ACPC Protocol Specification, version 2.0.0, October 2010

This document describes the format of messages between the ACPC dealer program (server) and a player (client.) Communication is over TCP, with clients connecting to the server (the client will be given a numeric address) using the ports printed out by the server. All messages are terminated by a carriage return and a new line ('\r\n' or ASCII characters 13 and 10, respectively.) Any message from the server which starts with '#' or ';' is a comment or GUI command, and may be safely ignored.

The first message from a player must be a version string

```
<version> := 'VERSION:2.0.0\r\n'
```

After this, the server will repeatedly send messages to the client giving their view of the match state, including states where the client is not acting, and the final state when the game is over. If the client is acting, they will respond by returning a message with the same state, followed by an action.

```
<serverMessage> := <matchState> '\r\n'
<matchState> := 'MATCHSTATE:' <position> <handNumber> <betting> <cards>
<position> := <unsigned integer> ':'
<handNumber>:= <unsigned integer> ':'
<betting> := { <limitBetting> } { <nolimitBetting> } ':'
<limitBetting> := ( <round1LimitBetting>
        { | <round1LimitBetting> '/' <round2LimitBetting> … } )
<roundXLimitBetting> = <limitAction>*
<limitAction> := <fold> | <call> | <limitRaise>
<fold> := 'f'
<call> := 'c'
<limitRaise> := 'r'
<nolimitBetting> := ( <round1NolimitBetting>
        { | <round1NolimitBetting> '/' <round2NolimitBetting> … } )
<roundXNolimitBetting> := <noLimitAction>*
<noLimitAction> := <fold> | <call> | <nolimitRaise>
<nolmitRaise> := 'r' <nolimitRaiseSize>
<nolimitRaiseSize> := <unsigned integer>
<cards> := <holeCards> <boardCards>
<holeCards> := <player1Cards> '|' <player2Cards> { '|' <player3Cards> … }
<boardCards> := <round1BoardCards> { '/' <round2BoardCards> … }
<playerXCards> := '' | <card> { <card> … }
<roundXBoardCards> := { <card> … }
<card> := <rank> <suit>
<rank> := '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | 'T' | 'J' | 'Q' | 'K' | 'A'
<suit> := 's' | 'h' | 'd' | 'c'
<clientResponse> := <matchState> ':'  { <limitAction> } { <noLimitAction> } '\r\n'
```

Parts of the specification in brackets indicate game specific definitions. For example, in Texas Hold'em, there are four rounds with zero, three, one, and one board cards on the rounds respectively, so the actual definition would be

```
<round1BoardCards> := ''
<round2BoardCards> := <card> <card> <card>
<round3BoardCards> := <card>
<round4BoardCards> := <card>
```

The <position> field tells the client their position relative to the dealer button. A value of 0 indicates that for the current hand, the client is the first player after the button (the small blind in ring games, or the big blind in reverse-blind heads-up games.)

The <handNumber> is simply an identifier for the current hand. Clients should make no assumption about these values, other than that it will be unique across hands within a match, and that it will not change within a single hand.

<betting> is a list of actions taken by all players. There is no distinction made between calling and checking, or betting and raising. In no-limit betting strings, the raise action includes a size, which indicates the total number of chips the player will have put into the pot after raising (ie the value they are raising to, not the value they are raising by.) Valid betting strings will be described in a later section.

<cards> is the complete set of cards visible to the player, given their <position>. <holecards> describes the view of the private cards, and the number of <playerCards> sections is determined by the game being player. For example, heads-up games will have two sections, 3 player ring games will have 3 sections, and so on. Each <playerCard> section will either be an empty string, indicating that the player does not know what those cards are, or a number of <card> strings determined by the game. A <card> is simply two characters, one for the rank and one for the suit. The <boardCards> description will also have a number of sections, one for each round, up to the current round as indicated by the <betting>. The number of <cards> in each section is fixed, and determined by the game. Note that if it ever becomes desirable to play a game with board cards in the first round, the protocol should probably be changed again to add a separator between the hole cards and board cards.

A valid <betting> string consists of a sequence of valid actions for successive acting players. Calling is always valid. Folding is only valid if the acting player would need to add money to the pot to call. To describe raises, it is worth making a distinction between raising by and raising to. Unless this document specifically mention otherwise, when it speaks of a raise size, it is talking about a raise to a value, using the term to mean the total number of chips the player will have put into the pot, including chips added in previous rounds. In contrast, raising by a value is adding that number of chips to the pot after calling the current bet.

A raise is valid if a) it raises by at least one chip, b) the player has sufficient money in their stack to raise to the value, and c) the raise would put the player all-in (they have spent all their chips) or the amount they are raising by is at least as large as the big blind and the raise-by amount for any other raise in the round. This description properly handles some odd cases where some players may be all-in. Informally, at the beginning of a round a player may bet as low as the big blind, and each subsequent raise must increase the bet by at least as much as the last raise. In limit betting games, there may also be a limit on the number of raises in a round, and any raise action over this limit is not valid.

An active player is one that has not folded, and is not all-in. Label the players from 0 to *N*-1, where *N* is the number of players in the game. After an action by player *p*, the next acting player is *p'* for the minimum *d*>0 such that *p'* is active and *p'* =(*p*+*d*) modulo *N*. That is, the next player around the table who has at least two valid actions. The first player in a round is specified by the game.

A betting round ends when all active players remaining have either called, or were the player which initiated the current bet. The game is over if the all betting rounds have finished, or if there are not at least two active players left in the game. If all players but one have folded, the remaining player wins the entire pot, and does not show other players their cards. Otherwise, the game has ended in a showdown and all non-folding players show everyone their cards.

Values in a showdown are determined by the rank of a players best hand, constructed from their hole cards and the public board cards. Folding players have a lower rank than any non-folding player. Every pot of money is split evenly between all participating players with the highest rank. There is a separate pot for every distinct amount *j* of money spent by a player (whether they fold or not), and a player is participating in the pot if they spent at least that much money (again, whether they fold or not.) The size of the pot is equal to the number of participating players multiplied by (*j* - the size of the next smallest amount spent by a player).

Changes from Version 1

There a number of changes from the first version of the protocol. Ring games are now documented. No-limit games no longer give a value for a call, and there are no blinds in the string. The question of whether all-in players got to act was previously left unspecified, and different implementations had different behaviour. Folding is no longer allowed if the player could have called for free. Finally, this updated description also covers the protocol for no-limit ring games, in the event that this game is added to the competition.

It is also strongly suggested to the chair that on a failed responses from a client, whether the message was malformed, the action was invalid, or the player did not respond in time, the entire match should be ended. If the problem can not be fixed easily by a very short interaction with the team, the program should be disqualified. The previous choice of having the player call in some cases, or fold all remaining hands led to some situations where the results of the competition depended on the chair's decision on exactly how to handle incorrect behaviour. Teams with buggy programs also took up large amounts of the chair's time. As long as players are given sufficient time for testing, using the exact same code used for the competition, it is not unreasonable to expect players to be able to run without producing error messages or warnings.

Examples

The rest of this document is examples of messages to and from a client in various games. Messages from the server are prefaced with S->. Responses from the client are prefaced with <-C.

Two player limit Texas Hold'em

```
S-> MATCHSTATE:0:0::TdAs|
S-> MATCHSTATE:0:0:r:TdAs|
<-C MATCHSTATE:0:0:r:TdAs|:r
S-> MATCHSTATE:0:0:rr:TdAs|
S-> MATCHSTATE:0:0:rrc/:TdAs|/2c8c3h
<-C MATCHSTATE:0:0:rrc/:TdAs|/2c8c3h:r
S-> MATCHSTATE:0:0:rrc/r:TdAs|/2c8c3h
S-> MATCHSTATE:0:0:rrc/rc/:TdAs|/2c8c3h/9c
<-C MATCHSTATE:0:0:rrc/rc/:TdAs|/2c8c3h/9c:c
S-> MATCHSTATE:0:0:rrc/rc/c:TdAs|/2c8c3h/9c
S-> MATCHSTATE:0:0:rrc/rc/cr:TdAs|/2c8c3h/9c
<-C MATCHSTATE:0:0:rrc/rc/cr:TdAs|/2c8c3h/9c:c
S-> MATCHSTATE:0:0:rrc/rc/crc/:TdAs|/2c8c3h/9c/Kh
<-C MATCHSTATE:0:0:rrc/rc/crc/:TdAs|/2c8c3h/9c/Kh:c
S-> MATCHSTATE:0:0:rrc/rc/crc/c:TdAs|/2c8c3h/9c/Kh
S-> MATCHSTATE:0:0:rrc/rc/crc/cr:TdAs|/2c8c3h/9c/Kh
<-C MATCHSTATE:0:0:rrc/rc/crc/cr:TdAs|/2c8c3h/9c/Kh:c
S-> MATCHSTATE:0:0:rrc/rc/crc/crc:TdAs|8hTc/2c8c3h/9c/Kh
S-> MATCHSTATE:1:1::|Qd7c
<-C MATCHSTATE:1:1::|Qd7c:r
S-> MATCHSTATE:1:1:r:|Qd7c
S-> MATCHSTATE:1:1:rr:|Qd7c
<-C MATCHSTATE:1:1:rr:|Qd7c:c
S-> MATCHSTATE:1:1:rrc/:|Qd7c/2h8h5c
S-> MATCHSTATE:1:1:rrc/r:|Qd7c/2h8h5c
<-C MATCHSTATE:1:1:rrc/r:|Qd7c/2h8h5c:c
S-> MATCHSTATE:1:1:rrc/rc/:|Qd7c/2h8h5c/Th
S-> MATCHSTATE:1:1:rrc/rc/r:|Qd7c/2h8h5c/Th
<-C MATCHSTATE:1:1:rrc/rc/r:|Qd7c/2h8h5c/Th:f
S-> MATCHSTATE:1:1:rrc/rc/rf:|Qd7c/2h8h5c/Th
S-> MATCHSTATE:0:2::9d7s|
S-> MATCHSTATE:0:2:r:9d7s|
<-C MATCHSTATE:0:2:r:9d7s|:c
S-> MATCHSTATE:0:2:rc/:9d7s|/5d2cJc
<-C MATCHSTATE:0:2:rc/:9d7s|/5d2cJc:c
S-> MATCHSTATE:0:2:rc/c:9d7s|/5d2cJc
S-> MATCHSTATE:0:2:rc/cc/:9d7s|/5d2cJc/3d
<-C MATCHSTATE:0:2:rc/cc/:9d7s|/5d2cJc/3d:c
S-> MATCHSTATE:0:2:rc/cc/c:9d7s|/5d2cJc/3d
S-> MATCHSTATE:0:2:rc/cc/cr:9d7s|/5d2cJc/3d
<-C MATCHSTATE:0:2:rc/cc/cr:9d7s|/5d2cJc/3d:f
S-> MATCHSTATE:0:2:rc/cc/crf:9d7s|/5d2cJc/3d
```

Two player no-limit Texas Hold'em

```
S-> MATCHSTATE:0:30::9s8h|
S-> MATCHSTATE:0:30:c:9s8h|
<-C MATCHSTATE:0:30:c:9s8h|:c
S-> MATCHSTATE:0:30:cc/:9s8h|/8c8d5c
<-C MATCHSTATE:0:30:cc/:9s8h|/8c8d5c:r250
S-> MATCHSTATE:0:30:cc/r250:9s8h|/8c8d5c
S-> MATCHSTATE:0:30:cc/r250c/:9s8h|/8c8d5c/6s
<-C MATCHSTATE:0:30:cc/r250c/:9s8h|/8c8d5c/6s:r500
S-> MATCHSTATE:0:30:cc/r250c/r500:9s8h|/8c8d5c/6s
S-> MATCHSTATE:0:30:cc/r250c/r500c/:9s8h|/8c8d5c/6s/2d
<-C MATCHSTATE:0:30:cc/r250c/r500c/:9s8h|/8c8d5c/6s/2d:r1250
S-> MATCHSTATE:0:30:cc/r250c/r500c/r1250:9s8h|/8c8d5c/6s/2d
S-> MATCHSTATE:0:30:cc/r250c/r500c/r1250c:9s8h|9c6h/8c8d5c/6s/2d
S-> MATCHSTATE:1:31::|JdTc
<-C MATCHSTATE:1:31::|JdTc:r300
S-> MATCHSTATE:1:31:r300:|JdTc
S-> MATCHSTATE:1:31:r300r900:|JdTc
<-C MATCHSTATE:1:31:r300r900:|JdTc:c
S-> MATCHSTATE:1:31:r300r900c/:|JdTc/6dJc9c
S-> MATCHSTATE:1:31:r300r900c/r1800:|JdTc/6dJc9c
<-C MATCHSTATE:1:31:r300r900c/r1800:|JdTc/6dJc9c:r3600
S-> MATCHSTATE:1:31:r300r900c/r1800r3600:|JdTc/6dJc9c
S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000:|JdTc/6dJc9c
<-C MATCHSTATE:1:31:r300r900c/r1800r3600r9000:|JdTc/6dJc9c:c
S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/:|JdTc/6dJc9c/Kh
S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000:|JdTc/6dJc9c/Kh
<-C MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000:|JdTc/6dJc9c/Kh:c
S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000c/:KsJs|JdTc/6dJc9c/Kh/Qc
```

Three player limit Texas Hold'em

```
S-> MATCHSTATE:2:55::||AsTs
<-C MATCHSTATE:2:55::||AsTs:r
S-> MATCHSTATE:2:55:r:||AsTs
S-> MATCHSTATE:2:55:rc:||AsTs
S-> MATCHSTATE:2:55:rcc/:||AsTs/4cJh8h
S-> MATCHSTATE:2:55:rcc/r:||AsTs/4cJh8h
S-> MATCHSTATE:2:55:rcc/rf:||AsTs/4cJh8h
<-C MATCHSTATE:2:55:rcc/rf:||AsTs/4cJh8h:c
S-> MATCHSTATE:2:55:rcc/rfc/:||AsTs/4cJh8h/Kd
S-> MATCHSTATE:2:55:rcc/rfc/r:||AsTs/4cJh8h/Kd
<-C MATCHSTATE:2:55:rcc/rfc/r:||AsTs/4cJh8h/Kd:c
S-> MATCHSTATE:2:55:rcc/rfc/rc/:||AsTs/4cJh8h/Kd/8c
S-> MATCHSTATE:2:55:rcc/rfc/rc/r:||AsTs/4cJh8h/Kd/8c
<-C MATCHSTATE:2:55:rcc/rfc/rc/r:||AsTs/4cJh8h/Kd/8c:f
S-> MATCHSTATE:2:55:rcc/rfc/rc/rf:||AsTs/4cJh8h/Kd/8c
S-> MATCHSTATE:0:90::Ad6h||
S-> MATCHSTATE:0:90:c:Ad6h||
<-C MATCHSTATE:0:90:c:Ad6h||:r
S-> MATCHSTATE:0:90:cr:Ad6h||
S-> MATCHSTATE:0:90:crf:Ad6h||
S-> MATCHSTATE:0:90:crfc/:Ad6h||/TsKd7h
<-C MATCHSTATE:0:90:crfc/:Ad6h||/TsKd7h:r
S-> MATCHSTATE:0:90:crfc/r:Ad6h||/TsKd7h
S-> MATCHSTATE:0:90:crfc/rc/:Ad6h||/TsKd7h/Kh
<-C MATCHSTATE:0:90:crfc/rc/:Ad6h||/TsKd7h/Kh:r
S-> MATCHSTATE:0:90:crfc/rc/r:Ad6h||/TsKd7h/Kh
S-> MATCHSTATE:0:90:crfc/rc/rc/:Ad6h||/TsKd7h/Kh/6d
<-C MATCHSTATE:0:90:crfc/rc/rc/:Ad6h||/TsKd7h/Kh/6d:r
S-> MATCHSTATE:0:90:crfc/rc/rc/r:Ad6h||/TsKd7h/Kh/6d
S-> MATCHSTATE:0:90:crfc/rc/rc/rc:Ad6h||Td2h/TsKd7h/Kh/6d
```