**CS 544 Computer Networks**

**PROTOCOL DESIGN**

**GROUP 10**

* Jennifer Lautenschlager
* Constantine Lazarakis
* Duc Anh Nguyen
* Carol Greco

# 

# 

Contents

[0. Preface 6](#_Toc355972531)

[0.1. About This Protocol 6](#_Toc355972532)

[0.2. Authors’ Note 6](#_Toc355972533)

[0.3. Revision History 6](#_Toc355972534)

[1. Service Description 7](#_Toc355972535)

[1.1.1 Conditions for Play 7](#_Toc355972536)

[1.1.2 Implementing the Game 7](#_Toc355972537)

[1.1.3 Text Card Descriptions 8](#_Toc355972538)

[1.1.4 Looking ahead to version 2.0 9](#_Toc355972539)

[1.2 Commands and Responses 9](#_Toc355972540)

[1.2.1 Character Set 10](#_Toc355972541)

[1.2.2 Commands 10](#_Toc355972542)

[1.2.3 Command Variants 10](#_Toc355972543)

[1.2.4 Keywords 10](#_Toc355972544)

[1.2.5 Responses 10](#_Toc355972545)

[1.2.6 Timers 11](#_Toc355972546)

[1.3 Response Codes 11](#_Toc355972547)

[1.3.1 Generic response code 12](#_Toc355972548)

[1.4 Capabilities and Extensions 13](#_Toc355972549)

[1.4.1 Capabilities Descriptions 13](#_Toc355972550)

[1.4.2 Standard Capabilities 14](#_Toc355972551)

[1.4.3 Extensions 15](#_Toc355972552)

[1.5 Mandatory and Optional Commands 15](#_Toc355972553)

[1.6 Game Session Administrative Commands 15](#_Toc355972554)

[1.7 Information Commands 15](#_Toc355972555)

[1.8 Private Commands 16](#_Toc355972556)

[2. Message Definition 16](#_Toc355972557)

[2.1 USERNAME 16](#_Toc355972558)

[2.1.1. Usage 16](#_Toc355972559)

[2.1.2 Examples 17](#_Toc355972560)

[2.2 PASSWORD COMMAND 17](#_Toc355972561)

[2.2.1 Syntax 17](#_Toc355972562)

[2.2.2. Usage 18](#_Toc355972563)

[2.2.3 Examples 18](#_Toc355972564)

[2.3 LISTGAMES COMMAND 19](#_Toc355972565)

[2.3.1 Syntax: 19](#_Toc355972566)

[2.3.2. Usage 19](#_Toc355972567)

[2.3.3. Game Descriptor Format 19](#_Toc355972568)

[2.3.4. Example 21](#_Toc355972569)

[2.4 QUIT COMMAND 22](#_Toc355972570)

[2.4.1 Syntax: Client Issues Request 22](#_Toc355972571)

[2.4.2 Usage 22](#_Toc355972572)

[2.4.3 Example 22](#_Toc355972573)

[2.5 ACCOUNT COMMAND 23](#_Toc355972574)

[2.5.1 Syntax: 23](#_Toc355972575)

[2.5.2 Usage 23](#_Toc355972576)

[2.5.3. Examples 23](#_Toc355972577)

[2.6 JOINSESSION COMMAND 23](#_Toc355972578)

[2.6.1 Syntax: 24](#_Toc355972579)

[2.6.2. Usage 24](#_Toc355972580)

[2.6.3. Examples 25](#_Toc355972581)

[2.7 LEAVESESSION COMMAND 25](#_Toc355972582)

[2.7.1 Syntax: 25](#_Toc355972583)

[2.7.2. Usage 26](#_Toc355972584)

[2.7.3. Example 26](#_Toc355972585)

[2.8 BET COMMAND 27](#_Toc355972586)

[2.8.1 Syntax 27](#_Toc355972587)

[2.8.2. Usage 27](#_Toc355972588)

[2.8.3. Examples 28](#_Toc355972589)

[2.9 HIT COMMAND 28](#_Toc355972590)

[2.9.1 Syntax 28](#_Toc355972591)

[2.9.2. Usage 28](#_Toc355972592)

[2.9.3. Examples 29](#_Toc355972593)

[2.10 STAND COMMAND 29](#_Toc355972594)

[2.10.1 Syntax 29](#_Toc355972595)

[2.10.2. Usage 29](#_Toc355972596)

[2.10.3. Example 30](#_Toc355972597)

[2.11 CAPABILITIES COMMAND 30](#_Toc355972598)

[2.11.1 Syntax: 30](#_Toc355972599)

[2.11.2. Usage 30](#_Toc355972600)

[2.11.3. Examples 31](#_Toc355972601)

[2.12 VERSION COMMAND 31](#_Toc355972602)

[2.12.1 Syntax: 32](#_Toc355972603)

[2.12.2. Usage 32](#_Toc355972604)

[2.12.3. Example 32](#_Toc355972605)

[2.13. Summary of response codes 32](#_Toc355972606)

[2.14. Timeouts 32](#_Toc355972607)

[2.14.1. Usage 33](#_Toc355972608)

[2.14.2. Example 33](#_Toc355972609)

[2.14.3 Summary of Timeouts per State 33](#_Toc355972610)

[2.15. Game State Messages 33](#_Toc355972611)

[2.15.1 Examples 36](#_Toc355972612)

[3. DFA 37](#_Toc355972613)

[3.1. Use cases considered in DFA construction 39](#_Toc355972614)

[4. Extensibility 39](#_Toc355972615)

[4.1 Version 39](#_Toc355972616)

[4.2 Private Commands Description 39](#_Toc355972617)

[4.2.1 Example 40](#_Toc355972618)

[4.3 Games-Hosted Description 40](#_Toc355972619)

[4.3.1 Example 40](#_Toc355972620)

[5. Security Implications 40](#_Toc355972621)

[Appendix A. USE CASES 42](#_Toc355972622)

[Use Case 1: Log-in 42](#_Toc355972623)

[Use Case 2: Register user 43](#_Toc355972624)

[Use Case 3: Join Group 44](#_Toc355972625)

[Use Case 4: Bet 45](#_Toc355972626)

[Use Case 5: Increase Bet 46](#_Toc355972627)

[Use Case 6: Play - Player blackjacks after HIT 47](#_Toc355972628)

[Use Case 7: Play game - Times Out 48](#_Toc355972629)

[Use Case 8: Play game - User has Blackjack - first 2 cards dealt 49](#_Toc355972630)

[Use Case 9: Place Bet 50](#_Toc355972631)

[Use Case 10: Update User status 51](#_Toc355972632)

[APPENDIX B. RESPONSE CODES DEFINED 52](#_Toc355972633)

[APPENDIX C. TERMINOLOGY DEFINED 58](#_Toc355972634)

[1.1 BASE ACTION PLAYS 58](#_Toc355972635)

[1.2 OPTIONAL PROCEDURES 58](#_Toc355972636)

[1.3 BETTING OPTIONS 59](#_Toc355972637)

[1.4 OTHER TERMINOLOGY 60](#_Toc355972638)

[APPENDIX D: REFERENCES 61](#_Toc355972639)

# 0. Preface

There is much speculation as to the origin of Blackjack, though historians believe it to have evolved from a Spanish card game known as “*Trente un” (thirty-one) and the French card game of “Vingt-un” (twenty-one)* The former mentioned by Spanish author Miguel de Cervantes (Don Quixote) in one of his mid-15th Century texts.[1] Though the earlier games were somewhat different in structure, the general idea was the same – to draw a group of cards whose value was as close to or equal to a determined numeric value without going over the point limit.

The actual term “Blackjack” is credited to early gambling houses in the United States. To encourage gambling during difficult times (circa WW1) a very attractive payout was offered; a ten to one bonus for a player who received an Ace of Spades and a Black Jack (Jack with a club or spade suit). Though that offer has long since expired and the game has changed from its origins, the name of Blackjack has stayed with the game

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## 0.1. About This Protocol

This protocol, entitled Blackjack Protocol (BJP) represents an Internet standards track protocol designed for a client-server implementation of playing the traditional card game known as Blackjack. The information contained within this document gives the initial concepts, procedures and framework to put into effect the playing of the game Blackjack, version 1.0 a protocol design project by Team #10/Spring ’13/CS544/Drexel U.

## 0.2. Authors’ Note

This protocol outline and many of the procedure paradigms have been referenced from the Network News Transfer Protocol, RFC #3977 [2]. The BJP commands are referenced in a manner as per recommended Best Practices, Keywords for Use in RFCS to Indicate Requirements Levels, sRFC #2119 [3].

## 0.3. Revision History

This document was initiated on May 5, 2013.

Most recent revision to this Protocol Version (1.0) is dated: May 10, 2013

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 1. Service Description

This section specifies the structure and basic format of the key components comprising Blackjack Protocol (BJP) version 1.0.

1.1. Overview - Problem Statement

At this time the Blackjack Protocol version, 1.0 features only a limited amount of methods, controls and open options that allow for the user to play a base version of the game. The game at this version (1.0) is played strictly in a textual format.

### 1.1.1 Conditions for Play

To be able to participate in a Blackjack game, the user must be a registered member of the Blackjack Game community. That is, the user must have successfully completed the registration process thereby setting up an account with an appropriate USERNAME and PASSWORD according to the conditions defined in this documentation (Sections 2.1 and 2.2). The user must be “logged in” to their account to access the features and available games.

The users account must be in good standing and contain the appropriate “funds” necessary to join/play a game.

User may join a game that is already in session, or initiate their own game at start. Dependent upon server capabilities, Blackjack can be played as Many vs House (multiplayer) or Individual vs House (single player).

### 1.1.2 Implementing the Game

*The information specified in this section gives a base description as to the playing of an elementary game of Blackjack and does not go into detail about special options that may be offered in this version of the BJP 1.0 or successive versions that follow. It has been written merely as a guide for the purpose of investigating the flow of information and as a check for possible situations in the same.*

Upon successful verification of USERNAME and PASSWORD, the user enters the Not in Game State where they are presented with options (section 2.1.4) representing type of available Blackjack games, requirements of said game (e.g. betting scale) and uses command LISTGAMES to view games that are presently in play and/or check which version of Blackjack they are playing. The LISTGAMES and VERSION commands can only be accessed in the Not in Game State and do not themselves change the state of the system.

Player chooses a game from available game list and enters the command to JOINSESSION to become part of game in session. Server checks user’s account to validate funding for the game and sets aside the appropriate funds until the results of the game are calculated. If funds are available, user is allowed to continue to the actual playing of the game. If funds are insufficient, user receives an error message from the server. User must now request through the BET command that the initial funds to participate in the game are set aside. Server updates the users account information, depreciating the users account by required amount.

Still in the JOINSESSION state, the dealer (system) issues two (2) text based cards to each player (defined in section 2.1.5) one “face down” - visible only to the individual the card has been dealt, and the other “face up” - visible to all participants in the game. Status of other players in the game session is visible to each user through a status box.

Each player is given a turn in which they decide the appropriate action for the cards they have been dealt. Player in turn has the option to stay with the hand that they have been dealt using the STAND command or receive another card to add to the existing total by using the HIT command. Request is sent to the server as to how the game is to proceed. If the user decides to receive a HIT, the server deals another card to the player. The server also sends a response code to the host indicating the players score status; under 21, exactly 21 or over 21 (bust), whereupon the player’s status is updated. If the situation calls for another response (under 21) the player again responds to the updated hand with either the HIT or STAND command. Play continues until the player decides to STAND with their hand, busts or quits. A player may leave the game at any time using the QUIT command.

After all players have reached a stand, all player’s cards are revealed in the status box, with the dealer (system) cards the last to be disclosed. System calculates scores of all the players, the winner is determined and appropriate funds are transferred to the winners account or “house” account. System shows all the players the final game scores and declares the player who is the winner. PLayer then may decide to continue playing Blackjack or exit the game using the QUIT command.

### 1.1.3 Text Card Descriptions

Text based cards are defined in the Blackjack Protocol version 1.0 as follows:

**The four suits of cards (Hearts, Diamonds, Clubs, Spades)**

* + Hearts - Uppercase “H”
  + Diamonds - Uppercase “D”
  + Clubs - Uppercase “C”
  + Spades - Uppercase “S”

Cards 1 - 10 (inclusive) have same rank.

An Ace can have two possible values - 1 or 11. An ace is assumed to be a value of 11 unless such value would place player over the desired 21 points. (See Soft and Hard hand in Appendix C)

* + A Jack has a numeric value of 10 and is denoted using an Uppercase “J”
  + A Queen has a numeric value of 10 and is denoted using an Uppercase “Q”
  + A King has a numeric value of 10 and is denoted using an Uppercase “K”

*The BJP v 1.0 does not make use of Jokers or other wild cards at this time.*

**Textual Card Examples:**

* + King of Diamonds = KD
  + Ace of Spades = AS
  + Five of Clubs = 5C
  + Ten of Hearts = 10C

### 1.1.4 Looking ahead to version 2.0

* + Messages to manage User account, e.g. add funds from outside source
  + Introduction of new versions of Blackjack, like “*five -card Charlie”*
  + Extension of the list of game options, with options such as Double Down, HIT Twice, Split Hand or the increasing of bets within the game.
  + Initiating the formation of a Blackjack group/team.
  + New methods of user-defined game competition, e.g. wildcards
  + Addition of HELP and SEARCH Commands

## 1.2 Commands and Responses

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This BJP version -version 1.0 - operates over TCP. When the connection is established, the BJP host MUST send a greeting. After the greeting, the client host and the server host exchange commands and responses (respectively) until the connection is closed or aborted. More specifically, the server host starts the BJP service by listening on a TCP port. When a client host wishes to use the service, it MUST establish a TCP connection with the server host, by connecting to that host on the same port on which the server is listening.

### 1.2.1 Character Set

The character set for all BJP commands is UTF-8 [1].

### 1.2.2 Commands

Commands in BJP MUST consist of a keyword, which MAY be followed by one or more arguments. All commands MUST use the delineator of Carriage Return followed by a Line Feed; in this document, this is represented as <CRLF>. A <CRLF> pair MUST terminate all commands. Multiple commands MUST NOT be on the same line. Unless otherwise noted elsewhere in this document, arguments SHOULD consist of printable US-ASCII characters. Keywords and arguments MUST each be separated by one or more space or TAB characters. Command lines MUST NOT exceed 1024 characters, which includes the terminating pair <CRLF>. The arguments MUST NOT exceed 497 octets. A server MAY relax these limits for commands defined in an extension. Commands cannot be pipelined, but must be given sequentially in request-response pairs between client and server.

### 1.2.3 Command Variants

Commands may have variants; in this situation, they use a second keyword immediately after the first, to indicate which variant is required. A command with two keywords e.g. the (hypothetical) HIT TWICE command should be read as shorthand for “the TWICE variant of the HIT command”. There are no such commands in this specification.

### 1.2.4 Keywords

Keywords are case-insensitive; the case of keywords for commands MUST be ignored by the server.

### 1.2.5 Responses

Each response MUST start with a three-digit response code that is sufficient to distinguish all responses. Certain valid responses are defined to be multi-line; for all others, the response is contained in a single line response. The initial line for the response MUST NOT exceed 1024 characters, which includes the response code and the terminating <CRLF> pair. An extension MAY specify a greater maximum for commands that it defines, but not for any other command. Single line responses consist of an initial line only. Multi-line responses consist of an initial line followed by a multi-line data block.

Example:

### Response message text <CRLF>

First line of multiline response <CRLF>

Second line of multiline response <CRLF>

<CRLF>

### 1.2.6 Timers

BJP version 1.0 has been specified to function on top of reliable transport protocol such as the TCP protocol. To perform their operations smoothly, most TCP implementations use at least four timers: retransmission, persistence, keepalive, and TIME-WAIT.

In the case that BJP version 1.0 is used on top of a TCP protocol implementation which does not support Keepalive or TIME-WAIT the BJP server MUST have at least one inactivity autologout timer for the login state. This timer is further described in Section 2.14: Timeouts.

## 1.3 Response Codes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Response code MUST be a numeric status indicator sent from the server to report the status of response to the last command received from client. ALL responses sent from server SHALL start with a response code. Response codes SHALL have the form of three digits. The first digit SHALL indicate success/failure/progress of the command and MUST conform to the following rules:

1xx - Informative message

2xx - Command completed

3xx - Command OK so far; send the rest of it

4xx - Command was syntactically correct but failed for some reasons

5xx - Command unknown, unsupported, unavailable, or syntax error

Additionally, a special 6xx pattern is reserved for game state messages that are broadcast from the server to clients in a game session. These messages are described in Section 2.15.

The second digit SHALL indicate the category of the response:

x0x - Administrative command group

x1x - Not-in-game command group

x2x - In game command group

In some of the responses, the response code MAY be followed by arguments in the same line; for example: the number of games for the command LISTGAMES. The arguments SHALL be separated from themselves and from the response code by one single space. The first line that starts with the response code MUST have less than 512 octets. There MAY be information following the response code or the last argument if exists; however the client SHOULD only use the response code as a key to perceive the command state. Using the response code, the client SHOULD know if there are multiple lines of block of data to come next.

For each command, the server SHALL send a generic response code or one of the response codes specifically listed in the command’s description. Extensions of this protocol CAN change the list of commands by adding commands, however the extensions CAN NOT change the list of possible response codes for any commands, so a client implemented to this protocol will only receive the response codes listed.

If in some cases, when the client receives and unrecognizable response code, it SHOULD use the first digit to determine the status.

### 1.3.1 Generic response code

The server MUST send a generic response code if other specifically listed response codes in the command’s description are not applicable. In other words, the generic response codes may reply to any commands if the situation forces them to, and the client SHOULD prepare for these codes.

**400: Internal error**. The server receives a valid command, however the server CANNOT move to the next state due to internal errors, for example: running out of space to store user’s information. In receiving of this command, the client SHOULD NOT resend the next commands immediately, but it SHOULD send the next commands in exponentially increasing delay, or when user requests the client to.

**401: Authentication error**. The server receives a valid command, however the client does not have permission to execute the command. In this case, the client SHOULD reconnect and authenticate using authentication commands (USERNAME/PASSWORD).

**500: Unknown command**. The server receives an unknown or unsupported command.

**501: Unsupported command.** The server receives a valid optional command, but does not provide any services for the command.

**502: Syntax error**. The server receives a supported command, however the syntax of the command is invalid.

## 1.4 Capabilities and Extensions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The BJP specification allows variation of facilities provided. This because servers may provide differing facilities and because this specification allows servers to provide extensions. A set of facilities that are related are called a “Capability”.

This specification provides a way to determine what capabilities are available; it includes a list of standard capabilities, either required or optional, and a mechanism for defining new capabilities.

### 1.4.1 Capabilities Descriptions

A client may determine the capabilities of the server by using the CAPABILITIES command. This returns a capability list, which is the list of capability lines. Each line describes one available capability.

Each capability line consists of one or more tokens which MUST be separated by one or more space or TAB characters. A token is a string of at least one printable UTF-8 character. Tokens are case-insensitive. Each capability line consists of the following:

* + The capability label, which is a keyword indicating the capability. A capability label may be defined by this specification or a successor, or may be an extension.
  + The label is followed by zero or more tokens, which are arguments of the capability. The form and meaning of these tokens are specific to each capability.

The server MUST ensure the list of capabilities accurately reflects the capabilities --including extensions-- currently available.

### 1.4.2 Standard Capabilities

The following capabilities are defined by this specification:

VERSION

This capability MUST be advertised by all servers and MUST be the first capability in the capability list. It indicates the version of BJP that the server supports. There MUST be at least one argument; each argument is a decimal number and MUST NOT have a leading zero. Version numbers are assigned only in RFCs -created by Team #10/Spring ’13/CS544/Drexel U. that update or replace this specification. Servers MUST NOT create their own version numbers. (See also Section 2.12, VERSION Command).

Section 4.1 further specifies versioning as a means of providing protocol extensibility.

USERNAME

This capability indicates the server implements the USERNAME command.

PASSWORD

This capability indicates the server implements the PASSWORD command.

CAPABILITIES

This capability indicates the server implements the CAPABILITIES command.

LISTGAMES

This capability indicates the server implements the LISTGAMES command.

ACCOUNT

This capability indicates the server implements the ACCOUNT command.

JOINSESSION

This capability indicates the server implements the JOINSESSION command.

LEAVESESSION

This capability indicates the server implements the LEAVESESSION command.

QUIT

This capability indicates the server implements the QUIT command.

BET

This capability indicates the server implements the BET command.

HIT

This capability indicates the server implements the HIT command.

STAND

This capability indicates the server implements the STAND command.

### 1.4.3 Extensions

DOUBLEDOWN

This capability indicates the server implements the DOUBLEDOWN command.

SPLIT

This capability indicates the server implements the SPLIT command.

SURRENDER

This capability indicates the server implements the SURRENDER command.

INSURANCE

This capability indicates the server implements the INSURANCE command.

## 1.5 Mandatory and Optional Commands

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The bundle of commands included in this version MUST be supported -as a minimum requirement- by all BJP servers. Therefore, this version includes mandatory commands. Future versions MAY include optional commands. Extensions include optional commands. In future versions the discrimination between mandatory and optional commands will be supported as follows:

* If the label is included in the capability list returned by the server, the server MUST support all commands in that bundle.
* If the label is not included the server MAY support none or some of the optional commands.

## 1.6 Game Session Administrative Commands

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Initial Connection
* CAPABILITIES
* QUIT

## 1.7 Information Commands

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This section lists other commands that may be used at any time between the beginning of a session and its termination. Using these commands does not alter any state information, but the response generated from their use may provide useful information to clients.

* LISTGAMES
* ACCOUNT

## 1.8 Private Commands

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Private Commands serve as way for specific servers to extend the protocol with custom or proprietary features designed to work with custom clients.

Private commands are detailed further in Section 4.2 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 2. Message Definition

This section specifies messaging which is comprised of commands and messages. The commands available with this version (1.0) are specified in detail in this section.

## 2.1 USERNAME

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The USERNAME command is mandatory for the Blackjack Protocol version 1.0.

USERNAME SHOULD be case-sensitive and it MUST contain a unique user-defined alphanumeric combination of 8 - 14 letters and numbers

Syntax

USERNAME <username>

Responses

300 USERNAME received, send password <CRLF>

404 USERNAME is not expected <CRLF>

### 2.1.1. Usage

In order to play Blackjack a player MUST login to the system using a combination of a user/system pre-established USERNAME. The command USERNAME must be authenticated by the system before the user is allowed access to the game and user account. The command USERNAME will be limited to three (3) attempts, after which time server will respond with message saying

If successful the system SHALL return a 300 success code indicating server has received username, client needs to send PASSWORD next.

If unsuccessful, the system SHALL return an error code reflecting the error, such as 404 code indicating that the USERNAME command is not allowed in this state.

.

### 2.1.2 Examples

**Example: Successful username login**

[C] USERNAME *validuser* <CRLF>

[S] 300 Transmission Successful, send password <CRLF>

[Server continues to PASSWORD state]

**Example: Unsuccessful - invalid username**

[C] USERNAME *validuser* <CRLF>

[S] 402 Invalid username/password, user needs to sign in again <CRLF>

[Server abort transaction]

**Example: Unsuccessful - username resubmission**

[C] USERNAME l*oggedinalready* <CRLF>

[S] 404 USERNAME is not expected <CRLF>

[Server aborts transaction]

## 2.2 PASSWORD COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The PASSWORD command is mandatory for the Blackjack Protocol version 1.0.

PASSWORD MUST be defined by user and it MUST be a case-sensitive, alphanumeric combination of 6 -12 letters and numbers. PASSWORD MUST contain at least one (1) uppercase letter and one (1) number. The use of the special symbols ( \*, -, \_ , ! ) are allowed.

### 2.2.1 Syntax

PASSWORD <password>

Response

200 valid PASSWORD <CRLF>

402 invalid login credentials <CRLF>

403 PASSWORD is not expected <CRLF>

### 2.2.2. Usage

In order to play Blackjack a player MUST login with a predefined PASSWORD. The command MUST be used and authenticated by the system before the user is allowed access to the game and user account.

If successful the system MUST return a 200 code indicating a valid Password.

If unsuccessful, the system MUST return either a 402 code indicating an invalid USERNAME and PASSWORD combination (i.e., authentication error), or a 403 code if user has already logged in, or user has not sent USERNAME.

### 2.2.3 Examples

**Example: Successful password submitted**

Precondition: User has not logged in

[C] USERNAME *validuser* <CRLF>

[S] 300 Transmission Successful, send PASSWORD <CRLF>

[C] PASSWORD *correctpassword* <CRLF>

[S] 200 Transmission Successful <CRLF>

[Server continues to Not in Game Group state]

**Example: Incorrect password submitted**

Precondition: User has not logged in

[C] USERNAME validuser <CRLF>

[S] 300 Transmission Successful, send PASSWORD <CRLF>

[C] PASSWORD incorrectpassword <CRLF>

[S] 402 Invalid username/password <CRLF>

[Server continues to Disconnected state]

**Example: Incorrect state**

Precondition: User has not sent USERNAME

[C] PASSWORD <CRLF>

[S] 403 PASSWORD is not expected <CRLF>

[Server continues to Disconnected state]

## 2.3 LISTGAMES COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The LISTGAMES command is required for all 1.0 protocol implementations.

### 2.3.1 Syntax:

LISTGAMES

Response:

102 NUMBER\_OF\_GAMES <CRLF>

*Lines of description for game #1*

…

*Lines of description for game #n*

*<*CRLF>

### 2.3.2. Usage

The LISTGAMES command is used to request a list of all games currently hosted on the server. There are no parameters for this command. A client MAY execute this command only if they are authenticated on the server, and not currently in a game group.

If a minimum of one game is hosted, the server MUST respond with the 102 response code, optionally followed by explanatory text, with the line terminated by a <CRLF>. All hosted games then follow, conveyed in a series of multi-line records, one per game, as described below. All games MUST be listed; there is no capability for private games to be hosted on the server. The list of games MUST be terminated by a <CRLF> pair on its own line.

If there are no games currently hosted, the server MUST respond with a 105 response code, optionally followed by explanatory text, and terminated by a <CRLF>.

### 2.3.3. Game Descriptor Format

A game descriptor SHALL be represented as a multi-line response that consists of a series of game descriptors, one per game hosted on the server, as described above.

Each game descriptor SHALL start with a line identifying the game in question. This line MUST begin with the keyword GAME, followed by a single-word unique identifier used to reference the game in subsequent JOINSESSION commands, and MAY be followed optionally by explanatory text. The tokens on this line MUST be separated by one or more blank or tab characters, or any combination thereof. The unique identifier of the game SHALL be treated as case-insensitive text.

Following this single line are a series of attributes of the game, represented one per line. Each attribute of the game SHALL begin with the keyword ATTRIBUTE, and it SHALL be followed by a single-word identifier of the attribute being specified, and then a single-word value of the attribute. This MAY be followed by optional explanatory text. The tokens on this line MUST be separated by one or more blank or tab characters, or any combination thereof.

Attributes MAY either be mandatory or optional (e.g., custom implementation of the server, presumably known by customized clients that connect to the server). Private attribute identifiers MUST start with the characters “X-”; there is no mechanism currently for deconfliction of private identifiers across multiple implementations. All mandatory attributes MUST be specified in a game descriptor. There is no required ordering for these attributes, though the list of attributes MUST follow the initial header line and precede the optional RULE lines described below.

Mandatory attributes include the following:

* + STATUS: Indicates the status of the game as either ACTIVE (if there are currently players involved in the game) or INACTIVE (if there are no players in the game as yet).
  + MAXPLAYERS: Indicates the maximum number of players allowed in the game, represented as text characters (for example, the character ‘8’ to represent the number 8).
  + MINBET: The minimum size of a bet, represented as a character string representing number of dollars.
  + MAXBET: The maximum size of a bet, represented as a character string.
  + NUMPLAYERS: Indicates the current number of players in the game, represented as a character string.

Optional attributes include the following:

* + MINPLAYERS: Indicates the minimum number of players required to start a game, represented as a character string.
  + NUMDECKS: Indicates the number of decks used in this game, represented as a character string.

Following the lines of attributes MAY come an optional block representing the rules of the game in effect, and serve to specify non-default behavior of the game. An example of a rule is “the dealer must hit on a soft 17.” These rules MUST be represented one per line, with each line beginning with the keyword RULE, and MUST be followed by one or more space or tab characters or any combination thereof, which MUST be followed by the text of the rule. This text SHOULD be displayed to an end user, in the native language that the game server supports. It is not interpreted by the client, simply displayed.

The descriptor of a single game SHALL be concluded with a line that begins with the keyword ‘ENDGAME’, and MAY be followed by explanatory text.

A set of game descriptors MUST be followed by a <CRLF> on a single line, to indicate the end of the response.

### 2.3.4. Example

[C] LISTGAMES <CRLF>

[S] 102 2 List of 2 games follows <CRLF>

[S] GAME game1 Low stakes table <CRLF>

[S] ATTRIBUTE STATUS ACTIVE <CRLF>

[S] ATTRIBUTE MAXPLAYERS 6 <CRLF>

[S] ATTRIBUTE MINBET 5 <CRLF>

[S] ATTRIBUTE MAXBET 50 <CRLF>

[S] ATTRIBUTE NUMPLAYERS 3 <CRLF>

[S] ATTRIBUTE NUMDECKS 4 <CRLF>

[S] ENDGAME <CRLF>

[S] GAME game2 High stakes table <CRLF>

[S] ATTRIBUTE STATUS ACTIVE <CRLF>

[S] ATTRIBUTE MAXPLAYERS 4 <CRLF>

[S] ATTRIBUTE MINBET 100 <CRLF>

[S] ATTRIBUTE MAXBET 500 <CRLF>

[S] ATTRIBUTE NUMPLAYERS 1 <CRLF>

[S] ATTRIBUTE NUMDECKS 6 <CRLF>

[S] ATTRIBUTE MINPLAYERS 1 <CRLF>

[S] RULE Dealer stands on soft 17 <CRLF>

[S] ENDGAME <CRLF>

[S] <CRLF>

## 2.4 QUIT COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The QUIT command is a mandatory command for all versions of the Blackjack Protocol version 1.0

### 2.4.1 Syntax: Client Issues Request

QUIT

Response:

201 Quit successfully <CRLF>

[Server closes connection]

### 2.4.2 Usage

The QUIT command is used to exit the BJP version 1.0 game. The command MAY be used after a successful login at any time during the session - including before, during or after the completion of a game.

Using the QUIT command during an active game MUST result in the player forfeiting the bet that was required to enter the game. The bet MUST be held by the “house” until the remaining players have completed the game in session and then it MUST be reflected either in the appropriate winner’s account status or it MUST remain in possession of the “house”.

If successful, the server MUST respond with an acknowledgment of 200; the line MUST be terminated by a <CRLF> and then MUST immediately close only the connection to the player requesting to leave. The remaining players connections MUST remain intact until they request QUIT.

If unsuccessful, the server MUST respond with either a 400 or 500 failure response code dependent upon the situation regarding failure. Former represents an unsuccessful termination, cause unknown. Latter represents an unknown command or syntax error.

### 2.4.3 Example

[C] QUIT <CRLF>

[S] 201 Service Terminated Normally <CRLF>

[Server closes connection to player]

## 2.5 ACCOUNT COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The ACCOUNT command is required for all 1.0 protocol implementations.

### 2.5.1 Syntax:

ACCOUNT

Response:

104 *balance* <CRLF>

401 Client not authenticated so command is invalid <CRLF>

### 2.5.2 Usage

The ACCOUNT command is used to request the amount of funds in the account balance for the authenticated user. It takes no parameters and allows no optional text, and is terminated by a <CRLF>. A client SHOULD execute this command only if they are authenticated on the server.

### 2.5.3. Examples

**Example: If the user has authenticated and has a balance of $500**

[C] ACCOUNT <CRLF>

[S] 104 500 <CRLF>

**Example: If the client has not authenticated**

[C] ACCOUNT <CRLF>

[S] 401 No user is authenticated <CRLF>

## 2.6 JOINSESSION COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The JOINSESSION command is required for all 1.0 protocol implementations.

### 2.6.1 Syntax:

JOINSESSION *gameid*

Response:

211 Joined the session <CRLF>

410 No such session exists <CRLF>

411 Session exists, but session has maximum number of players already <CRLF>

412 Session exists, but user does not have sufficient funds in their account to cover minimum bet <CRLF>

### 2.6.2. Usage

The JOINSESSION command is used to request that the current user join a game session hosted on the server. It requires a single parameter which is the ID of a game session hosted on the server; such IDs are returned in the LISTGAMES command. The parameter SHOULD be separated from the command by one or more spaces or tab characters, or combination thereof, and the line terminated by a <CRLF>. A client may execute this command only if they are authenticated on the server, and not currently in a game session.

The server SHOULD verify that the game session ID is valid, and that the user meets the criteria for entering the game session. The MINPLAYERS and MAXPLAYERS attributes of the game session are used to determine if there is space for the player to join. Additionally, the user account MUST have sufficient funds to cover the minimum bet (specified by the MINBET attribute) of the game.

If the client successfully joins the game session, any other clients in the session SHOULD receive a notification about the event; see section 2.15 on messages pertaining to game state changes. Additionally, the client SHOULD receive a message corresponding to the current game session state, and subsequent updates pertaining to changes in state. However, they SHALL NOT be able to participate in the game until the next betting state occurs. Until that time they remain as observer, and MUST receive game state updates pertaining to the state of play.

If the client has successfully joined the game session, the server SHOULD respond with a 211 code indicating the successful joining of the game, followed by a game state message and subsequent game update messages.

Several failure conditions are possible, and SHOULD use the following codes:

* + 410: The specified game session ID does not exist on the server
  + 411: The game session is currently not accepting new players. This error occurs when the addition of this client would cause the session to exceed its MAXPLAYERS attribute
  + 412: The user does not have sufficient funds to cover a minimum bet, as specified by the MINBET attribute of the game session

### 2.6.3. Examples

**If the user is not in a session and uses a valid session ID**  
[C] JOINSESSION game137 <CRLF>

[S] 211 Current game state to follow <CRLF>

**If an invalid session ID is used**  
[C] JOINSESSION doesnotexist <CRLF>

[S] 410 No such game <CRLF>

**If a valid session ID is used, but the session already has MAXPLAYERS**  
[C] JOINSESSION game137 <CRLF>

[S] 411 Game is full, try a different one <CRLF>

**If a valid session ID is used, but the user can’t cover the minimum bet**

[C] JOINSESSION game137 <CRLF>

[S] 412 Insufficient funds <CRLF>

## 2.7 LEAVESESSION COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The LEAVESESSION command is required for all 1.0 protocol implementations.

### 2.7.1 Syntax:

LEAVESESSION

Response:

221 Left the session, not mid-play <CRLF>

222 AMOUNT Left the session in mid-play; bet forfeited <CRLF>

413 User is not in a session <CRLF>

### 2.7.2. Usage

The LEAVESESSION command is used to request that the current user leave their current game session. It requires no parameters. A client should execute this command only if they are authenticated on the server, and currently in a game session.

A request to LEAVESESSION MUST be honored by the server, even if the game is currently in progress. The response and server action will vary based on the state of the session, as follows:

* + Response code 221: If the user is in the ‘not in an active session’ state, or in the ‘waiting for bets’ state, the client is removed from the session and there is no adjustment to the user’s account balance. The server must adjust the NUMPLAYERS attribute of the session.
  + Response code 222: If the user is in any other state within an active session, the client is removed from the session and their current bet is forfeited. The account balance must be adjusted to reflect the loss of funds corresponding to the bet amount. There must be a single parameter to this response code, separated by one or more spaces or tabs or combination thereof, which specifies the (positive) amount that the user’s account balance will be debited.
  + Response code 413: User was not currently in a session to leave.

In all successful responses, other clients connected to the same game session should receive a game state update message indicating which user has left the session.

As requests to LEAVESESSION must always be honored, the only failure responses that should be given are ones that reflect internal server error.

### 2.7.3. Example

**If the user has bet $50 but not finished the hand**

[C] LEAVESESSION <CRLF>

[S] 222 50 <CRLF>

**If the user has not yet placed a bet**

[C] LEAVESESSION <CRLF>

[S] 221 Come back soon! <CRLF>

**If the user is not in a session**

[C] LEAVESESSION <CRLF>

[S] 413 Not currently in a session <CRLF>

## 2.8 BET COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 2.8.1 Syntax

BET *amount* <CRLF>

Response:

220 Bet command completed <CRLF>

420 Server is not expecting a bet from this user <CRLF>

421 The amount of bet is not allowed in this section <CRLF>

422 User does not have enough funds to cover the bet <CRLF>

### 2.8.2. Usage

The BET command is used to place a bet in the BJP version 1.0 game. The command MUST NOT be used before a successful JOINSESSION command. The BET request MUST be sent to the system, which MUST validate the availability of funding in the player’s account. System confirms bet with the player, player (client) responds with an OK command and system moves on to the next player in the group; otherwise, if funds are insufficient, an error message MUST be returned consisting of a Response Code which MUST include an argument indicating the actual funds in the account . System MUST allow player to repeat the BET command or to QUIT.

BJP version 1.0 allows the dealer (server) to have a minimum amount and a maximum amount for bets placed in a game. If the player specifies an amount which is lower than the MINBET then server MUST respond with a response code and an attribute to indicating bet below minimum set in game. If the player specifies an amount which results in a bet which is --cumulatively-- higher than the MAXBET then server MUST respond with a response code and an attribute to indicating bet above maximum set in game. If successful server MUST respond with a 220 response code; if BET command aborted, servers MUST respond with a 420 response code.

### 2.8.3. Examples

**Example: Initial Bet is successful**

[C] BET *10* <CRLF>

[S] 220 Bet Command Completed <CRLF>

[Server continues with next player]

**Example: Client in a session but server not waiting for a bet for this client**

[C] BET *10* <CRLF>

[S] 420 Server is not expecting a bet from this user <CRLF>

**Example: Insufficient funds in account; account balance is $30; $50 bet**

[C] BET *50* <CRLF>

[S] 422 Not enough funds in account

**Example: Initial Bet is lower than MINBET**

[C] BET *amount* <CRLF>

[S] 421 Request Aborted; Bet lower than MINBET <CRLF>

**Example: Initial bet is higher than MAXBET**

[C] BET *AMOUNT* <CRLF>

[S] 421 Request Aborted; Total Bet higher than MAXBET <CRLF>

## 2.9 HIT COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 2.9.1 Syntax

HIT

Response:

320 *carddealt* still in game after hit: 10C next hit? <CRLF>

423 *carddealt* user is busted; bet forfeited <CRLF>

424 server is not expecting a HIT from user <CRLF>

### 2.9.2. Usage

The HIT command is used to draw one more card from the deck. Each player is given a turn in which they decide the appropriate action for the cards they have been dealt. Player in turn has the option receive another card to add to the existing total by using the HIT command (or to stay with the hand that they have been dealt using the STAND command). The user decides to receive a HIT; the server deals another card to the player. The HIT command MUST be used in all 1.0 protocol implementations.

### 2.9.3. Examples

**Example: Player chooses to hit, gets a 3 of Spades card and does not bust.**

[C] HIT <CRLF>

[S] 320 3S still in game after HIT <CRLF>

**Example: Player chooses to hit, gets a 9 of Clubs card, and busts.**

[C] HIT <CRLF>

[S] 423 9C busted; bet forfeited <CRLF>

[Server continues with next player]

## 2.10 STAND COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 2.10.1 Syntax

STAND

Response

223 STAND command is completed <CRLF>

425 Unexpected STAND command <CRLF>

### 2.10.2. Usage

The STAND command is used to stay with the hand that player has been dealt. Each player is given a turn in which they decide the appropriate action for the cards they have been dealt. Player in turn has the option to stay with the hand that they have been, dealt using the STAND command (or receive another card to add to the existing total by using the HIT command). The player selects to stand; the server sends back response code 223 and continues with the next player.

If user sends a STAND command when no card is dealt to user, when user is already busted, or it’s not user’s turn to stand, then an error code 425 will be sent to user.

The STAND command MUST be used in all 1.0 protocol implementations.

### 2.10.3. Example

**Example: Player selects to stand**

[C] STAND <CRLF>

[S] 223 STAND command is successful <CRLF>

[Server continues with to the next player]

Example: Clients sends the STAND command twice; the second is disallowed

[C] STAND <CRLF>

[S] 223 STAND command is successful

[C] HIT <CRLF>

[S] 424 Unexpected HIT command <CRLF>

[C] STAND <CRLF>

[S] 425 Unexpected STAND command <CRLF>

[Server continues session]

## 2.11 CAPABILITIES COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The CAPABILITIES command is a required command for all version 1.0 implementations.

### 2.11.1 Syntax:

CAPABILITIES

Response:

101 *optional text* <CRLF>

*CAPABILITY1 optional text* <CRLF>

…

*CAPABILITYn optional text* <CRLF>

<CRLF>

### 2.11.2. Usage

The CAPABILITIES command is used to request a list of all capabilities (mandatory, optional, and private) that the server has implemented that are valid in the current connection state. There are no parameters for this command. A client may execute this command at any state in the protocol, including the non-authenticated state.

If successful, the server must respond with the 101 response code, optionally followed by explanatory text, with the line terminated by a <CRLF>. All server capabilities allowed in the current state then follow, one per line, with each line beginning with the command corresponding to the capability in all caps, optionally followed by one or more parameters if applicable to the command. The command and parameters must be separated by one or more space or tab character. The line must be terminated by a <CRLF>. All capabilities valid for the connection state, even ones required by the protocol version, MUST be listed. Any capabilities that are not valid in the current connection state MUST NOT be listed. Capabilities must only be listed once in a given response. There is no requirement for the ordering of capabilities. The list must be terminated by a <CRLF> pair on its own line.

### 2.11.3. Examples

**Example: Client has just connected and requests capabilities**

[C] CAPABILITIES <CRLF>

[S] 101 Capability list follows <CRLF>

[S] VERSION <CRLF>

[S] CAPABILITIES <CRLF>

[S] USERNAME username<CRLF>

[S] QUIT <CRLF>

[S] <CRLF>

**Example: Client has authenticated and is not in a game, and requests capabilities**

[C] CAPABILITIES <CRLF>

[S] 101 Capability list follows <CRLF>

[S] VERSION <CRLF>

[S] CAPABILITIES <CRLF>

[S] ACCOUNT <CRLF>

[S] LISTGAMES <CRLF>

[S] JOINSESSION sessionid <CRLF>

[S] QUIT <CRLF>

[S] <CRLF>

## 2.12 VERSION COMMAND

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The VERSION command is a required command for all version 1.0 implementations. See also section 1.4.2. (Standard Capabilities)

### 2.12.1 Syntax:

VERSION

Response:

103 version *versionnumber* *optional text* <CRLF>

### 2.12.2. Usage

The VERSION command is used to request the current protocol version that the server implements. There are no parameters for this command. A client may execute this command at any state in the protocol, including the non-authenticated state.

If successful, the server must respond with the 103 response code, followed by the word ‘version’ (case insensitive), followed by the version number, followed by optional text, with the line terminated by a <CRLF>. The different parts of the response line must be separated by one or more tab or space characters, or combination thereof.

### 2.12.3. Example

[C] VERSION <CRLF>

[S] 103 Version 1.0 Group 10 implementation <CRLF>

## 2.13. Summary of response codes

Response codes are defined in detail in Appendix B.

## 2.14. Timeouts

The BJP server in this version (1.0) MUST have at least one inactivity autologout timer for the login state. This timer SHOULD be of at least 45 seconds duration for the login state. The receipt of any command from the client during the timer interval SHOULD suffice to reset the auto-logout timer. When the login timer expires, the server SHOULD close the connection, without sending any response to the client. Timers SHOULD be used while player is within states. Timers MAY be used for handling unexpected server errors. Also, timer SHOULD be used for returning to state server was in, before selecting to enter one of the Information states (ACCOUNT, CAPABILITIES, LISTGAMES, VERSION).

### 2.14.1. Usage

The autologout timer is used in the login state to prevent the system from remaining idle because of a client not responding while in the USENAME or PASSWORD states. Additionally, a timer MAY be used if the client does not respond to any other state after the states mentioned above.

In the case where the client remains idle for more than 45 seconds the server SHALL abort the action and respond with a 50X response code which MAY include an argument.

### 2.14.2. Example

**Example: A timeout occurs during the authentication process**

[C] USERNAME <CRLF>

[Client is idle for more than 45 seconds]

[Sever disconnects]

Timeout examples for other states are provided in the table below.

### 2.14.3 Summary of Timeouts per State

|  |  |  |  |
| --- | --- | --- | --- |
| **State** | **Minimum Timeout** | **Next State** | **Response Code** |
| Awaiting username | 45 seconds | Disconnected | *None; connection closes* |
| Awaiting password | 45 seconds | Disconnected | *None; connection closes* |
| Not in session | 900 seconds | Disconnected | *None; connection closes* |
| In session, awaiting bet | 60 seconds | In session as observer | 106 timed out |
| In session, awaiting play | 60 seconds | Not in session | 107 timed out |

## 2.15. Game State Messages

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Game state messages are broadcast by the server to all clients within an active game session to indicate the progress of the game. They are an exception to the general request-response pattern of protocol interactions, in that they are broadcast from the server without a client request to prompt them. Game state messages are used to indicate the cards that are dealt in the game and the actions of other players, as well as to the prompt for actions that are needed from various connected clients in the session.

Game state messages are of one of two types: informative messages concerning gameplay, and messages that indicate a response is needed from the client. The former case involves actions like users placing bets, cards being dealt, or bets being settled. The latter case indicates that it is that user’s turn to play their cards, either with the HIT or STAND commands (or any relevant private command that may be implemented on the server). Purely informative messages are broadcast in a line beginning with code 600, while messages that indicate user action is required have code 601.

Game state messages are multiline messages, with each line terminated with a <CRLF> and a final <CRLF> indicating the end of the entire message. Tokens within the strings must be separated by one or more spaces or tabs, or any combination thereof. The message consists of one mandatory and at least one of four optional parts, as follows:

1. The MANDATORY header line  
   This line consists of either the message code 600 or 601, as described above, and optional text that may or may not be presented to the user by their user interface. This line is terminated by <CRLF>.  
   Example: 600 Cards are being dealt!
2. An OPTIONAL dealer hand line  
   This line indicates the cards that are held by the dealer. Facedown cards are indicated by an X, and faceup cards are indicated by a combination of rank and suit as described in Section 1.1.3. The dealer line begins with the word DEALER, which is followed by the facedown card that is dealt first, followed by the other cards in the order they are dealt. This line is terminated by <CRLF>.  
   Example: DEALER X 7C
3. Zero or more OPTIONAL player hand lines  
   The number of player card lines is equal to the number of players in the current session, including the client who the message is being sent to. Each player hand is represented in the same manner as with the dealer hand. The line contains the following parts, in order and separated by one or more spaces or tabs:
   1. The word ‘PLAYER’
   2. The word ‘ACTIVE’ (indicating it’s that player’s turn), ‘INACTIVE’ (indicating that it’s not that player’s turn but that they are still in the session), or ‘GONE’ (indicating that the player was dealt into the game but has left)
   3. The user’s username (guaranteed to be unique across users)
   4. The user’s full name in quotes (not guaranteed to be unique across users)
   5. The cards in the hand in the order they are dealt

Note that for the user who is receiving the message, all cards in their hand will be presented as faceup with their value shown.

For a game state message without any player hand lines, or with missing player hand lines, the status of that player’s hand should be assumed to be unchanged from the last message received.

1. Zero or more OPTIONAL lines indicating the last action(s)  
   These lines contains the following parts, in order and separated by one or more spaces or tabs:
   1. The word ‘LASTACTION’
   2. The username of the user who performed the action, or the word ‘DEALER’ in the case of the dealer
   3. The user’s full name in quotes, or the word “Dealer” in the case of the dealer
   4. A keyword indicating the action the user performed, as described below
   5. Zero or more parameters as determined by the keyword
2. One or more OPTIONAL lines indicating the next action(s) that the serving is waiting on. These lines contain the following parts, in order and separated by one or more spaces or tabs:
   1. The word ‘NEXTACTION’
   2. The username of the user who is requested to perform the action
   3. The user’s full name in quotes
   4. A keyword indicating the action that is being waited on

For describing the last actions as per part 4, the following keywords (with the indicated parameters) can be used:

* JOINED  
  Indicates that the user has joined the game session with the JOINSESSION command
* LEFT  
  Indicates that the user has left the game session with the LEAVESESSION command
* BET *amount*Indicates that the user has bet the indicated amount of dollars
* NEWCARD *card*Indicates that the user has been dealt a new card of the indicated value (either X for a facedown card, or a combination as described in section 1.1.3).
* HIT  
  The user decided to take another card with the HIT command
* STAND  
  The user decided to stand with the STAND command
* BLACKJACK  
  The user hit 21 exactly
* BUST  
  The user totaled more than 21 points in their hand, and busted

For requesting next actions that the server is waiting for as per part 5, the following keywords can be used:

* NEEDBET  
  The server is waiting for the user to place their bet
* NEEDPLAY  
  The server I waiting for the player to either HIT or STAND

### 2.15.1 Examples

**An initial game state message before cards are dealt and bets made:**

601 A new hand is beginning! <CRLF>

NEXTACTION billy627 “Billy Bobson” NEEDBET <CRLF>

NEXTACTION blackjackstud “Jimmy Johnson” NEEDBET <CRLF>

NEXTACTION kcanderson “Kary Anderson” NEEDBET <CRLF>

<CRLF>

**After all bets are made, a message indicating the hands dealt with the first player needing to make their play; the message is coded as 600 (informative) as it is being sent to the second player, who does not need to make an action:**

600 It’s Billy Bobson’s turn <CRLF>

DEALER X 6H <CRLF>

PLAYER ACTIVE billy627 “Billy Bobson” X 3C<CRLF>

PLAYER INACTIVE blackjackstud “Jimmy Johnson” 3S JH<CRLF>

PLAYER INACTIVE kcanderson “Kary Anderson” X AD <CRLF>

NEXACTION billy627 “Billy Bobson” NEEDPLAY <CRLF>

<CRLF>

**The first player decided to hit, and receives a new card. This message is sent to the second player, who does not need to make an action:**

600 It’s Billy Bobson’s turn <CRLF>

PLAYER ACTIVE billy627 “Billy Bobson” X 3C 2S <CRLF>

LASTACTION billy627 “Billy Bobson” HIT <CRLF>

NEXACTION billy627 “Billy Bobson” NEEDPLAY <CRLF>

<CRLF>

(Note that in the message above, the dealer’s hand and the hands of the second and third player are not included. This is because they are unchanged since the last message, and are not required.)

**The first player decides to stand, and it is now player two’s turn to play. This message is sent to the second player, who needs to make an action:**

601 It’s Jimmy Johnson’s turn <CRLF>

PLAYER INACTIVE billy627 “Billy Bobson” X 3C 2S <CRLF>

PLAYER ACTIVE blackjackstud “Jimmy Johnson” 3S JH <CRLF>

LASTACTION billy627 “Billy Bobson” STAND <CRLF>

NEXTACTION blackjackstud “Jimmy Johnson” NEEDPLAY <CRLF>

<CRLF>

**While waiting for player two to make their play, a fourth player joins the game:**

600 Look, a new player! <CRLF>

LASTACTION ketchupking “Mr. Heinz” JOINED <CRLF>

<CRLF>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 3. DFA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Deterministic Finite Automaton (DFA) below represents the states of version 1.0 of the blackjack protocol. Commands are in red text as state transitions.

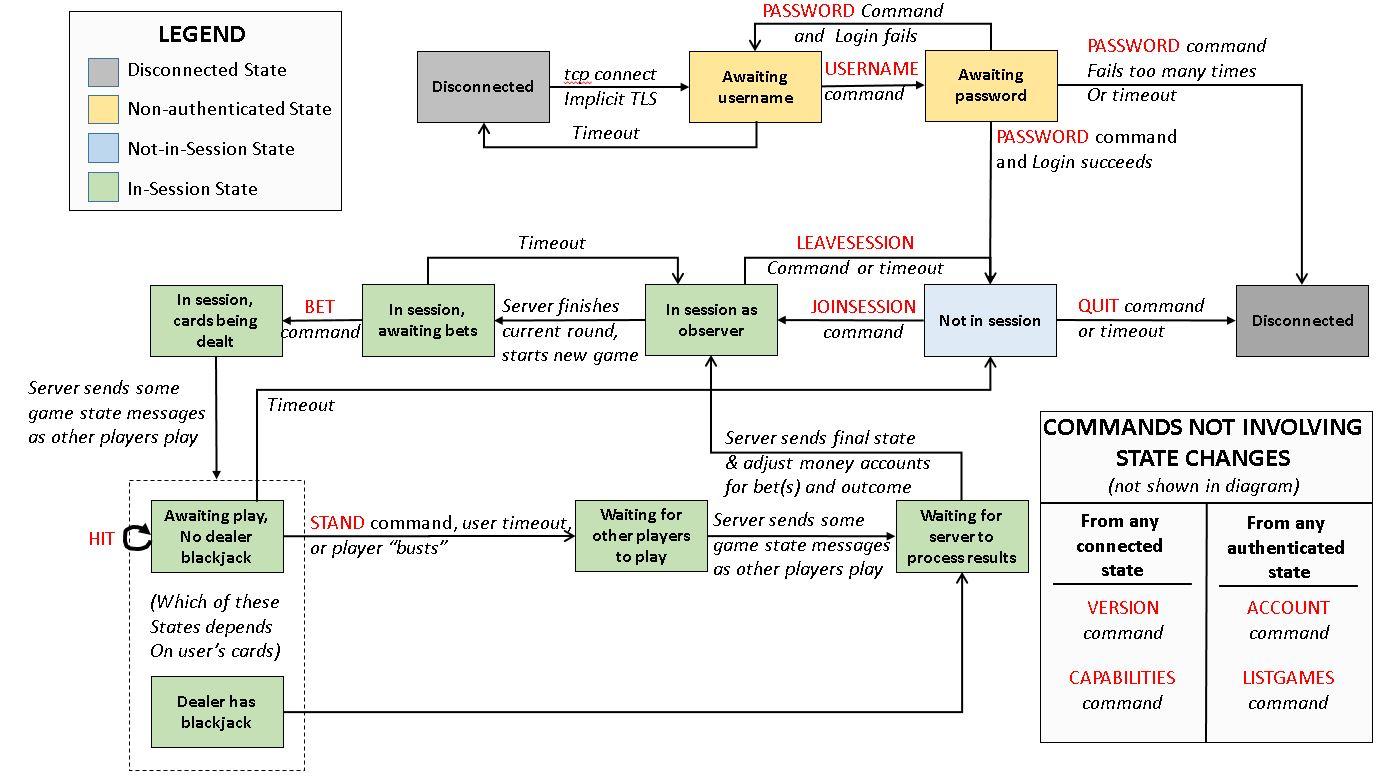


Figure 1 - Blackjack DFA

The top line illustrates the simple username / password method of user authentication. Below that, the light blue box represents a state where the user is authenticated, but not currently in a game group and playing the game. This MUST be the only state -- authenticated but not in a game -- from which the LISTGAMES and ACCOUNT commands are permissible. The green boxes represent game-playing states and represent a fairly linear progression of placing a bet, receiving updates on game play, playing the cards in one’s own hand, and viewing the outcome of the game.

Not depicted in the diagram are two commands, the CAPABILITIES and VERSION command, which MAY be made from any connected game state (including non-authenticated states) and do not cause a state transition. Also not fully specified are the details of the “user timeout” mechanism, which is shown explicitly from a few states but can, in actuality, affect the user in any state after a certain period of inactivity. Section 2.14 explains the timeout mechanism in greater detail.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## 3.1. Use cases considered in DFA construction

Since the requirement of this project includes the implementation of a client-server application working on top of the here-specified protocol, Use Cases are used to describe some possible scenarios at the implementation level; these serve as examples of scenarios of how an application would behave over said protocol.

Detailed Use Cases are presented in Appendix A.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 4. Extensibility

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Extensibility in BJP 1.0 is enabled by the specification of (i) Version, (ii) Private Commands, (iii) Games Hosted Description, and (iv) Options.

## 4.1 Version

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Section 1.4.2 Standard Capabilities, describes the basic specification of Version. When a new minor version of the protocol is created, it MUST be backward compatible with the original X.0 major revision of the prototype.

Example:

Future version 1.1 of the protocol MUST be backward compatible with the current version 1.0. New major versions of the protocol MAY be compatible, so that a future version 2.0 of the protocol is RECOMMENDED to maintain backward compatibility (though such compatibility SHOULD be encouraged where feasible).

## 4.2 Private Commands Description

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The specification lists three types of commands: Game Session Administrative commands, Information commands, and Private commands. The third type of commands, **Private Commands**, serve as way for specific servers to extend the protocol with custom or proprietary features designed to accommodate custom clients.

Private commands SHOULD begin with an “X-” prefix.

Private commands SHOULD be implemented under the same general paradigm as existing Mandatory and Optional commands. Where applicable, they SHOULD use existing response codes. When it is necessary to create extended response codes, the range of response codes from x80-x99 MAY be used.

### 4.2.1 Example

* An extended command that allows searching for games meeting a specific criteria MAY be called X-SEARCH. There is currently no mechanism in place for deconfliction against possible usage of the same command name by multiple implementers.
* The previously mentioned X-SEARCH command MAY have a response code of 580 to indicate that search parameters were not recognized as valid.

## 4.3 Games-Hosted Description

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A final way in which the protocol is extensible lies in the way that games hosted in the server are described. As part of the PDU that represents information about a game, there is a textual list of game rules that are in effect for that game. Some of these game rules are defined in Section 2.3 which talks about the LISTGAMES command, such as ‘Number of decks’. Game rules that are not in the specification MAY be returned, but MUST be prefaced with the word ‘Custom’ before the text of the rule (which need not be ‘human-readable’ but should be reasonably descriptive).

### 4.3.1 Example

A game MAY choose to implement a rule where the dealer hits, as opposed to stands, on a soft 17. They MAY advertise this rule with the line “Custom Dealer hits on soft 17” in the Game Rules section of the PDU. Of course, the server MUST support the rule for the game to take advantage of it. The client then, MUST convey these custom rules to users.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 5. Security Implications

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Due to its sensitivity in regards to passwords and other account information, all client-server connections MUST take place over a secure socket connection. Upon connecting to the server, the secure connection MUST be made prior to any commands, including user authentication, being made. In version 1.0 protocol implementations, secure connections SHALL be made with standard TLS security measures on the default blackjack port.

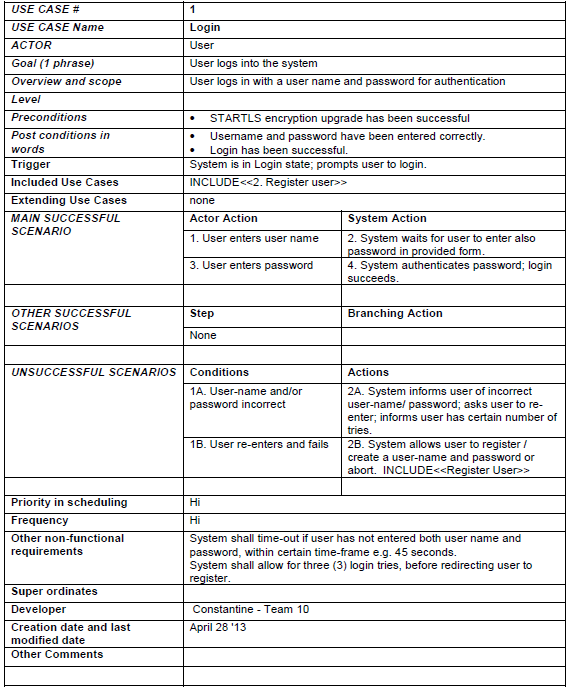
Once an encrypted connection is established, the user MUST be authenticated. Such an authentication MUST be made before any other capabilities have been enacted upon, with the exception of the VERSION and CAPABILITIES commands, which can be made. These capabilities are allowed because they are used to advertise what methods of authentication are available. The only required method of authentication in version 1.0 of the protocol is the SIMPLE method (AUTH SIMPLE), which requires a username and password. Accounts SHOULD be disabled after three incorrect attempts at logging in, and that errors only indicate whether or not the combination of username and password failed, as opposed to revealing whether or not such an account exists under that username.

Security of account information is beyond the scope of this protocol. In general, best practices SHOULD be adhered to when it comes to safeguarding user information when the protocol is used in a production environment. Within a development environment, such security measures on account information MAY be used; however, the protocol MUST use an encrypted connection, and it MUST use simple authentication.

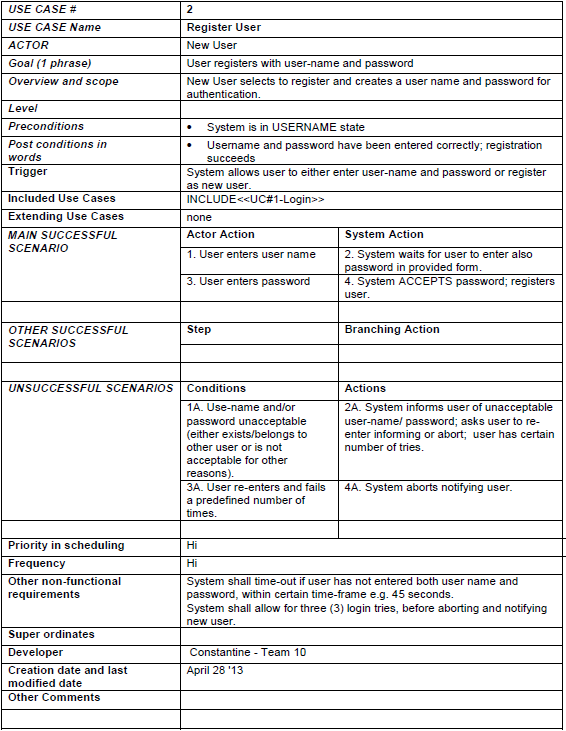
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Appendix A. USE CASES

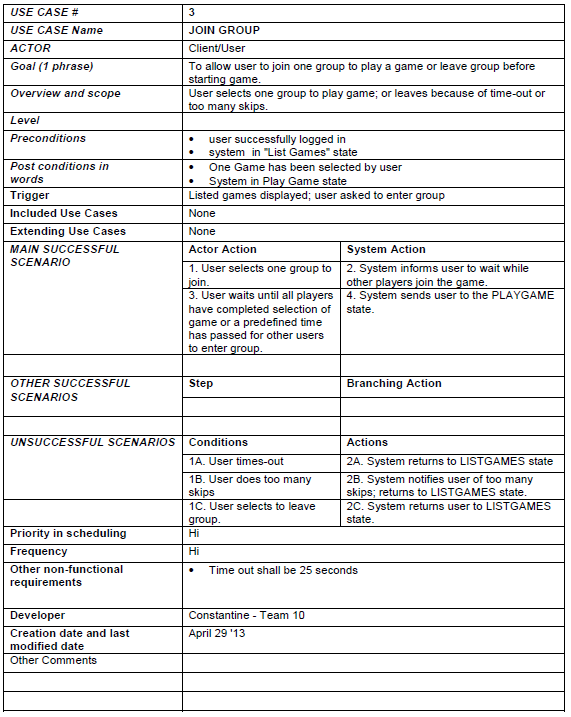
## Use Case 1: Log-in



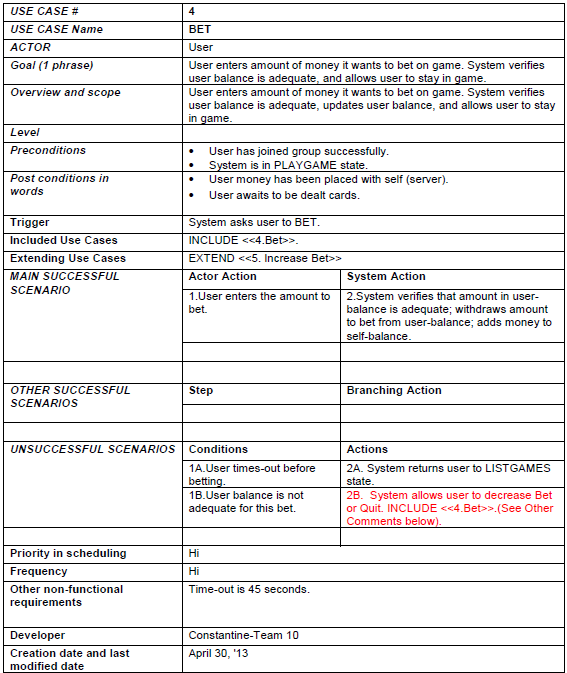
## Use Case 2: Register user



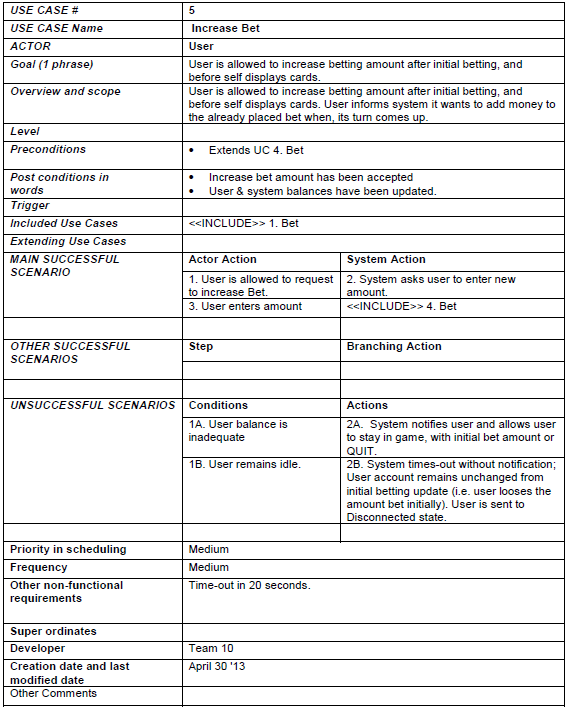
## Use Case 3: Join Group



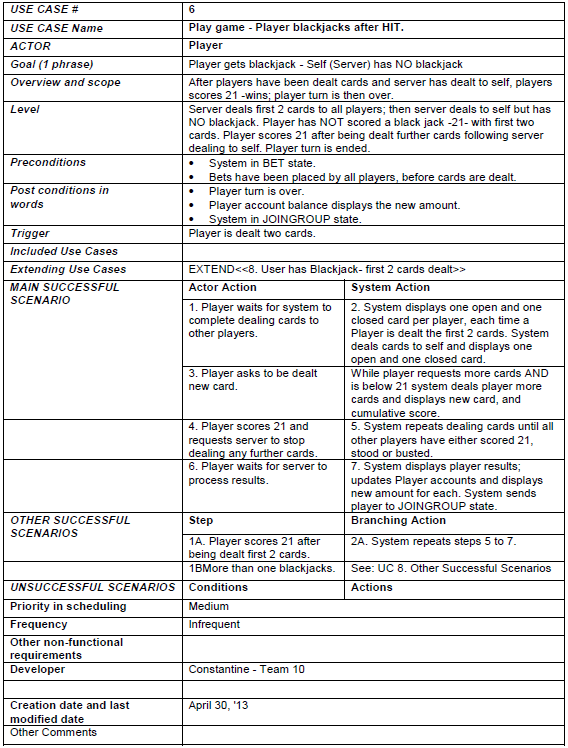
## Use Case 4: Bet



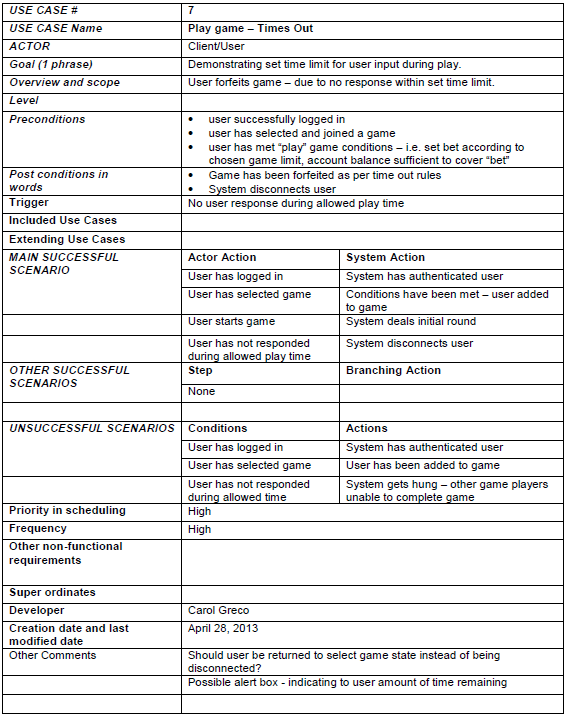
## Use Case 5: Increase Bet



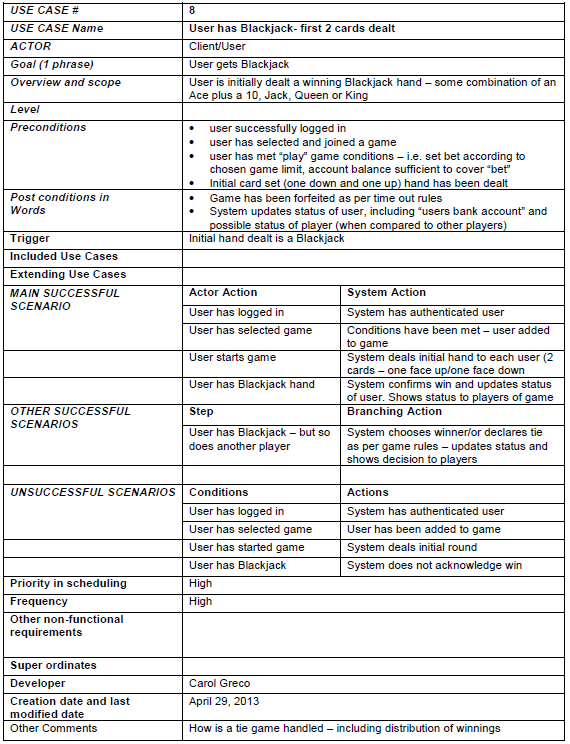
## Use Case 6: Play - Player blackjacks after HIT



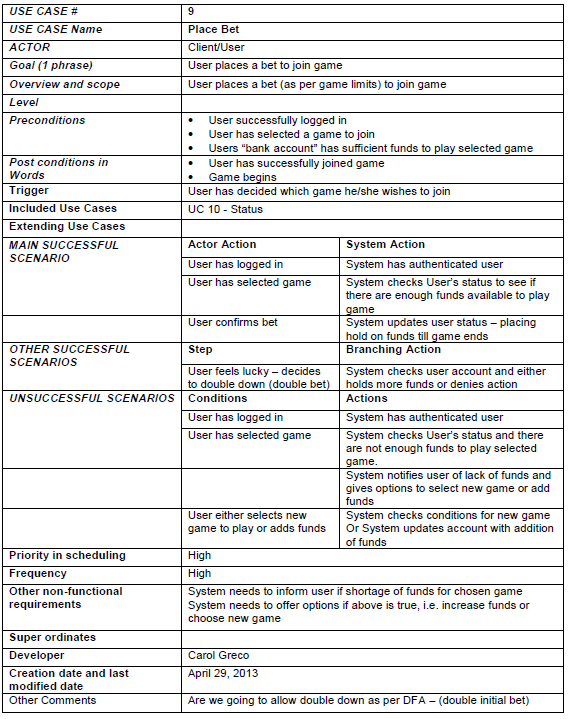
## Use Case 7: Play game - Times Out



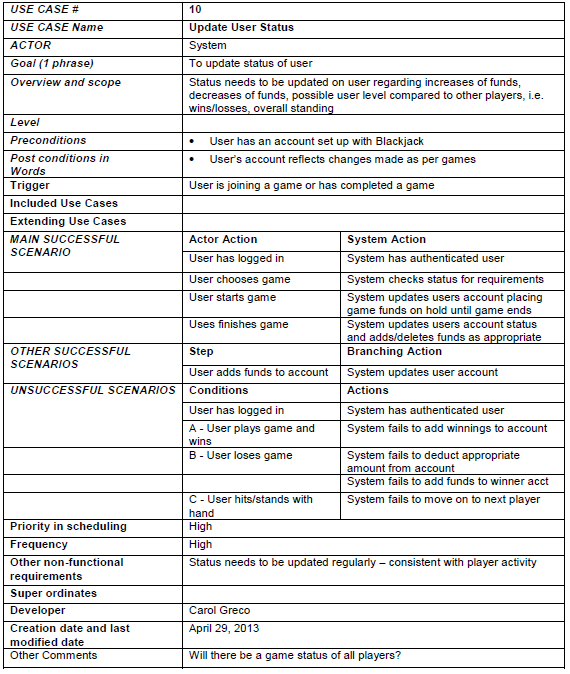
## Use Case 8: Play game - User has Blackjack - first 2 cards dealt



## Use Case 9: Place Bet



## Use Case 10: Update User status



# APPENDIX B. RESPONSE CODES DEFINED

Response code 101

Generated by CAPABILITIES

Arguments: 0

Multiline, capabilities followed

Meaning: List all the mandatory and private capabilities supported by the server

Response code 102

Generated by LISTGAMES

Arguments: <NUMBER\_OF\_GAMES>

Multiline, game descriptors follows

Meaning: Server returns a list of games for user to choose

Response code 103

Generated by VERSION

Arguments: <VERSION>

Singleline

Meaning: Current protocol version that the server implements

Response code 104

Generated by ACCOUNT

Arguments: <BALANCE>

Singleline

Meaning: The amount of funds in the account balance for the logged-in user

Response code 105

Generated by LISTGAMES

Arguments: 0

Singleline

Meaning: Server doesn’t host any games

Response code 106

Generated by *timeout exceeded while awaiting bet*

Arguments: 0

Singleline

Meaning: The connection was idle too long while a bet was expected

Response code 107

Generated by *timeout exceeded while awaiting play*

Arguments: 1) the amount of bet that was forfeited as a text string

Singleline

Meaning: The connection was idle too long; the bet made was forfeited

Response code 200

Generated by PASSWORD

Arguments: 0

Singleline

Meaning: Valid username/password, user is logging in

Response code 201

Generated by QUIT

Arguments: 0

Singleline

Meaning: Client has quitted the game successfully

Response code 211

Generated by JOINSESSION

Arguments: 0

Singleline

Meaning: User has successfully joined the requested session

Response code 220

Generated by BET

Arguments: 0

Singleline

Meaning: BET command is successful, money is withdrawn from account

Response code 221

Generated by LEAVESESSION

Arguments: 0

Singleline

Meaning: User has successfully left the session when not in mid-play

Response code 222

Generated by LEAVESESSION

Arguments: AMOUNT

Singleline

Meaning: User has left the session when in mid-play, bet is forfeited, amount of bet forfeited is reported back

Response code 223

Generated by STAND

Singleline

Meaning: STAND command is successful

Response code 300

Generated by USERNAME

Arguments: 0

Singleline

Meaning: Username received, send password to log in

Response code 320

Generated by HIT

Arguments: 1) The card dealt in its textual form

Singleline

Meaning: Card is dealt to user and user is not busted with the new card. Wait for HIT or STAND to come next

Response code 400

Generic Error

Arguments: 0

Singleline

Meaning: Internal error, server could not execute this command

Response code 401

Generic Error

Arguments: 0

Singleline

Meaning: User has to authenticate to execute this command

Response code 402

Generated by PASSWORD

Arguments: 0

Singleline

Meaning: Invalid username/password, user needs to sign in again

Response code 403

Generated by PASSWORD

Arguments: 0

Singleline

Meaning: Password is not expected, user needs to sign in with USER first

Response code 404

Generated by USERNAME

Arguments: 0

Singleline

Meaning: Username is not expected, user has already logged in

Response code 410

Generated by JOINSESSION

Arguments: 0

Singleline

Meaning: The requested session does not exist

Response code 411

Generated by JOINSESSION

Arguments: 0

Singleline

Meaning: The requested session has already reached maximum players

Response code 412

Generated by JOINSESSION

Arguments: 0

Singleline

Meaning: User does not have enough fund to cover the minimum bet

Response code 413

Generated by LEAVESESSION, BET, HIT, STAND

Arguments: 0

Singleline

Meaning: User is not in a game/session

Response code 420

Generated by BET

Arguments: 0

Singleline

Meaning: User is in game, but server is not expecting a BET from this client

Response code 421

Generated by BET

Arguments: 0

Singleline

Meaning: Amount of bet is not allowed in this session

Response code 422

Generated by BET

Argument: 0

Singleline

Meaning: User does not have enough fund to cover amount of bet

Response code 423

Generated by HIT

Argument: 1) The card dealt in its textual format

Singleline

Meaning: Card is dealt to user, user is busted

Response code 424

Generated by HIT

Argument: 0

Singleline

Meaning: HIT is not expected because game is not in play, user is not dealt any cards, or user is busted

Response code 425

Generated by STAND

Argument: 0

Singleline

Meaning: STAND is not expected because game is not in play, user is not dealt any cards, or user is busted

Response code 500

Generic Error

Argument: 0

Singleline

Meaning: Unknown or unsupported command

Response code 501

Generic Error

Argument: 0

Singleline

Meaning: Unsupported command

Response code 502

Generic Error

Argument: 0

Singleline

Meaning: Some sort of unspecified internal error occurred

# APPENDIX C. TERMINOLOGY DEFINED

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*This document contains a glossary of common terms and actions associated with the game of Blackjack, which may be used or referenced in the Blackjack Protocol version 1.0.*

### 1.1 BASE ACTION PLAYS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*These two procedures are minimum playing options that any server must have to play the Blackjack Protocol v.1.0*

***HIT:***

One of the two (2) base options that are offered by any server. The Player requests another card from the dealer. Player may be “hit” as many times during a hand as they request, as long as the hand total does not exceed 21.

***STAND:***

*The second of the two (2) base options that are offered by any server. The Player stays with the cards they are dealt. No more cards are given to the player during the round.*

***BUST:***

When a players hand goes over the value limit of 21, the loss is irrevocable and the player’s bet is forfeited to the house. In a multiplayer game, the play moves forward to next player in the game.

***WIN:***

After each player in the game has finished their round, the dealer reveals his turned down card and either stands or draws more cards towards the goal of blackjack. Final scores are calculated and displayed to all players. The bets on the loser’s hands are forfeited and the winner receives the payout.

### 1.2 OPTIONAL PROCEDURES

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*A server may or may not offer the following features for playing the BJP v.1.0. These descriptions only serve to provide a basic understanding of the term and may not reflect a server’s full description of the feature as many variations of each command exists in a real world game. A player must refer to the guidelines of commands at an individual server.*

***FOLD:***

When the player decides not to continue the game at play. At a fold, the house takes half the players bet and returns the other half to the player.

***SPLIT:***

Is only available as a first decision of a hand. If the player has been dealt two cards of equal value, they may opt to split their hand and play two hands. This also requires that the player double their bet, betting the required amount for each hand. Blackjack after a split of the hand is considered a non-blackjack 21. This means that the hand can be a higher value than the dealer’s non blackjack hand, but is not considered a win if the dealer has blackjack.

***SURRENDER:***

Represents an option as a first decision of a hand is when the player decides not to continue the game at play. At surrender, the house takes half the players bet and returns the other half to the player. Some servers may have both early (before dealer checks hand) and late (after dealer checks hand) surrender options.

### 1.3 BETTING OPTIONS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*A server may or may not offer the following betting options for BJP 1.0*

***DOUBLE-DOWN*** [5]***:***

This is a betting procedure used by a player that feels confident enough that they will beat the house and win the game using only the two(2) cards dealt plus a HIT(1) of one more card. Using a double-down, a player is allowed to double their initial bet on the hand in play. However, the stipulation exists that they may only be HIT one(1) time - receive one more card. If player wins the winning is 2:1 of a doubled bet.

***INSURANCE*** [6] ***:***

An option to a player when the dealer’s hand shows an Ace face up. A player may buy Insurance in case the dealer has Blackjack. A players with the Insurance option can get paid a 2:1 to insurance bet, up to ½ of the initial bet.

### 1.4 OTHER TERMINOLOGY

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*The following are typical terms used in the playing of Blackjack* [7]*.*

***HARD HAND:***

**I**n this hand a player may or may not have an Ace, it may have one, but it is not required. If the player has an Ace, the value of that Ace is always one (1) and is inflexible.

***SOFT HAND:***

Is a hand that has an Ace. In this hand the value of the Ace is eleven (11). In this hand the Ace and its value are key elements. This type of hand gives the player an advantage - they choose to add a third card without worrying about going “bust” as the value of the Ace can change to a 1. For example, the term “soft seventeen” is used in the protocol description. In this scenario, the player initially receives an Ace and a six (6) card. Suit is unimportant. The player can choose the HIT command, be dealt a card (say a 10) that would normally bring the card count to over twenty-one (21) - however, in the soft hand the player is safe as the Ace value would change to a one (1).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# APPENDIX D: REFERENCES

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| [1] | L. Ofton, "The History of Blackjack," 2013. [Online]. Available: http://www.blackjackapprenticeship.com/resources/history-of-blackjack/. [Accessed 11 May 2013]. |
| [2] | C. Feather, "RFC 3977: Network News Transfer Protocol (NNTP)," October 2006. [Online]. Available: https://tools.ietf.org/html/rfc3977. [Accessed 10 May 2013]. |
| [3] | S. Bradner, "RFC 2119: Key words for use in RFCs to Indicate Requirement Levels," March 1997. [Online]. Available: https://tools.ietf.org/html/rfc2119. [Accessed 11 May 2013]. |
| [4] | F. Yergeau, "RFC 3629: UTF-8, a transformation format of ISO 10646," November 2003. [Online]. Available: https://tools.ietf.org/html/rfc3629. [Accessed 10 May 2013]. |
| [5] | 21onlinecasinos.com, "Blackjack Doubling Down - Understanding the Double Down in Blackjack," 21onlinecasinos.com, 2013. [Online]. Available: http://www.21onlinecasinos.com/doubling-down.html. [Accessed 10 May 2013]. |
| [6] | 21onlinecasinos.com, "Blackjack Insurance - Understanding the Concept of Insurance in Blackjack," 21onlinecasinos.com, 2013. [Online]. Available: http://www.21onlinecasinos.com/blackjack-insurance.html. [Accessed 10 May 2013]. |
| [7] | 21onlinecasinos.com, "Hard and Soft Hands - Learning About Hard and Soft Blackjack Hands," 21onlinecasinos.com, 2013. [Online]. Available: http://www.21onlinecasinos.com/hard-and-soft-hands.html. [Accessed 10 May 2013]. |