
CAT Document 001
TAC Market Design: Communication Protocol
Specification

Version 1.11
December 22, 2006

DRAFT

Contents

1	Introduction	5
2	Change Log	5
3	Game Configuration	6
3.1	Timing	7
3.2	Trading Agents	7
3.2.1	Trading strategies	8
3.2.2	Market Selection strategies	8
3.3	Fees	8
3.4	Evaluation	9
4	Notational Conventions and Generic Grammar	9
5	Protocol Parameters	10
5.1	CATP Version	10
6	Connections	11
7	CATP Messages	11
7.1	Client ID	12
7.2	Shout ID	12
7.3	Transaction ID	12
7.4	Message Types	12
7.5	Message Headers	13
7.6	Message Body	13
8	Requests	14
8.1	CHECKIN	14
8.2	OPTIONS	16
8.2.1	GAMESTARTING	16
8.2.2	GAMESTARTED	18
8.2.3	GAMEOVER	18
8.2.4	DAYOPENING	18
8.2.5	DAYOPENED	19
8.2.6	DAYCLOSED	19
8.2.7	ROUNDOPENED	20
8.2.8	ROUNDCLOSED	20
8.3	REGISTER	21

8.4	SUBSCRIBE	21
8.5	POST	23
8.6	GET	25
8.7	ASK	27
8.8	BID	30
8.9	TRANSACTION	31
9	Responses	32
9.1	OK	33
9.2	INVALID	33
9.3	ERROR	34
10	Message Headers	34
10.1	Time	34
10.2	Tag	35
10.3	Id	35
10.4	Type	35
10.5	Value	35
10.6	Text	35
10.7	Version	36

List of Figures

1	CHECKIN messages.	16
2	OPTIONS messages.	17
3	REGISTER messages.	22
4	SUBSCRIBE messages.	23
5	POST messages.	26
6	Dialogs regarding asks.	29
7	Dialogs regarding bids.	30
8	Dialogs regarding transactions.	32

1 Introduction

This document presents the specification of communication protocol for TAC Market Design Competition, or CAT, games. The protocol is denoted as CAT Protocol, or in short CATP. The TAC/CAT competition platform has been implemented in Java through the JCAT project [1].

This document has been developed as a result of inputs and discussions from teams at Brooklyn College (Rayman Chan, Jinzhong Niu, Simon Parsons, Elizabeth Sklar), the University of Liverpool (Peter McBurney, Steve Phelps), and the University of Southampton (Enrico Gerding, Albert Mmoloke). The CAT Mechanism Design Competition will be run in summer 2007 under the auspices of the Trading Agents Competition (TAC), and sponsored by the UK EPSRC Project, “Market Based Control of Complex Computational Systems” (GR/T19742/01), a research project undertaken jointly by the Universities of Birmingham, Liverpool and Southampton, UK. Further information is available from the project web-site:

<http://www.marketbasedcontrol.com/>.

© 2006, University of Liverpool and Brooklyn College. All rights reserved.

2 Change Log

- Version 1.00 : first version by Jinzhong Niu in July, 2006.
- Version 1.01 : (Jinzhong Niu, 08/02/2006) additional diagrams by Albert Mmoloke showing possible interactions during a CAT game.
- Version 1.02 : (Jinzhong Niu, 08/06/2006) typos and inconsistencies fixed.
- Version 1.03 : (Jinzhong Niu, 08/28/2006) HELLO and MATCH messages renamed to CHECKIN and TRANSACTION; SUBSCRIBE message added.
- Version 1.04 : (Jinzhong Niu, 09/01/2006) POST messages used to notify of successful shouts and transactions instead of ASK, BID, and TRANSACTION, see Sections 8.5, 8.7, and 8.9.
- Version 1.05 : (Jinzhong Niu, 09/18/2006) typos fixed.

- Version 1.06 : (Simon Parsons, 10/05/2006) typographical fixes, some additional explanation added.
- Version 1.07 : (Jinzhong Niu, 11/07/2006)
 - Section 3.2 reorganized and updated;
 - Section 3.3 updated;
 - **V**ersion header added and protocol parameter in **C**HECKIN messages replaced with a **V**ersion header;
 - Section 8.2 updated with additional diagrams, and newly added game and day initialization;
 - Sections 8.3, 8.4, 8.5, and 8.6 updated with additional diagrams;
 - Sections 8.7, 8.8, and 8.9 updated reflecting changes on **A**SK, **B**ID, and **T**RANSACTION processing;
 - **T**ag header added to allow detection of obsolete messages due to network delay; and
 - The section on possible dialogs in different scenarios added in Version 1.01 removed; diagrams are now distribute in other related sections.
- Version 1.08 : (Jinzhong Niu, 11/20/2006) list of figures added after table of contents.
- Version 1.09 : (Jinzhong Niu, 11/22/2006) notification of profits added at the end of each trading day through **O**PTIONS **D**AYCLOSED messages; **R**OUNDOPENING changed to **R**OUNDOPENED to be consistent with **J**CAT code.
- Version 1.10 : (Jinzhong Niu, 12/04/2006) **P**OST **P**ROFIT added to notify of specialist' profits at the end of each trading day (see Section 8.5); daily trader distribution announcement added at the end of each day through **O**PTIONS **D**AYCLOSED (see Section 8.2.6).
- Version 1.11 : (Jinzhong Niu, 12/22/2006) inputs from Peter McBurney added.

3 Game Configuration

This section introduces the configuration of CAT games, helping make this document self-explanatory. For more information, please refer to [2].

It is assumed that a CAT game consists of a CATP server and several CATP clients, which may be trading agents or specialists (markets). CATP clients do not talk to each other directly; instead they connect to the CATP server through socket and the server responds to messages from clients and forward information if needed.

NOTE: The meta-exchange in the CAT game architecture in [2] is not taken into consideration in this specification for now.

3.1 Timing

A CAT game lasts a certain number of *days* and each day consists of *rounds*. The CATP server in a game has the control on a central clock and notifies clients of the following events:

- start of game
- end of game
- start of a day (day open)
- end of a day
- start of a round (round open)
- end of a round

In order to allow traders and specialists to exploit some deadline effects, the server also notifies clients of the length of a day — in terms of number of rounds — and the length of each round — in terms of the number of physical milliseconds — before a game starts.

NOTE: As described below, the length of a game — in terms of the number of days — is not disclosed by the server to any of the clients.

3.2 Trading Agents

In a CAT competition, trading agents are provided by the CAT game. Each trading agent is associated with a *trading strategy* and a *market selection strategy* and assigned private values for the goods being traded.

Private values, which determines the demand and supply of the markets, are allocated by the CATP server, and change from day to day. However, private values remain constant during a day.

3.2.1 Trading strategies

The set of trading strategies that may be used by traders in CAT games include:

- GD [4]
- ZIP [3]
- RE (Roth and Erev) [7]
- ZI-C [5]

NOTE: The sizes of the sub-populations of trading agents adopting a different trading strategy may or may not be made public before a game takes place.

3.2.2 Market Selection strategies

Traders transact business through specialists. When a trading day opens, each trader is registered with one specialist (and only one specialist) and stays with that specialist until the day closes. Traders may move to other specialists over days. A trader is only allowed to make shouts and transactions through the specialist it is registered with.

NOTE: The full set of strategies that traders will use in order to select between specialists have yet to be developed and determined, and it is still under discussion whether they should be made public or not in advance.

3.3 Fees

A specialist can make a profit by charging a trader when the latter:

- is registered with the specialist (*registration fee*);
- requests to receive information on shouts and/or transactions taking place through the specialist (*information fee*);
- makes a shout (*shout fee*);
- is involved in a transaction (*transaction fee*); and
- makes a profit in a transaction (*profit fee*).

A specialist is free to set the level of these charges (from zero up). The first 4 types of fees are each a flat charge, and the last one is a percentage charged on the profit made by a trader.

Specialists are required to send information to the server on how much they charge for each type of activity when a trading day opens. Specialists are permitted to adapt their charging policies over time.

3.4 Evaluation

Specialists in a CAT game will be evaluated across a number of performance metrics. One such metric is the profit they make through the game. This profit is the sum of all the fees the specialist receives, plus the sum of all sales the specialist makes, minus the cost of all the purchases the specialist makes. If a specialist pays to obtain information on shouts and transactions taking place in markets run by other specialists, the charges incurred are deducted from its profit.

The current version of the CAT game *only* evaluates specialists by the profit that they make. And at the end of each trading day, the CAT server informs all the specialists of the profits they have made so far¹ 8.5.

The evaluation process to be used in the 2007 CAT Tournament is described in a separate document, CAT Document 003 — TAC/CAT Tournament 2007: Evaluation of Entrants.

4 Notational Conventions and Generic Grammar

The following rules from the HTTP 1.1 Specification [6] are used throughout this specification to describe basic parsing constructs.

OCTET	= <any 8-bit sequence of data>
CHAR	= <any US-ASCII character (octets 0 - 127)>
UPALPHA	= <any US-ASCII uppercase letter "A".."Z">
LOALPHA	= <any US-ASCII lowercase letter "a".."z">
ALPHA	= UPALPHA LOALPHA
DIGIT	= <any US-ASCII digit "0".."9">
CTL	= <any US-ASCII control character (octets 0 - 31) and DEL (127)>
CR	= <US-ASCII CR, carriage return (13)>

¹This notification of profits, together with notifications of price lists at the beginning of each day and trader distribution among specialists, somehow allows specialists to learn to adapt their charging policies to maximize profits over the long-run.

LF	= <US-ASCII LF, linefeed (10)>
SP	= <US-ASCII SP, space (32)>
HT	= <US-ASCII HT, horizontal-tab (9)>
<">	= <US-ASCII double-quote mark (34)>
CRLF	= CR LF
LWS	= [CRLF] 1*(SP HT)
TEXT	= <any OCTET except CTLs, but including LWS>
HEX	= "A" "B" "C" "D" "E" "F" "a" "b" "c" "d" "e" "f" DIGIT
token	= 1*<any CHAR except CTLs or separators>
separators	= "(" ")" "<" ">" "@" "," ";" ":" "\" <"> "/" "[" "]" "?" "=" "{" "}" SP HT

5 Protocol Parameters

5.1 CATP Version

CATP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further CATP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values. The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

CATP-Version = "CATP" "/" 1*DIGIT "." 1*DIGIT

See Section 8.1 for how CATP version is used in CHECKIN messages.

NOTE: The major and minor numbers must be treated as separate integers and that each may be incremented higher than a single digit.

6 Connections

A CATP connection is setup between a CATP server and a CATP client for message transmission.

A server listens on port 9090² upon startup. Clients connect to the port and establish connections. CATP connections work in a way similar to a persistent HTTP connection. Normally they should be kept alive from the beginning of a game through the end³.

In case a connection is broken while a competition is running, the client involved may attempt to establish a new connection to the server and take part in the game beginning with the next trading day. It is the responsibility of the client to detect that a connection has been broken, and to establish a new connection⁴.

7 CATP Messages

In a CAT game, plain-text messages are transmitted over CATP connections for the CATP server and CATP clients to communicate with each other.

Since a message is always transmitted over a point-to-point CATP connection, there is no need to include the identities of the sender and the receiver in the message.

It is the server's responsibility to keep copies of messages from clients in some way for the purposes of security, robustness, and game-wide information integrity⁵.

²The port number could be chosen arbitrary as long as it does not conflict with any existing known service.

³In JCAT, multiple games can be run in a batch mode and the CATP server will not attempt to close the connections with clients until the batch run is completed

⁴When a client needs to reestablish a connection, it should check in first before synchronizing with the server on next trading day. See Section 8.1.

⁵[2] proposes a stand-alone registry component for logging all the messages, but discussion of it is beyond the scope of this document since its existence and behavior do not affect the way CATP is designed and implemented.

7.1 Client id

Each client, trader or specialist, in a CAT game is assigned a unique ID by the server, which is used in messages whenever needed to specify a client. A client ID should be a **token**.

```
client-id = token
```

7.2 Shout id

After a shout is placed by a trader and confirmed valid by the server, it is assigned a unique ID by the server, which has identical syntax to client IDs.

```
shout-id = token
```

7.3 Transaction id

After a transaction is made by a specialist and confirmed valid by the server, it is assigned a unique ID by the server, which has identical syntax to client IDs.

```
transaction-id = token
```

7.4 Message Types

Following the convention with HTTP messages, CATP messages include requests and responses.

NOTE: Requests can be made by either a client or server in CATP, which is not true in HTTP however.

```
message = request | response
```

Both types of message consist of a **start-line**, zero or more header fields (also known as **headers**), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and possibly a **message-body**.

```
generic-message = start-line
                  *(message-header CRLF)
                  CRLF
                  [ message-body ]
start-line       = message-type CRLF
message-type     = request-type
                  | response-type
```

Often, the `message-type` is sufficient to completely identify the purpose of a message. When this is not the case, a `Type` header should be used to provide additional information (see Sections 8.1, 8.2, 8.5, 8.6, 8.7, 8.8, and 10.4 for more information).

7.5 Message Headers

CATP message header fields are included in messages providing additional information.

Each header field consists of a name followed by a colon (:) and the field value. Field names are case-insensitive. The field value may be preceded by any amount of LWS, though a single SP is preferred.

```

message-header = field-name ":" [ field-value ]
field-name     = token
field-value    = *( field-content | LWS )
field-content  = <the OCTETs making up the field-value
                  and consisting of either *TEXT or
                  combinations of token, separators,
                  and quoted-string>

```

The `field-content` does not include any leading or trailing LWS occurring before the first non-whitespace character of the `field-value` or after the last non-whitespace character of the `field-value`. Such leading or trailing LWS may be removed without changing the semantics of the field value. Any LWS that occurs between `field-content` may be replaced with a single SP before interpreting the field value or forwarding the message downstream.

The order in which header fields with differing field names are received is not significant.

Multiple `message-header` fields with the same `field-name` may be present in a message if and only if the entire `field-value` for that header field is defined as a comma-separated list, i.e. `#(value)`. It must be possible to combine the multiple header fields into one `field-name: field-value` pair, without changing the semantics of the message, by appending each subsequent `field-value` to the first, each separated by a comma.

7.6 Message Body

No CATP message currently uses a message body.

8 Requests

A request message requests the receiver to take actions. Types of request include:

```
request-type = "CHECKIN"
              | "REGISTER"
              | "SUBSCRIBE"
              | "ASK"
              | "BID"
              | "TRANSACTION"
              | "GET"
              | "POST"
              | "OPTIONS"
```

8.1 CHECKIN

A CHECKIN request is used by a CATP client to take part in a CAT game once it connects to the server. A `Version` header, a `Type` header and a `Text` header are required in a CHECKIN request to provide the version of CATP the client supports, the type and the human-readable name of the CATP client.

EXAMPLE:

```
CHECKIN CRLF
Version: CATP/1.0 CRLF
Type: Seller CRLF
Text: CUNY Bot 1.0 CRLF
CRLF
```

The type of a client must be a string starting with (case insensitive) one of the following:

- **Buyer**: indicating the client is a trading agent that buys;
- **Seller**: indicating the client is a trading agent that sells; and
- **Specialist**: indicating the client is a specialist.

On receiving a CHECKIN request, the server allocates a client ID and sends back an OK response if the type of client is valid and the version of CATP supported by the server matches the version of CATP on the client side.

EXAMPLE:

```
OK CRLF
Id: Seller5 CRLF
CRLF
```

Otherwise, an INVALID message is replied.

EXAMPLE:

```
INVALID CRLF
Type: Version CRLF
Text: catp versions don't match. CRLF
CRLF
```

In the case that a client loses the connection with the server and attempts to reestablish a new connection, the client needs to make a CHECKIN request again before getting back to the on-going competition starting with next trading day. The CHECKIN request sent in this scenario includes an Id header, which presents the ID allocated to it before, and no other headers are required.

EXAMPLE:

```
CHECKIN CRLF
Id: Seller5 CRLF
CRLF
```

If the ID is valid, the server simply sends back an empty OK response, or otherwise an INVALID.

It is also possible that a CATP client proposes an ID for itself in its CHECKIN request by using an Id header. If the ID does not conflict with IDs already allocated to other clients, the server will respond with an OK with an optional Id header; otherwise, the ID in the request is simply overlooked and a new ID allocated to the client and sent back as usual.

EXAMPLE:

```
CHECKIN CRLF
Version: CATP/1.0 CRLF
Type: Seller CRLF
Text: CUNY Bot 1.0 CRLF
Id: bot0 CRLF
CRLF
```

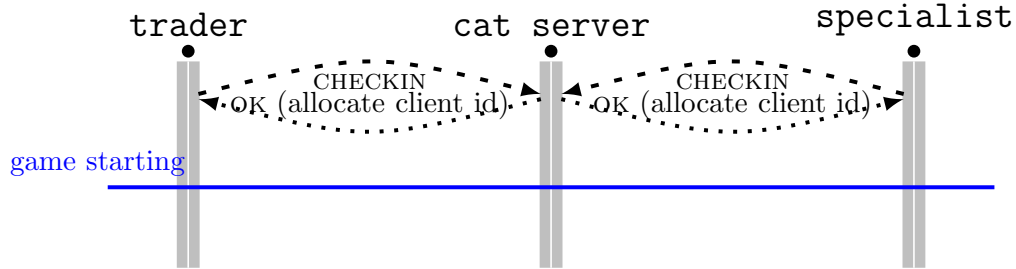


Figure 1: CHECKIN messages.

8.2 OPTIONS

OPTIONS requests are control messages sent by the CATP server to clients, involving timing, demand/supply schedule, etc. Types of OPTIONS messages are described in a `Type` header.

As Section 3.1 already stated, a CAT game lasts a certain number of days and each day consists of rounds. The CATP server notifies clients of the start and the end of a game, a day, and a round. There are both a *game starting* event and a *game started* event, between which initialization operations can be done. Similarly, there are both a *day opening* event and a *day opened* event on each trading day allowing daily initialization operations⁶.

Figure 2 shows the order of various OPTIONS messages from the CATP server to clients. These messages show the time control in a CAT game.

8.2.1 GAMESTARTING

An OPTIONS `GAMESTARTING` message notifies CATP clients of the starting of the game. A `Value` header with 2 integer values should be used along with the `GAMESTARTING` to inform traders of the length of a day and the length of a round. An `OK` response from clients is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: GAMESTARTING CRLF
Value: 20, 3600 CRLF
CRLF
```

⁶There is only a *round opened* event for a round currently.

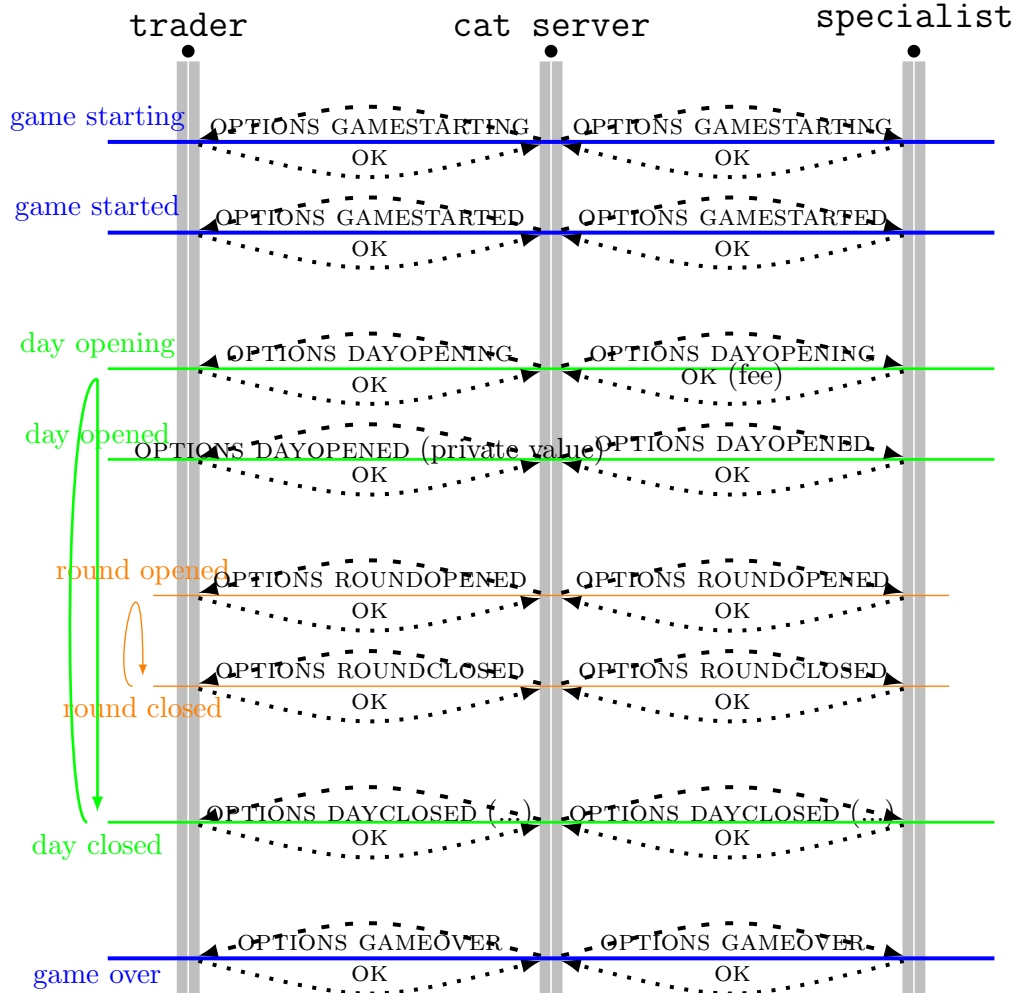


Figure 2: OPTIONS messages.

The `OPTIONS GAMESTARTING` message is followed by a `POST TRADER` message and a `POST SPECIALIST` message.

NOTE: The length of a game in days will not be known to CATP clients. The reason for this is to prevent specialists exploiting deadline effects due to the end of the game. The reason for this is that the CAT game is about designing markets with good steady-state behavior, and thus markets that work well on any given day.

8.2.2 GAMESTARTED

An `OPTIONS GAMESTARTED` message notifies CATP clients that the game just started. An OK response from clients is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: GAMESTARTED CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.2.3 GAMEOVER

An `OPTIONS GAMEOVER` message notifies CATP clients of the end of the game. An OK response from clients is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: GAMEOVER CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.2.4 DAYOPENING

An `OPTIONS DAYOPENING` message notifies CATP clients of the opening of a trading day.

The responses from specialists should include a price list in a `Value` header. See Sections [3.3](#), [8.5](#), and [8.6](#).

EXAMPLE:

```
OPTIONS CRLF
Type: DAYOPENING CRLF
CRLF
```

```
OK CRLF
Value: 2, 3, 6, 9, 0.2 CRLF
CRLF
```

A price list includes in order registration fee, information fee, shout fee, transaction fee, and profit fee as described in Section 3.3. In the above example, the specialist charges \$2 on registration per day, \$3 on subscription for information from one specialist per day, \$6 on each shout made⁷, \$9 on each transaction a trader is involved in, and 20% of profit a trader makes through a transaction.

8.2.5 DAYOPENED

An OPTIONS DAYOPENED message notifies CATP clients of the opening of a trading day.

In OPTIONS DAYOPENED messages to traders, a Value should be used to assign private values. Multiple values indicate multiple items for the trader to trade. An OK response from traders is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: DAYOPENED CRLF
Value: 65 CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.2.6 DAYCLOSED

An OPTIONS DAYCLOSED message notifies CATP clients that a trading day is closed. In the message, the CAT server also provides the number of traders being registered with each specialist by using an Id header with a list of specialists and a Value header with a list of integers. An OK response from clients is expected.

EXAMPLE:

⁷Only new shouts are charged, and shout modification is allowed and will not be charged again.

```
OPTIONS CRLF
Type: DAYCLOSED CRLF
Id: Specialist0, Specialist1, Specialist2 CRLF
Value: 20, 14, 53, 13 CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.2.7 ROUNDOPENED

An OPTIONS ROUNDOPENED message notifies CATP clients of the start of a trading round. An OK response from clients is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: ROUNDOPENED CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.2.8 ROUNDCLOSED

An OPTIONS ROUNDCLOSED message notifies CATP clients that a trading round is closed. An OK response from clients is expected.

EXAMPLE:

```
OPTIONS CRLF
Type: ROUNDCLOSED CRLF
CRLF
```

```
OK CRLF
CRLF
```

8.3 REGISTER

A REGISTER request from a trader client to the server expresses its desire to trade through the specialist specified in an Id header.

EXAMPLE:

```
REGISTER CRLF
Id: specialist3 CRLF
CRLF
```

If the registration is successful, the server responds with an empty OK message.

EXAMPLE:

```
OK CRLF
CRLF
```

The server then notifies the corresponding specialist of the trader's registration with a REGISTER request, which uses an Id header to specify which trader is registered.

EXAMPLE:

```
REGISTER CRLF
Id: Seller3 CRLF
CRLF
```

If everything is okay, the specialist replies with an OK response.

REGISTER requests are expected to be sent by traders after a trading day is opened. Its processing is illustrated in Figure 3.

8.4 SUBSCRIBE

A SUBSCRIBE request from a trader to the server tells the latter the trader would pay for information regarding shouts and transactions made through some specialists. An Id header gives a list of specialists from which the trader want to purchase information.

EXAMPLE:

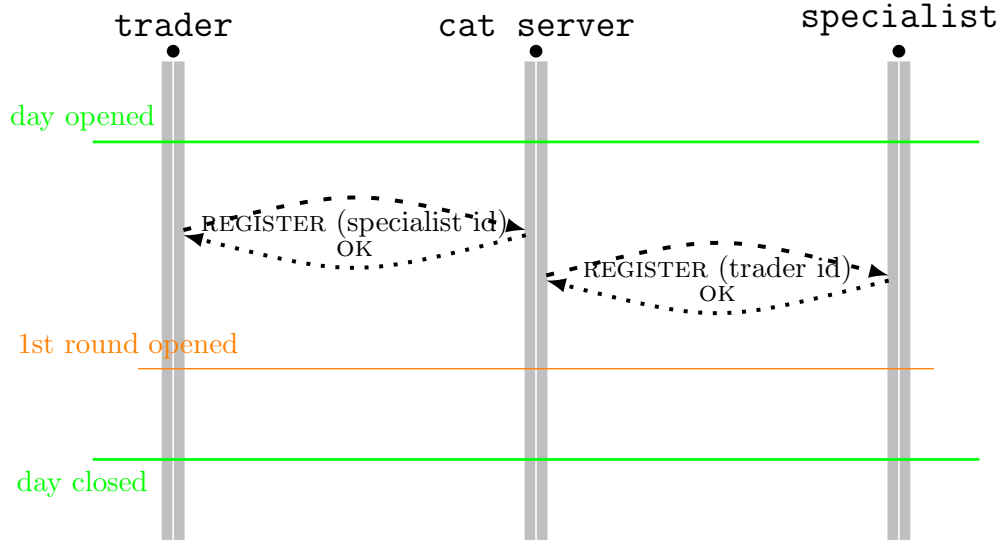


Figure 3: REGISTER messages.

```

SUBSCRIBE CRLF
ID: specialist1, specialist3 CRLF
CRLF

```

The server responds to a SUBSCRIBE message with an OK message and then notifies those specialists by sending a SUBSCRIBE message with an Id header telling which trader subscribes.

EXAMPLE:

```

SUBSCRIBE CRLF
ID: trader2 CRLF
CRLF

```

A specialist replies to a SUBSCRIBE message with an OK and may use the notification to calculate its profit for its personal use, e.g. feeding the signal into some learning algorithm to adjust its parameters.

SUBSCRIBE requests can be sent any time during a trading day and the server is responsible for broadcasting the necessary information to subscribed traders⁸. Its processing is illustrated in Figure 4.

⁸In JCAT, if a trader's strategies requires information on shouts and transactions at specialists, a trader sends a SUBSCRIBE request after it is successfully registered with the specialist through which it chose to do business.

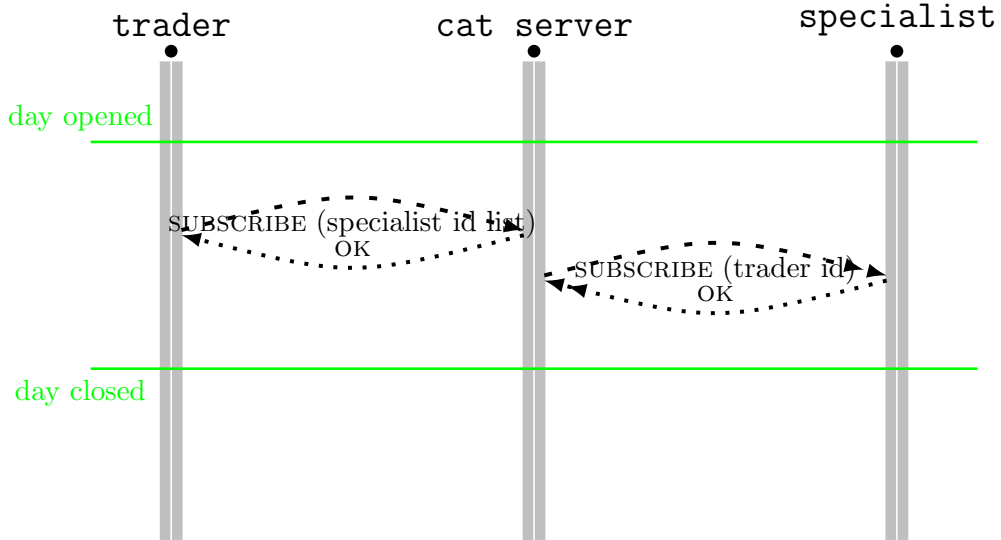


Figure 4: SUBSCRIBE messages.

8.5 POST

POST messages are used to proactively provide (push) information to receivers that is accessible without charge. The information may also be requested otherwise by CATP clients through GET messages.

The following information may be provided through POST messages:

- **TRADER and SPECIALIST:** After a CAT game starts but before the first day starts, the server is required to send POST messages to all the traders and specialists providing the list of traders, and the list of specialists.

EXAMPLE:

```

POST CRLF
Type: TRADER CRLF
Id: Buyer0, Buyer1, Seller0 CRLF
CRLF

POST CRLF
  
```

```
Type: SPECIALIST CRLF
Id: Specialist0, Specialist1, Specialist2 CRLF
CRLF
```

- **FEE:** fees charged by a specialist, i.e. a price list. The specialist ID should be specified in an Id header. The price lists are sent from specialists in the responses to **OPTIONS DAYOPENING** messages from the server, and **POST FEE** messages are broadcast by the server within the day initialization period, namely between **DAYOPENING** and **DAYOPENED**.

EXAMPLE:

```
POST CRLF
Type: FEE CRLF
Id: Specialist0 CRLF
Value: 0, 0, 0, 0, 0 CRLF
CRLF
```

...

```
POST CRLF
Type: FEE CRLF
Id: Specialist2 CRLF
Value: 3, 3, 6, 9, 0.3 CRLF
CRLF
```

In the above example, **Specialist2** charges \$3 on registration per day, \$3 on subscription for information from one specialist per day, \$6 on each shout made⁹, \$9 on each transaction a trader is involved in, and 20% of profit a trader makes through a transaction.

- **PROFIT:** The server notifies clients of the profits made by specialists. An Id header presents the ID list of specialists and a Value header presents the amounts of their profits up to the moment in the order of specialists in the Id header.

EXAMPLE:

⁹Only new shouts are charged, and shout modification is allowed and will not be charged again.

```

POST CRLF
Type: PROFIT CRLF
Id: Specialist0, Specialist1, Specialist2, Specialist3 CRLF
Value: 123, 456, 789, 1023 CRLF
CRLF

```

POST PROFIT messages are broadcast at the end of each trading day, just before the OPTIONS DAYCLOSED message.

- ASK and BID: The server sends notification of successful ASKS and BIDS, see Sections 8.7 and 8.8.
- TRANSACTION: The server sends notification of successful TRANSACTIONS, see Section 8.9.

POST TRADER, POST SPECIALIST, POST FEE, and POST PROFIT messages are illustrated in Figure 5, while POST ASK, POST BID, and POST TRANSACTION are presented respectively in Figure 6, Figure 7, and Figure 8.

8.6 GET

A GET request from a CATP client to the server asks for information that is accessible without charge. The information is also broadcast by the CATP server through POST messages as described in Section 8.5.

A Type header should be used to specify what type of information is requested. It can be

- TRADER and SPECIALIST: list of trader clients, and list of specialist clients. An Id header should be used by the server in the response given the list of client IDs requested.

GET TRADER and GET SPECIALIST requests can be made any time after a CAT game is started.

- FEE: fees charged by a specialist. The specialist ID should be specified in an Id header.

EXAMPLE:

```

GET CRLF
Type: FEE CRLF
Id: Specialist3 CRLF
CRLF

```

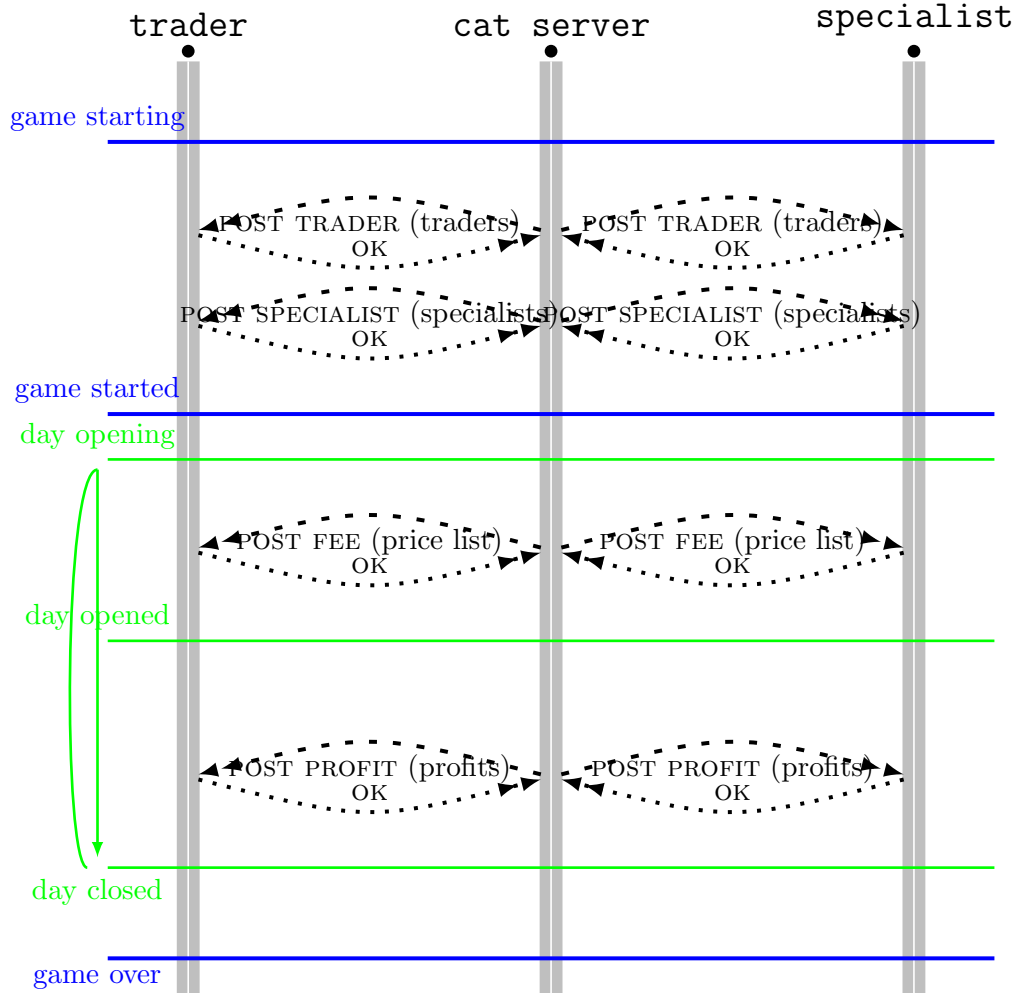


Figure 5: POST messages.

A positive response from the server should include a `Value` header presenting the price list with comma as separator and optionally the specialist ID.

EXAMPLE:

```
OK CRLF
Id: Specialist3 CRLF
```

```
Value: 2, 5, 4, 6, 0.2 CRLF
CRLF
```

A GET FEE request can only be made after a trading day is opened and before it is closed. The price list returned is the one the specialist of concern announced for the present trading day.

8.7 ASK

An ASK request from a seller to the server expresses an offer to sell through the specialist the seller is registered with. A `Value` header is required to hold the ask price.

EXAMPLE:

```
ASK CRLF
Value: 87 CRLF
CRLF
```

After a new ask arrives at the server,

- if the ask fails in validation, the server simply sends an `INVALID` response.

EXAMPLE:

```
INVALID CRLF
CRLF
```

- Otherwise, the server allocates a unique ID to the ask and forwards the ask to the specialist the seller is registered with.

EXAMPLE:

```
ASK CRLF
Id: ask2 CRLF
Value: 87 CRLF
CRLF
```

After an ASK message is received by a specialist, it is accepted or rejected according to the rules of the specialist. For example, if the specialist requires that every ask beat the standing ask, then an ask that is too high will be rejected.

- If accepted, an empty OK response is sent to the server. The server then notifies the seller and possibly other traders and specialists if they paid for information on shouts placed. To the seller, an OK response with an `Id` header is sent.

EXAMPLE:

```
OK CRLF
Id: ask2 CRLF
CRLF
```

To the information subscribers, a `POST` message is used with `ASK` in a `Type` header, the price in a `Value` header, and the ask ID, the seller ID, and the specialist ID in order in an `ID` header. The receivers then simply respond with an empty `OK` message to confirm.

EXAMPLE:

```
POST CRLF
Type: ASK CRLF
Id: ask2, seller0, specialist1 CRLF
Value: 87 CRLF
CRLF
```

- If rejected, an `INVALID` response should be created and sent to the server with optional `Type` and `Text` headers giving the type of reason and detailed description.

EXAMPLE:

```
INVALID CRLF
CRLF
```

The server then sends an `INVALID` response to the seller with `SPECIALIST` in a `Type` header. Since the ask is not accepted, only the seller is informed.

EXAMPLE:

```
INVALID CRLF
```

Type: SPECIALIST CRLF
CRLF

Please refer to Figure 6 for illustrations of how asks are processed in different scenarios.

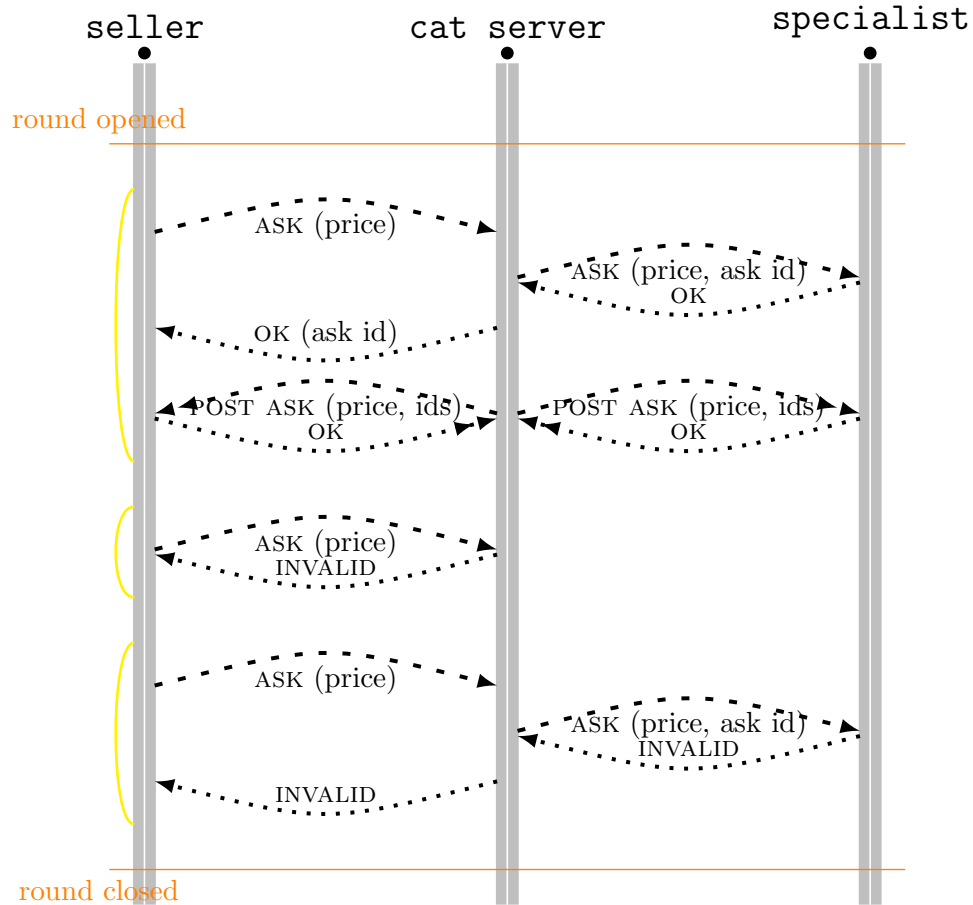


Figure 6: Dialogs regarding asks.

NOTE: A specialist does not know which seller made an ask according to the above protocol. This can be changed in later versions of CATP though.

8.8 BID

A BID sent from a buyer to the server expresses an offer to buy through a specialist. The processing of BID messages is similar to that of ASK messages.

Please refer to Figure 6 for illustrations of how asks are processed in different scenarios.

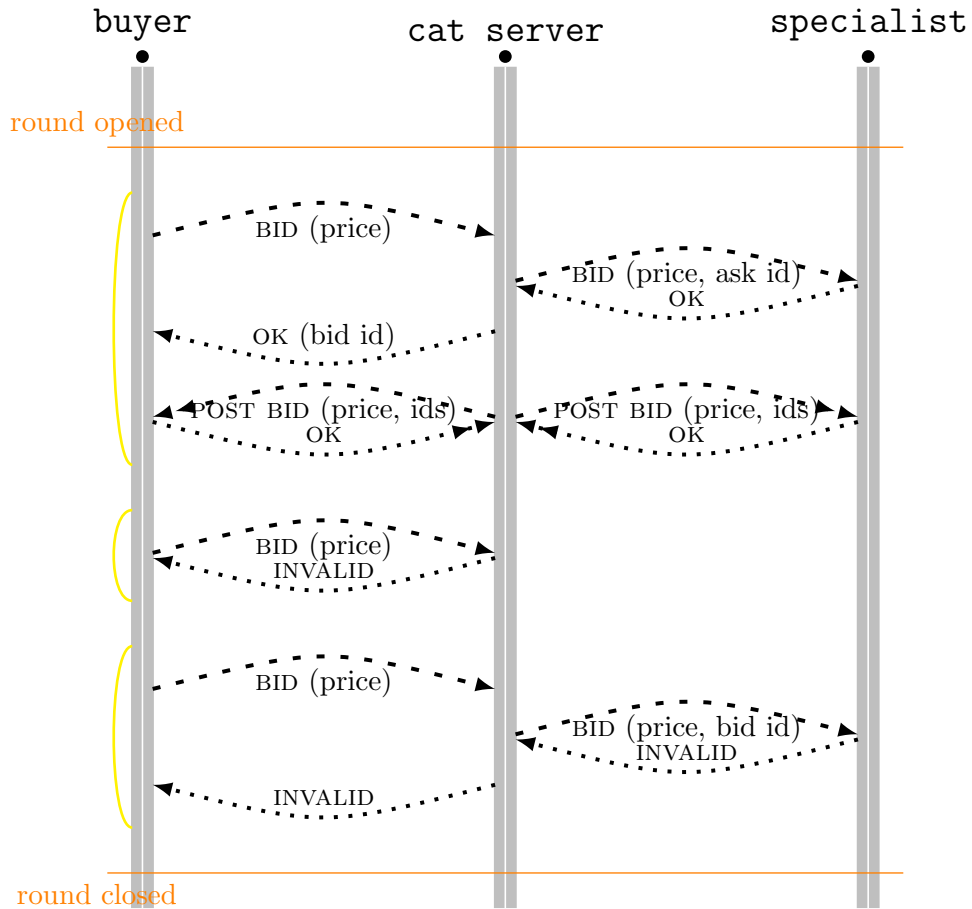


Figure 7: Dialogs regarding bids.

8.9 TRANSACTION

A **TRANSACTION** request, made by a specialist to the server, announces a transaction made by the specialist. An **Id** header should be used to provide the corresponding ask ID and bid ID in order and a **Value** header for the transaction price.

EXAMPLE:

```
TRANSACTION CRLF
Id: ask3, bid2 CRLF
Value: 90 CRLF
CRLF
```

The server then checks whether this transaction is valid or not.

- If the transaction is valid, the server should allocate a unique transaction ID for the transaction and respond with an **OK** message with the ID in an **Id** header.

EXAMPLE:

```
OK CRLF
Id: transaction2 CRLF
CRLF
```

The server further sends a **POST** message to both the buyer and the seller involved in the transaction as well as other traders who paid for the information of transaction. The message should include **TRANSACTION** in a **Type** header, the transaction ID, the ask ID, the bid ID, the specialist ID in order in an **Id** header, and the transaction price, ask price, and bid price in order in a **Value** header.

EXAMPLE:

```
POST CRLF
Type: TRANSACTION CRLF
Id: transaction2, ask3, bid2, specialist0 CRLF
Value: 90, 67.5, 100.3 CRLF
CRLF
```

Traders receiving this message successfully should respond with an empty **OK** message.

- If invalid, the server should send back an `INVALID` message optionally with optional `Type` and `Text` headers giving respectively the reason of the message being invalid and detailed description.

Figure 8 illustrates possible dialogs regarding transactions.

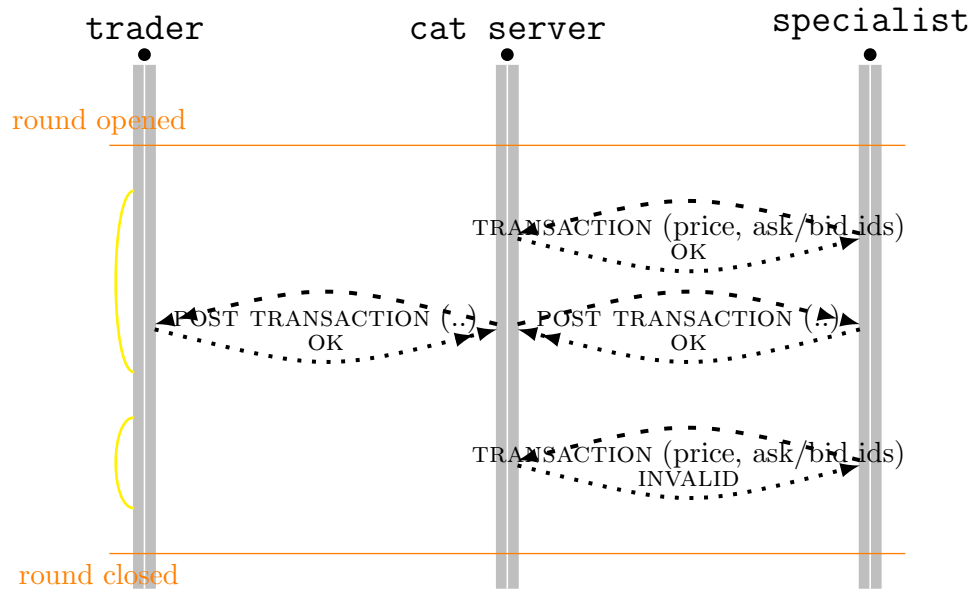


Figure 8: Dialogs regarding transactions.

NOTE: A transaction cannot be rejected by the traders involved in a transaction. Its validity is guaranteed by the CATP server. This makes the CAT markets different to some (for example that in the Santa Fe auction tournament [8]) which allows traders to decide whether to accept matching offers.

9 Responses

After a request message is received and interpreted, a CATP response message should be sent back.

The first line of a response message consists of the type of response, and an optional sender and receiver ID pair.

```

response-type = "OK"
               | "INVALID"
               | "ERROR"

```

9.1 OK

An OK response acknowledges the success of a request. For example, a seller agent may receive a response like the following:

```

OK CRLF
Id: ask3
CRLF

```

informing it that the ASK request is successful.

9.2 INVALID

An INVALID response indicates that the corresponding request is invalid, meaning the request is syntax-correct but it is not expected or logically sound — for example when a seller sends out a BID request, or a seller sends out an ASK request that violates some market rule (like having to beat the current quote).

The INVALID message may need a **Type** header to tells why the request is invalid. The value of a **Type** header can be:

- **WRONGTIME**: When a request is made by a client at a wrong time, for example an ASK request made after a trading round is closed and before the next opens, an INVALID response with a WRONGTIME type can be responded.

EXAMPLE:

```

INVALID CRLF
Type: WRONGTIME CRLF
CRLF

```

- **SPECIALIST**: When a shout is rejected by a specialist, the server sends an INVALID response to the trader with a SPECIALIST type. Please refer to the description on ASK requests in Section 8.7.

EXAMPLE:

```
INVALID CRLF
Type: SPECIALIST CRLF
CRLF
```

9.3 ERROR

An **ERROR** response notifies the sender of the corresponding request of occurrence of error. The **ERROR** message may need a **Type** header to tell what type of error it is and an optional **Text** header for additional description.

Types of error supported include:

- **REQUEST**: syntax error in request or failure in reading request

NOTE: It is up to the receiver whether to disconnect from the sender and try to connect again.

- **RESPONSE**: failure while trying to respond

EXAMPLE:

```
ERROR CRLF
Type: REQUEST CRLF
Text: syntax error in request CRLF
CRLF
```

10 Message Headers

This section gives detailed descriptions of CATP header fields, as well as links to related message types.

10.1 Time

A **Time** header field specifies the time when an event occurred. Only CATP requests from the server may use a **Time** header. It is required to include a **Time** header in **POST ASK**, **POST BID**, and **POST TRANSACTION** messages.

The content of a **Time** is a list of 3 numbers separated by commas, specifying respectively the day, the round, and the ticks into the round.

EXAMPLE:

```
Time: 2, 3, 156
```

10.2 Tag

A **Tag** header field carries a tag that tells whether the current message is sent on the current day or not. Sometimes, due to network delay, a message sent the previous day may arrive on the day next. The **Tag** header is used to detect obsolete messages.

The CATP server generates a tag when the following events occur:

- the game is starting;
- a day is opening; and
- the game is over.

And every message sent by the sever during the game includes the current tag in a **Tag** header.

Each client maintains a tag from the server and updates it when the above events are informed through **OPTIONS** messages. Clients use the tag to recognize obsolete messages.

10.3 Id

An **Id** header field provides one or more IDs, separated by commas if necessary. The number and semantics of IDs are determined by the context of the current message. See Sections [8.7](#), [8.8](#), [8.9](#), and [9.1](#).

10.4 Type

A **Type** header field gives the type of an entity. See Sections [8.1](#), [8.2](#), [8.5](#), [8.6](#), [9.2](#), and [9.3](#).

10.5 Value

A **Value** header field provides one or more numeric values. The number(s) and semantics of values depends upon the context of the current message. See Sections [8.7](#), [8.8](#), and [8.9](#).

10.6 Text

A **Text** header field carries a plain-text string providing additional description. See Sections [8.1](#), [9.2](#), and [9.3](#).

10.7 Version

A `Version` header field tells the version of CATP being supported. For now, it is used only when a CATP client checks in. See Section 8.1.

References

- [1] CAT Document 002 — JCAT: TAC/CAT Competition Platform.
- [2] TAC Market Design: Planning and Specification. Document confidential to MBC-CAT Development team.
- [3] Dave Cliff. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HP-97-91, Hewlett-Packard Research Laboratories, Bristol, England, 1997.
- [4] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behaviour*, 22:1–29, 1998.
- [5] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *The Journal of Political Economy*, 101(1):119–137, February 1993.
- [6] Hypertext Transfer Protocol – HTTP/1.1: Notational Conventions and Generic Grammar.
- [7] A. E. Roth and I. Erev. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, 8:164–212, 1995.
- [8] J. Rust, J. H. Miller, and R. Palmer. Behaviour of trading automata in a computerized double auction market. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 6, pages 155–199. Perseus Publishing, Cambridge, MA, 1993.