same packet may be routed using two different routes, causing problems with datagrams. If multiple beacons are received from the same server with same BeaconID, multiple routes are the cause.
- If a server is restarted, it will most likely start sending BeaconID values from beggining (0). Such situation must be anticipated.

9.2. `CA_REPEATER_CONFIRM`

Command `CA_REPEATER_CONFIRM`

ID 17 (`0x11`)

Description Confirms successful client registration with repeater. Sent over UDP.

9.2.1. Response

Table 15. Header

Field	Value	Description
Command	17	Command identifier for CA_REPEATER_CONFIRM.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Repeater address	IP address	Address with which the registration succeeded.

Comments

- Since repeater can bind to different local address, its IP is reported in Repeater address. This address will be either 0.0.0.0 or 127.0.0.1.

9.3. CA_REPEATER_REGISTER

Command CA_REPEATER_REGISTER

ID 24 (0x18)

Description Requests registration with the repeater. Repeater will confirm successful registration using CA_REPEATER_CONFIRM. Sent over TCP.

9.3.1. Request

Table 16. Header

Field	Value	Description
Command	CA_REPEATER_REGISTER	Command identifier
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0
Client IP address	IP address	IP address on which the client is listening

10. Commands (TCP)

The following commands are used within the context of Virtual Circuit and are sent using TCP.

10.1. CA_PROTO_EVENT_ADD

Command CA_PROTO_EVENT_ADD

ID 1 (0x01)

Description Creates a subscription on a channel, allowing the client to be notified of changes in value. A request will produce at least one response. Sent over TCP.

10.1.1. Request

Table 17. Header

Field	Value	Description
Command	1	Command identifier for CA_PROTO_EVENT_ADD
Payload Size	16	Payload size is constant

Field	Value	Description
Data Type		Desired DBR type of the return value.
Data Count	>= 0	Desired number of elements
SID	SID of the channel.	SID of the channel on which to register this subscription. See SID - Server ID.
SubscriptionID	Client provided Subscription ID	Subscription ID identifying this subscription.See Subscription ID.

Payload

Name	Type	Value	Description
Low val	FLOAT32	0.0	Low value
High val	FLOAT32	0.0	High value
To val	FLOAT32	0.0	To value
Mask	UINT16	Monitor mask	Mask indicating which events to report

Comments

- All payload fields except Mask are initialized to 0 and are present only for backward compatibility.
- Successful subscription will result in an immediate response with the current value. Additional responses will be sent as the change occurs based on the Mask parameter.
- Mask defines a filter on which events will be sent.
- A subscription should be destroyed when no longer needed to reduce load on server. See CA_PROTO_EVENT_CANCEL.

10.1.2. Response

Table 18. Header

Field	Value	Description
Command	1	Command identifier for CA_PROTO_EVENT_ADD
Payload Size	>= 0	Size of the response.
Data Type	same as request	Payload data type.
Data Count	same as request	Payload data count.
Status code	One of ECA codes	Status code (ECA_NORMAL on success).
SubscriptionID	same as request	Subscription ID

Table 19. Payload

Name	Type	Value	Description
Values	DBR		Value stored as DBR type specified in Data Type field. See Payload Data Types.

Comments

- Response data type and count match that of the request.
- To confirm successful subscription, first response will be sent immediately. Additional responses will be sent as the change occurs based on mask parameters.

10.2. CA_PROTO_EVENT_CANCEL

Command	CA_PROTO_EVENT_CANCEL
ID	2 (0x02)
Description	Clears event subscription. This message will stop event updates for specified channel. Sent over TCP.

10.2.1. Request

Table 20. Header

Field	Value	Description
Command	2	Command identifier for CA_PROTO_EVENT_CANCEL.
Payload Size	0	Must be 0.
Data Type		Same value as in corresponding CA_PROTO_EVENT_ADD.
Data Count	>= 0	Same value as in corresponding CA_PROTO_EVENT_ADD.
SID	SID of channel	Same value as in corresponding CA_PROTO_EVENT_ADD.
SubscriptionID	Subscription ID	Same value as in corresponding CA_PROTO_EVENT_ADD.

Comments

- Both SID and SubscriptionID are used to identify which subscription on which monitor to destroy.
- Actual data type and count values are not important, but should be the same as used with corresponding CA_PROTO_EVENT_ADD.

10.2.2. Response

Table 21. Header

Field	Value	Description
Command	1	Command identifier for CA_PROTO_EVENT_ADD.
Payload Size	0	Must be 0.
Data Type	Same as request.	Same value as CA_PROTO_EVENT_ADD request.
Data Count	0	Must be 0.
SID	Same as request.	Same value as CA_PROTO_EVENT_ADD request.
SubscriptionID	Same as request.	Same value as CA_PROTO_EVENT_ADD request.

Comments

- Notice that the response has CA_PROTO_EVENT_ADD command identifier!
- Regardless of data type and count, this response has no payload.

10.3. CA_PROTO_READ

Command	CA_PROTO_READ
ID	3 (0x03)
Description	Read value of a channel. Sent over TCP.

Deprecated since protocol version 3.13.

10.3.1. Request

Table 22. Header

Field	Value	Description
Command	3	Command identifier for CA_PROTO_READ_NOTIFY.
Payload Size	0	Must be 0.
Data Type	DBR type	Desired type of the return value.
Data Count	>= 0	Desired number of elements to read.
SID	Channel SID	SID of the channel to read.
IOID	Client provided IOID	IOID of this operation.

Comments

- Channel from which to read is identified using SID.
- Response will contain the same IOID as the request, making it possible to distinguish multiple responses.

10.3.2. Response

Table 23. Header

Field	Value	Description
Command	3	Command identifier for `CA_PROTO_READ_NOTIFY`.
Payload size	Size of payload	Size of DBR formatted data in payload.
Data type	DBR type	Payload format.
Data count	>= 0	Payload element count.
SID	Same as request	SID of the channel.
IOID	Same as request	IOID of this operation.

Table 24. Payload

Name	Type	Value	Description
DBR formatted data	DBR	DBR formatted data	Value stored as DBR type specified in Data type field. Data count specifies number of elements of DBR value field.

10.4. `CA_PROTO_WRITE`

Command `CA_PROTO_WRITE`

ID 4 (`0x04`)

Description Writes new channel value. Sent over TCP.

10.4.1. Request

Table 25. Header

Field	Value	Description
Command	`CA_PROTO_WRITE`	Command identifier
Payload size	Size of DBR formatted payload	Size of padded payload
Data type	DBR type	Format of payload
Data count	`ELEMENT_COUNT`	Number of elements in payload
SID	SID provided by server	Server channel ID
IOID	Client provided IOID	Request ID

Table 26. Payload

Name	Type	Value	Description
DBR formatted data	DBR	DBR formatted data	Value stored as DBR type specified in Data type field. Data count specifies number of elements of DBR value field.

Comments

- There is no response to this command.

10.5. `CA_PROTO_SNAPSHOT`

Command `CA_PROTO_SNAPSHOT`

ID 5 (`0x05`)

Description Obsolete.

10.6. CA_PROTO_BUILD

Command `CA_PROTO_BUILD`

ID 7 (`0x07`)

Description Obsolete.

10.7. CA_PROTO_EVENTS_OFF

Command `CA_PROTO_EVENTS_OFF`

ID 8 (`0x08`)

Description Disables a server from sending any subscription updates over this virtual circuit. Sent over TCP. This mechanism is used by clients with slow CPU to prevent congestion when they are unable to handle all updates received. Effective automated handling of flow control is beyond the scope of this document.

10.7.1. Request

Table 27. Header

Field	Value	Description
Command	8	Command identifier for `CA_PROTO_EVENTS_OFF`
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

Comments

- This request will disable sending of subscription updates on the server to which it is sent.
- Command applies to a single virtual circuit, so having multiple priority virtual circuit connections to the server would only affect the one on which the message is sent.
- No response will be sent for this request.

10.8. CA_PROTO_EVENTS_ON

Command `CA_PROTO_EVENTS_ON`

ID 9 (`0x09`)

Description Enables the server to resume sending subscription updates for this virtual circuit. Sent over TCP. This mechanism is used by clients with slow CPU to prevent congestion when they are unable to handle all updates received. Effective automated handling of flow control is beyond the scope of this document.

10.8.1. Request

Table 28. Header

Field	Value	Description
Command	9	Command identifier for `CA_PROTO_EVENTS_ON`
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

Comments

- This request will enable sending of subscription updates on the server to which it is sent.

- Command applies to a single virtual circuit, so having multiple priority virtual circuit connections to the server would only affect the one on which the message is sent.
- No response will be sent for this request.

10.9. CA_PROTO_READ_SYNC

Command `CA_PROTO_READ_SYNC`

ID 10 (`0x0A`)

Description **Deprecated since protocol version 3.13.**

10.9.1. Request

Table 29. Header

Field	Value	Description
Command	10	Command identifier for `CA_PROTO_READ_SYNC`.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

10.10. CA_PROTO_ERROR

Command `CA_PROTO_ERROR`

ID 11 (`0x0B`)

Description Sends error message and code. This message is only sent from server to client in response to any request that fails and does not include error code in response. This applies to all asynchronous commands. Error message will contain a copy of original request and textual description of the error. Sent over UDP.

10.10.1. Response

Table 30. Header

Field	Value	Description
Command	11	Command identifier for `CA_PROTO_ERROR`
Payload Size		Size of the request header that triggered the error plus size of the error message.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
CID	Channel CID	CID of the channel for which request failed.
Status Code	One of ECA codes	Error status code.

Table 31. Payload

Name	Type	Value	Description
Original Request	Message Header		Header of the request that caused the error.
Error Message	`STRING`		A null-terminated string conveying the error message.

Comments

- Complete exception report is returned. This includes error message code, CID of channel on which the request failed, original request and string description of the message.
- CID value depends on original request and may not actually identify a channel.
- First part of payload is original request header with the same structure as sent. Any payload that was part of this request is not included. Textual error message starts immediately after the header.

10.11. CA_PROTO_CLEAR_CHANNEL

Command	CA_PROTO_CLEAR_CHANNEL
ID	12 (0x0C)
Description	Clears a channel. This command will cause server to release the associated channel resources and no longer accept any requests for this SID/CID.

10.11.1. Request

Table 32. Header

Field	Value	Description
Command	12	Command identifier of CA_PROTO_CLEAR_COMMAND
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
SID	SID of the channel	SID of channel to clear.
CID	CID of the channel	CID of channel to clear.

10.11.2. Response

Table 33. Header

Field	Value	Description
Command	12	Command identifier of CA_PROTO_CLEAR_COMMAND
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
SID	Same as request	SID of cleared channel.
CID	Same as request	CID of cleared channel.

Comments

- Server responds immediately and only then releases channel resources.
- Once a channel with a given SID has been cleared, any request sent with this SID will fail.
- Sent over TCP.

10.12. CA_PROTO_READ_NOTIFY

Command	CA_PROTO_READ_NOTIFY
ID	15 (0x0F)
Description	Read value of a channel. Sent over TCP.

10.12.1. Request

Table 34. Header

Field	Value	Description
Command	15	Command identifier for CA_PROTO_READ_NOTIFY.
Payload Size	0	Must be 0.
Data Type	DBR type	Desired type of the return value.
Data Count	>= 0	Desired number of elements to read.
SID	Channel SID	SID of the channel to read.
IOID	Client provided IOID	IOID of this operation.

Field	Value	Description

Comments

- Channel from which to read is identified using SID.
- Response will contain the same IOID as the request, making it possible to distinguish multiple responses.

10.12.2. Response

Table 35. Header

Field	Value	Description
Command	15	Command identifier for `CA_PROTO_READ_NOTIFY`.
Payload size	Size of payload	Size of DBR formatted data in payload.
Data type	DBR type	Payload format.
Data count	>= 0	Payload element count.
SID	Same as request	SID of the channel.
IOID	Same as request	IOID of this operation.

Table 36. Payload

Name	Type	Value	Description
DBR formatted data	DBR	DBR formatted data	Value stored as DBR type specified in Data type field. Data count specifies number of elements of DBR value field.

10.13. `CA_PROTO_READ_BUILD`

Command `CA_PROTO_READ_BUILD`

ID 16 (`0x10`)

Description Obsolete

10.13.1. Request

10.14. `CA_PROTO_CREATE_CHAN`

Command `CA_PROTO_CREATE_CHAN`

ID 18 (`0x12`)

Description Requests creation of channel. Server will allocate required resources and return initialized SID. Sent over TCP.

10.14.1. Request

Table 37. Header

Field	Value	Description
Command	18	Command identifier for `CA_PROTO_CREATE_CHAN`
Payload size	Size of payload	Padded length of channel name.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
CID	Channel CID	CID of the channel to create.
Client version	Version number	Client minor protocol version.

Payload

|[options="header"]

Name	Type	Value	Description
Channel name	`STRING`		Name of channel to create.

Comments

- CID sent should be the same as used with `CA_PROTO_SEARCH`.

10.14.2. Response

Table 38. Header

Field	Value	Description
Command	`CA_PROTO_CREATE_CHAN`	
Payload size	0	Must be 0
Data type	DBR type	Native channel data type
Data count	>= 0	Native channel data count
CID	Same as request	Channel client ID
SID	SID provided by server	Channel server ID

Comments

- SID will be associated with CID on the server and will be reused sending certain commands that require it as a parameter.
- SID will be valid until the channel is cleared using `CA_PROTO_CLEAR` or server destroys the PV the channel references.

10.15. `CA_PROTO_WRITE_NOTIFY`

Command	`CA_PROTO_WRITE_NOTIFY`
ID	19 (`0x13`)
Description	Writes new channel value. Sent over TCP.

10.15.1. Request

Table 39. Header

Field	Value	Description
Command	`CA_PROTO_WRITE_NOTIFY`	Command identifier
Payload size	Size of DBR formatted payload	Size of padded payload
Data type	DBR type	Format of payload
Data count	`ELEMENT_COUNT`	Number of elements in payload
SID	SID provided by server	Server channel ID
IOID	Client provided IOID	Request ID

Table 40. Payload

Name	Type	Value	Description
DBR formatted data	DBR	DBR formatted data	Value stored as DBR type specified in Data type field. Data count specifies number of elements of DBR value field.

10.15.2. Response

Table 41. Header

Field	Value	Description
Command	`CA_PROTO_WRITE_NOTIFY`	Command identifier
Payload size	0	Must be 0
Data type	Same as request	Format of data written
Data count	Same as request	Number of elements written
Status	Status code	Status of write success
IOID	Same as request	Request ID

10.16. CA_PROTO_CLIENT_NAME

Command	CA_PROTO_CLIENT_NAME
ID	20 (0x14)
Description	Sends local username to virtual circuit peer. This name identifies the user and affects access rights.

10.16.1. Request

Table 42. Header

Field	Value	Description
Command	CA_PROTO_CLIENT_NAME	Command identifier
Payload size	>=0	Length of string in payload
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0

Table 43. Payload

Name	Type	Value	Description
User name	STRING		0-terminated username string

Comments

- This is a one-way message and will not receive response.
- String in payload must be 0 padded to a length that is multiple of 8.
- Sent over TCP.

10.17. CA_PROTO_HOST_NAME

Command	CA_PROTO_HOST_NAME
ID	21 (0x15)
Description	Sends local host name to virtual circuit peer. This name will affect access rights. Sent over TCP.

10.17.1. Request

Table 44. Header

Field	Value	Description
Command	21	Command identifier for CA_PROTO_HOST_NAME.
Payload size	Size of payload	Length of host name string.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.

Table 45. Payload

Name	Type	Value	Description
Host name	STRING		Client host name.

Comments

- This is one-way message and will receive no response.

10.18. CA_PROTO_ACCESS_RIGHTS

Command CA_PROTO_ACCESS_RIGHTS

ID 22 (0x16)

Description Notifies of access rights for a channel. This value is determined based on host and client name and may change during runtime. Client cannot change access rights nor can it explicitly query its value, so last received value must be stored.

10.18.1. Response

Table 46. Header

Field	Value	Description
Command	22	Command identifier for CA_PROTO_ACCESS_RIGHTS.
Payload size	0	Must be 0.
Reserved	0	Must be 0.
Reserved	0	Must be 0.
CID	Channel CID	Channel affected by change.
Access Rights	Access Rights	Access rights for given channel.

Comments

- Access Rights affect CA_PROTO_READ_NOTIFY, CA_PROTO_WRITE_NOTIFY and CA_PROTO_WRITE.
- CA_PROTO_ACCESS_RIGHTS will be sent immediately after a channel is created using CA_PROTO_CREATE_CHAN. If they change during runtime, this message sent to report new value.
- Changes are only sent to currently connected channels, since it requires valid CID.
- Sent over TCP.

10.19. CA_PROTO_SIGNAL

Command CA_PROTO_SIGNAL

ID 25 (0x19)

Description Obsolete.

10.20. CA_PROTO_CREATE_CH_FAIL

Command CA_PROTO_CREATE_CH_FAIL

ID 26 (0x1A)

Description Reports that channel creation failed. This response is sent to when channel creation in CA_PROTO_CREATE_CHAN fails.

10.20.1. Response

Table 47. Header

Field	Value	Description
Command	CA_PROTO_CREATE_CH_FAIL	Command identifier
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0
CID	Same as request	Client channel ID
Reserved	0	Must be 0

Comments

- Sent over TCP.

10.21. CA_PROTO_SERVER_DISCONN

Command CA_PROTO_SERVER_DISCONN

ID 27 (0x1B)

Description Notifies the client that server has disconnected the channel. This may be since the channel has been destroyed on server. Sent over TCP.

10.21.1. Response

Table 48. Header

Field	Value	Description
Command	CA_PROTO_SERVER_DISCONN	Command identifier
Reserved	0	Must be 0
Reserved	0	Must be 0
Reserved	0	Must be 0
CID	CID provided by client	CID that was provided during CA_PROTO_CREATE_CHAN
Reserved	0	Must be 0

11. Payload Data Types

Channel access defines special structures to transferring data. These types are organized in typed hierarchies with loose inheritance. There are six basic data types: DBR_STRING, DBR_SHORT, DBR_FLOAT, DBR_ENUM, DBR_CHAR, DBR_LONG and DBR_DOUBLE. The type DBR_INT is present as an alias for DBR_SHORT. Each of these types can represent an array of elements.

In addition to element values, some DBR types include meta-data. These types are status (DBR_STS_*), time stamp (DBR_TIME_*), graphic (DBR_GR_*) and control (DBR_CTRL_*). All these structures contain value as the last field.

All DBR data MUST be zero padded to ensure that message body length is a multiple of 8 bytes. Therefore, when receiving a message, it is necessary to use the DBR type and element count to determine the number of body bytes to use. Additional body bytes MUST be ignored.

In addition to zero padding at the end of the message, some padding is placed between the meta-data and the value array.

The following table lists the identifier, meta-data size, padding between meta-data and value, and value element sizes of each DBR type.

Table 49. DBRs

Name	ID	Meta size	padding	Element size
DBR_STRING	0	0	0	40
DBR_INT	1	0	0	2
DBR_SHORT	1	0	0	2
DBR_FLOAT	2	0	0	4
DBR_ENUM	3	0	0	2
DBR_CHAR	4	0	0	1
DBR_LONG	5	0	0	4
DBR_DOUBLE	6	0	0	8
DBR_STS_STRING	7	4	0	40
DBR_STS_INT	8	4	0	2
DBR_STS_SHORT	8	4	0	2
DBR_STS_FLOAT	9	4	0	4
DBR_STS_ENUM	10	4	0	2
DBR_STS_CHAR	11	4	1	1
DBR_STS_LONG	12	4	0	4
DBR_STS_DOUBLE	13	4	4	8
DBR_TIME_STRING	14	12	0	40
DBR_TIME_INT	15	12	2	2
DBR_TIME_SHORT	15	12	2	2
DBR_TIME_FLOAT	16	12	0	4

Name	ID	Meta size	padding	Element size
DBR_TIME_ENUM	17	12	2	2
DBR_TIME_CHAR	18	12	3	1
DBR_TIME_LONG	19	12	0	4
DBR_TIME_DOUBLE	20	12	4	8
DBR_GR_STRING	21	4	0	40
DBR_GR_INT	22	GR_INT	0	2
DBR_GR_SHORT	22	GR_INT	0	2
DBR_GR_FLOAT	23	GR_REAL	2	4
DBR_GR_ENUM	24	GR_ENUM	0	2
DBR_GR_CHAR	25	GR_INT	1	1
DBR_GR_LONG	26	GR_INT	0	4
DBR_GR_DOUBLE	27	GR_REAL	0	8
DBR_CTRL_STRING	28	4	0	40
DBR_CTRL_INT	29	CTRL_INT	0	2
DBR_CTRL_SHORT	29	CTRL_INT	0	2
DBR_CTRL_FLOAT	30	CTRL_REAL	0	2
DBR_CTRL_ENUM	31	GR_ENUM	0	2
DBR_CTRL_CHAR	32	CTRL_INT	1	1
DBR_CTRL_LONG	33	CTRL_INT	0	4
DBR_CTRL_DOUBLE	34	CTRL_REAL	0	8
DBR_PUT_ACKT	35	?	?	2
DBR_PUT_ACKS	36	?	?	2
DBR_STSACK_STRING	37	?	?	40
DBR_CLASS_NAME	38	?	?	40

11.1. DBR_STS_* meta-data

Alarm meta-data. Length: 4 bytes

```
struct metaSTS {
    epicsInt16 status;
    epicsInt16 severity;
};
```

11.2. DBR_TIME_* meta-data

Alarm and time stamp meta-data. Length: 12 bytes

```
struct metaTIME {
    epicsInt16 status;
    epicsInt16 severity;
    epicsInt32 secondsSinceEpoch;
    epicsUInt32 nanoSeconds;
};
```

Note that the EPICS Epoch is 1990-01-01T00:00:00Z. This is 631152000 seconds after the POSIX Epoch of 1970-01-01T00:00:00Z.

11.3. DBR_GR_SHORT meta-data

Alarm and integer display meta-data (no timestamp). Length: ?? bytes

```
struct metaGR_INT {
    epicsInt16 status;
    epicsInt16 severity;
    char units[8];
    epicsInt16 upper_display_limit;
    epicsInt16 lower_display_limit;
```

```
    epicsInt16 upper_alarm_limit;
    epicsInt16 upper_warning_limit;
    epicsInt16 lower_warning_limit;
    epicsInt16 lower_alarm_limit;
};
```

11.4. DBR_GR_CHAR meta-data

Alarm and integer display meta-data (no timestamp). Length: ?? bytes

```
struct metaGR_INT {
    epicsInt16 status;
    epicsInt16 severity;
    char units[8];
    epicsInt8 upper_display_limit;
    epicsInt8 lower_display_limit;
    epicsInt8 upper_alarm_limit;
    epicsInt8 upper_warning_limit;
    epicsInt8 lower_warning_limit;
    epicsInt8 lower_alarm_limit;
};
```

11.5. DBR_GR_FLOAT meta-data

Alarm and floating point display meta-data (no timestamp). Length: ?? bytes

```
struct metaGR_FLOAT {
    epicsInt16 status;
    epicsInt16 severity;
    epicsInt16 precision;
    epicsInt16 padding;
    char units[8];
    epicsFloat32 upper_display_limit;
    epicsFloat32 lower_display_limit;
    epicsFloat32 upper_alarm_limit;
    epicsFloat32 upper_warning_limit;
    epicsFloat32 lower_warning_limit;
    epicsFloat32 lower_alarm_limit;
};
```

11.6. DBR_GR_DOUBLE meta-data

Alarm and floating point display meta-data (no timestamp). Length: ?? bytes

```
struct metaGR_FLOAT {
    epicsInt16 status;
    epicsInt16 severity;
    epicsInt16 precision;
    epicsInt16 padding;
    char units[8];
    epicsFloat64 upper_display_limit;
    epicsFloat64 lower_display_limit;
    epicsFloat64 upper_alarm_limit;
    epicsFloat64 upper_warning_limit;
    epicsFloat64 lower_warning_limit;
    epicsFloat64 lower_alarm_limit;
};
```

11.7. GR_ENUM and CTRL_ENUM meta-data

Alarm and enumerated display meta-data (no timestamp). Length: ?? bytes

```
struct metaGR_ENUM {
    epicsInt16 status;
    epicsInt16 severity;
    epicsInt16 number_of_string_used;
    char strings[16][26];
};
```

The `strings` field is an array of 16 string of 26 characters. The `number_of_string_used` gives the number of entries in the `strings` field which are valid. Additional `strings` should be ignored, even if they contain non-null bytes.

12. Constants

12.1. Port numbers

Although there is no requirement as to which port numbers are used by either servers or clients, there are some standard values which must be used as defaults, unless overriden by application.

Port numbers are dependant on protocol versions and are calculated using the folowing definitions:

`CA_PORT_BASE` = 5056

`CA_SERVER_PORT` = `CA_PORT_BASE` + `MAJOR_PROTOCOL_VERSION` * 2

`CA_REPEATER_PORT` = `CA_PORT_BASE` + `MAJOR_PROTOCOL_VERSION` * 2 + 1

Based on protocol version described in this document (4.11), port numbers used are `CA_SERVER_PORT` = 5064 and `CA_REPEATER_PORT` = 5065.

Since registration of port numbers with IANA and in the interest of compatibility, the version numbers are unlikely to change. Therefore, the port numbers described here (5064 and 5065) may be considered final.

12.2. Representation of constants

This section lists various constants, their types and values used by protocol.

Some constants can be combined using logical OR operation. Example: Monitor mask of `DBE_VALUE` and `DBE_ALARM` are combined using `(DBE_VALUE or DBE_ALARM)` resulting in `(1 or 4 == 5)`.

To query the whether certain value is present in such combined value, and operation is used. Example: to query whether `DBE_ALARM` of monitor mask is set, `(DBE_VALUE and MASK > 0)` will return 0 if `DBE_VALUE` is not present, otherwise `DBE_ALARM` is present.

12.3. Monitor Mask

Indicates which changes to the value should be reported back to client library. Different values can be combined using logical OR operation.

Type: not defined, depends on the field it is in (usually `UINT16`)

- `DBE_VALUE` - value 1 (`0x01`) - Value change events are reported. Value changes take into consideration a dead band within which the value changes are not reported.
- `DBE_LOG` - value 2 (`0x02`) - Log events are reported. Similiar to `DBR_VALUE`, `DBE_LOG` defines a different dead band value that determines frequency of updates.
- `DBE_ALARM` - value 4 (`0x04`) - Alarm events are reported whenever alarm value of the channel changes.
- `DBE_PROPERTY` - value 8 (`0x08`) - Property events are reported when some metadata value associated with the channel changes. (Introduced in EPICS Base 3.14.11).

Notes

- CA Servers SHOULD ignore unknown monitor mask bits.
- Older PCAS versions will respond to unknown bits with `ECA_BADMASK`.

12.4. Search Reply Flag

Indicates whether server should reply to failed search messages. If a server does not know about channel name, it has the option of replying to request or ignoring it. Usually, servers contacted through address list will receive request for reply.

Type: not defined, depends on the field it is in (usually `UINT16`).

- `DO_REPLY` - value 10 (`0x0a`) - Server should reply to failed search requests.
- `DONT_REPLY` - value 5 (`0x05`) - Server should ignore failed requests.

12.5. Access Rights

Defines access rights for a given channel. Accss rights are defined as logicaly ORred value of allowed access.

Type: not defined, depends on the field it is in (usually `UINT16`).

- `CA_PROTO_ACCESS_RIGHT_READ` - value 1 (`0x01`) - Read access is allowed
- `CA_PROTO_ACCESS_RIGHT_WRITE` - value 2 (`0x02`) - Write access is allowed.

As a reference, the following values are valid.

- 0 - No access
- 1 - Read access only
- 2 - Write access only
- 3 - Read and write access

Servers MUST set undefined bits to zero. Clients MUST ignore undefined bits in this field.

13. Example message

This example shows construction of messages. For details of individual structures, see message and data type reference (`CA_PROTO_READ_NOTIFY` and `DBR_GR_INT16`).

A client will send `CA_PROTO_READ_NOTIFY` message with the following contents.

- Data type: `DBR_GR_INT16`

- Element count: 5
- Server ID: 22 (obtained during channel creation)
- Sequence ID: 56 (each read or write request increases value by one)

The messsage would be represented as follows:

```
00 0F (command) 00 00 (payload size) 00 16 (data type) 00 05 (element count)
00 00 00 16 (server ID) 00 00 00 38 (sequence ID)
```

Server would respond with success and return requested value with individual `DBR_GR_INT16` fields having the following values.

- Status: `ECA_NORMAL`
- Severity: `NO_ALARM` (0)

```
00 0f (command) 00 20 (payload size) 00 16 (data type) 00 05 (element count)
00 00 00 16 (server ID) 00 00 00 38 (sequence ID)
00 05  00 02  43 6f 75 6e 74 73 00 00  00 0a  00 00
00 08  00 06  00 04  00 02  00 00  00 00  00 00 00 00
    8      6      4      2      0      0      0     0
```

14. Repeater Operation

A repeater MUST be used by clients to collect CA_PROTO_RSRV_IS_UP messages. Each client host will have one repeater.

14.1. Startup

Each client MUST test for presence of repeater on startup, before any access to EPICS hosts is made. This check is made by attempting to bind to `CA_REPEATER_PORT`. If binding fails, the client may assume the repeater is already running and may attempt to register. This is done by sending `CA_REPEATER_REGISTER` datagram to `CA_REPEATER_PORT`. If repeater is already active, it will respond with `CA_REPEATER_CONFIRM` datagram back to client. At this point the registration is complete, and the repeater will begin forwarding messages to the client.

If binding succeeds, then this client process MUST either close the bound socket (and report at error) or begin functioning as a repeater.

If an error is encountered with sending `CA_REPEATER_REGISTER`, the the binding test SHOULD be repeated after a short timeout (1 second is RECOMMENDED).

14.2. Client detection

The repeater SHOULD test to see if its clients exist by periodically attempting to bind to their ports. If unsuccessful when attempting to bind to the client's port, then the repeater concludes that the client no longer exists. A technique using connected UDP sockets and ICMP destination unreachable MAY also used. If a client is determined to no longer be present then the repeater un-registers that client and no longer sends messages to it.

14.3. Operation

Each message the repeater receives MUST be forwarded to all registered clients.

14.4. Shutdown

Repeater should not shutdown on its own, if it does, there should be no active clients registered with it.

15. Searching Strategy

This section describes one possible strategy for handling CA_PROTO_SEARCH messages by a CA client. It is designed to limit the maximum rate at which search messages are sent to avoid overwhelming servers.

For each outstanding search request the following information is kept.

```
struct searchPV {
    const char *pvname;
    epicsTimeStamp nextSend;
    double intervalMult;
};
```

A priority queue should be maintained which is sorted in order of increasing `nextSend`.

When a new search request is made, a new `searchPV` is added to the queue with `initialMult` at a minimum (eg. 0.05 sec.) and `nextSend` at the present time plus `nextSend`.

When a search request is canceled it should be removed from the queue.

A task should run whenever the first entry expires (`nextSend` before the present time). This task should extract some expired entries up to a maximum limit (eg. enough for 4 UDP packets).

Search messages are then sent for these entries and their `intervalMult` is increased (eg. doubled), their `nextSend` is set to the present time plus `nextSend`, and they are re-added to the queue.

The task should then wait for the minimum search interval (eg. 0.05 sec.) before checking the queue again. This prevents a flood of search messages.

The combination of the minimum interval between sending search messages, and the limit on the maximum number of messages sent in each interval, acts to limit to total network bandwidth consumed by searches.

16. ECA Error/Status Codes

This section covers return codes and exceptions that can occur during CA command processing. In general, exceptions will be used to report various events to the application. Return codes are predefined values for conditions that can occur, where as exceptions are actually reported. Apart from exceptions that occur on server or due to network transport, additional error conditions may be reported on the client side as local exceptions.

Return codes are represented as `UINT16`. The 3 least significant bits indicate severity, remaining 13 bits are return code ID.

Return codes are communicated in the protocol by the `CA_PROTO_READ_NOTIFY`, `CA_PROTO_WRITE_NOTIFY`, monitor subscription responses, and the `CA_PROTO_ERROR` responses.

Severity codes

Code	Value	Description
CA_K_SUCCESS	1	Successful (not an error)
CA_K_WARNING	0	Not successful
CA_K_INFO	3	Informational (not an error)
CA_K_ERROR	2	Recoverable failure
CA_K_SEVERE	4	None recoverable failure

Presently defined error conditions

Code	Severity	ID	Value	Description
ECA_NORMAL	CA_K_SUCCESS	0	0x001	Normal successful completion
ECA_ALLOCMEM	CA_K_WARNING	6	0x030	Unable to allocate additional dynamic memory
ECA_TOLARGE	CA_K_WARNING	9	0x048	The requested data transfer is greater than available memory or EPICS_CA_MAX_ARRAY_BYTES
ECA_TIMEOUT	CA_K_WARNING	10	0x050	User specified timeout on IO operation expired
ECA_BADTYPE	CA_K_ERROR	14	0x072	The data type specifed is invalid
ECA_INTERNAL	CA_K_FATAL	17	0x08e	Channel Access Internal Failure
ECA_DBLCLFAIL	CA_K_WARNING	18	0x090	The requested local DB operation failed
ECA_GETFAIL	CA_K_WARNING	19	0x098	Channel read request failed
ECA_PUTFAIL	CA_K_WARNING	20	0x0a0	Channel write request failed
ECA_BADCOUNT	CA_K_WARNING	22	0x0b0	Invalid element count requested
ECA_BADSTR	CA_K_ERROR	23	0x0ba	Invalid string
ECA_DISCONN	CA_K_WARNING	24	0x0c0	Virtual circuit disconnect
ECA_EVDISALLOW	CA_K_ERROR	26	0x0d2	Request inappropriate within subscription (monitor) update callback
ECA_BADMONID	CA_K_ERROR	30	0x0f2	Bad event subscription (monitor) identifier
ECA_BADMASK	CA_K_ERROR	41	0x14a	Invalid event selection mask
ECA_IODONE	CA_K_INFO	42	0x153	IO operations have completed
ECA_IOINPROGRESS	CA_K_INFO	43	0x15b	IO operations are in progress
ECA_BADSYNCGRP	CA_K_ERROR	44	0x162	Invalid synchronous group identifier
ECA_PUTCBINPROG	CA_K_ERROR	45	0x16a	Put callback timed out

Code	Severity	ID	Value	Description
ECA_NORDACCESS	CA_K_WARNING	46	0x170	Read access denied
ECA_NOWTACCESS	CA_K_WARNING	47	0x178	Write access denied
ECA_ANACHRONISM	CA_K_ERROR	48	0x182	Requested feature is no longer supported
ECA_NOSEARCHADDR	CA_K_WARNING	49	0x188	Empty PV search address list
ECA_NOCONVERT	CA_K_WARNING	50	0x190	No reasonable data conversion between client and server types
ECA_BADCHID	CA_K_ERROR	51	0x19a	Invalid channel identifier
ECA_BADFUNCPTR	CA_K_ERROR	52	0x1a2	Invalid function pointer
ECA_ISATTACHED	CA_K_WARNING	53	0x1a8	Thread is already attached to a client context
ECA_UNAVAILINSERV	CA_K_WARNING	54	0x1b0	Not supported by attached service
ECA_CHANDESTROY	CA_K_WARNING	55	0x1b8	User destroyed channel
ECA_BADPRIORITY	CA_K_ERROR	56	0x1c2	Invalid channel priority
ECA_NOTTHREADED	CA_K_ERROR	57	0x1ca	Preemptive callback not enabled - additional threads may not join context
ECA_16KARRAYCLIENT	CA_K_WARNING	58	0x1d0	Client's protocol revision does not support transfers exceeding 16k bytes
ECA_CONNSEQTMO	CA_K_WARNING	59	0x1d9	Virtual circuit connection sequence aborted
ECA_UNRESPTMO	CA_K_WARNING	60	0x1e0	?

Historical error conditions. Servers and clients SHOULD NOT send these codes, but MAY receive them.

Code	Severity	ID	Value	Description
ECA_MAXIOC	CA_K_ERROR	1	0x00a	Maximum simultaneous IOC connections exceeded
ECA_UKNHOST	CA_K_ERROR	2	0x012	Unknown internet host
ECA_UKNSERV	CA_K_ERROR	3	0x01a	Unknown internet service
ECA_SOCK	CA_K_ERROR	4	0x022	Unable to allocate a new socket
ECA_CONN	CA_K_WARNING	5	0x028	Unable to connect to internet host or service
ECA_UKNCHAN	CA_K_WARNING	7	0x038	Unknown IO channel
ECA_UKNFIELD	CA_K_WARNING	8	0x040	Record field specified inappropriate for channel specified
ECA_NOSUPPORT	CA_K_WARNING	11	0x058	Sorry, that feature is planned but not supported at this time
ECA_STRTOBIG	CA_K_WARNING	12	0x060	The supplied string is unusually large
ECA_DISCONNCHID	CA_K_ERROR	13	0x06a	The request was ignored because the specified channel is disconnected
ECA_CHIDNOTFND	CA_K_INFO	15	0x07b	Remote Channel not found
ECA_CHIDRETRY	CA_K_INFO	16	0x083	Unable to locate all user specified channels

Code	Severity	ID	Value	Description
ECA_DBLCHNL	CA_K_WARNING	25	0x0c8	Identical process variable name on multiple servers
ECA_ADDFAIL	CA_K_WARNING	21	0x0a8	Channel subscription request failed
ECA_BUILDGET	CA_K_WARNING	27	0x0d8	Database value get for that channel failed during channel search
ECA_NEEDSFP	CA_K_WARNING	28	0x0e0	Unable to initialize without the vxWorks VX_FP_TASK task option set
ECA_OVEVFAIL	CA_K_WARNING	29	0x0e8	Event queue overflow has prevented first pass event after event add
ECA_NEWADDR	CA_K_WARNING	31	0x0f8	Remote channel has new network address
ECA_NEWCONN	CA_K_INFO	32	0x103	New or resumed network connection
ECA_NOCACTX	CA_K_WARNING	33	0x108	Specified task isnt a member of a CA context
ECA_DEFUNCT	CA_K_FATAL	34	00x116	Attempt to use defunct CA feature failed
ECA_EMPTYSTR	CA_K_WARNING	35	0x118	The supplied string is empty
ECA_NOREPEATER	CA_K_WARNING	36	0x120	Unable to spawn the CA repeater thread- auto reconnect will fail
ECA_NOCHANMSG	CA_K_WARNING	37	0x0128	No channel id match for search reply- search reply ignored
ECA_DLCKREST	CA_K_WARNING	38	0x130	Reseting dead connection- will try to reconnect
ECA_SERVBEHIND	CA_K_WARNING	39	0x138	Server (IOC) has fallen behind or is not responding- still waiting
ECA_NOCAST	CA_K_WARNING	40	0x140	No internet interface with broadcast available

17. Example conversation

This is example conversation between client and server. Client first establishes TCP connection to the server and immediately requests creation of a channel. After server aknowledges channel creation, client reads the value of the channel twice. First as a single string value and second as a DBR_GR_INT16 type. After the response to both queries has been received, the channel is destroyed.

```
Client to Server
CA_PROTO_VERSION (handshake)
00 00  00 00  00 00  00 0b  00 00 00 00  00 00 00 00
    0      0      0     11            0           0
CA_PROTO_CLIENT_NAME (handshake)
00 14  00 08  00 00  00 00  00 00 00 00  00 00 00 00  61 70 75 63 65 6c 6a 00
   20      8      8      0           0           0   a  p  u  c  e  l  j \0
CA_PROTO_HOST_NAME (handshake)
00 15  00 08  00 00  00 00  00 00 00 00  00 00 00 00  63 73 6c 30 36 00 00 00
   21      8      0      0           0           0   c  s  l  0  6 \0 \0 \0
CA_PROTO_CREATE_CHAN (request)
00 12  00 18  00 00  00 00  00 00 00 01  00 00 00 0b
   18     24      0      0           1          11
61 70 75 63 65 6c 6a 3a  61 69 45 78 61 6d 70 6c 65 31 00 00 00 00 00 00
 a  p  u  c  e  l  j  :   a  i  E  x  a  m  p  l  e  1 \0 \0 \0 \0 \0 \0

Server to Client
CA_PROTO_ACCESS_RIGHTS (handshake)
00 16  00 00  00 00  00 00  00 00 00 01  00 00 00 03
   22      0      0      0           1           3
CA_PROTO_CREATE_CHAN (response)
```

```
00 12  00 00  00 06  00 01  00 00 00 01  00 00 00 04
   18      0      6      1           1           4
|
Client to Server
CA_PROTO_READ_NOTIFY (request)
00 0f  00 00  00 00  00 01  00 00 00 04  00 00 00 01
   15      0      0      1           4           1
CA_PROTO_READ_NOTIFY (request)
00 0f  00 00  00 16  00 01  00 00 00 04  00 00 00 02
   15      0     22      1           4          02

Server to Client
CA_PROTO_READ_NOTIFY (response)
00 0f  00 08  00 00  00 01  00 00 00 01  00 00 00 01  30 00 00 00 00 06 00 01
   15      8      0      1           1           1   0
CA_PROTO_READ_NOTIFY (response)
00 0f  00 20  00 16  00 01  00 00 00 01  00 00 00 02
   15     32     22      1           1          02
00 05  00 02  43 6f 75 6e 74 73 00 00  00 0a  00 00
    5      2   C  o  u  n  t  s \0 \0     10      0
00 08  00 06  00 04  00 02  00 00  00 00  00 00 00 00
    8      6      4      2      0      0      0     0

Client to Server
CA_PROTO_CLEAR_CHANNEL (request)
00 0c  00 00  00 00  00 00  00 00 00 04  00 00 00 01
   12      0      0      0           4           1

Server to Client
CA_PROTO_CLEAR_CHANNEL (response)
00 0c  00 00  00 00  00 00  00 00 00 04  00 00 00 01
   12      0      0      0           4           1
```

18. Glossary of Terms

IOC
: Input/Output Controller.

PV
: Process variable.

Virtual circuit
: Reusable TCP connection between client and server, through which all PVs hosted by the server can be conveyed to the client.

19. References

ID	Author	Reference	Revision	Date	Publisher
1	Jeffrey O. Hill	Channel Access Reference Manual	R3.14	2003	
2		Java Channel Access	2.0.1	2003	
3	Bradner, S.	RFC 2119: Key words for use in RFCs to Indicate Requirement Levels		1997-03	

Version 1.5
Last updated 2016-11-10 17:34:37 CST