C34 SDST Chess Network Protocol

***Squirrel Eiserloh****, Summer 2025*

# Network Protocol & Commands

In our Chess game we are using a simple NetworkSystem to send text (only) via TCP/IPv4 (only) to another computer, and interpreting incoming text as remote DevConsole commands for us to execute.

The following DevConsole commands/events and network protocols must be observed exactly, as they will be used in common for the entire class (and your players must be able to connect and play with each other over the network!).

Once a network connection is established, commands marked in this document as [remote] can be sent to the connected computer (and often execute local side-effects as well) as strings to be executed in the remote computer’s DevConsole (as Events). Commands marked as [local] are not sent as remote commands; they are only executed locally (though may cause other network activity). This document uses <angleBrackets> for argument descriptions and [squareBrackets] for optional arguments (< and > and [ and ] are not actually typed).

All commands received over the network shall automatically have the argument remote=trueadded to them automatically in code by the receiver just after being received before being executed (so that the subscribed function that handles that event can know whether this originated locally or remotely).

If you see the remote=true argument, you know the command came from someone else (via the network); if you do not see it then it is coming from your own computer.

1. [local] **ChessServerInfo [ip=<remoteIPAddress>] [port=<portNumber>]**: All arguments are optional; prints out all values (name, ip, port) and current connection status and game state regardless of which arguments are given (or none!).
   1. Example: “**ChessInfo**”
   2. Example: “**ChessInfo ip=192.168.0.53**”
   3. Example: “**ChessInfo port=3100**”
   4. *Note: If currently connected, any changes to these values should be rejected and warned.*
2. [local] **ChessListen [port=<portNumber>]**: Run StartServer on your NetworkSystem, then start a Listen socket at the specified port number (or use existing default or previously-overridden port number if omitted). Code default is port=3100.
   1. Example: “**ChessListen**” // use port from code or previous **ChessInfo**.
   2. Example: “**ChessListen port=3101**” // explicit port number override
3. [local] **ChessConnect [ip=<remoteAddress>] [port=<portNumber>]**: Run StartClient on your NetworkSystem, then try to Connect to a server at this remote IP address and port number. Add any omitted arguments using existing (default or previously-overridden) values. Code defaults are ip=127.0.0.1 and port=3100.
   1. Example: “**ChessConnect**” // use current IP and port from code, GameConfig.xml, or previous ChessInfo command.
   2. Example: “**ChessConnect ip=192.168.0.53**” // explicit IP, current port
   3. Example: “**ChessConnect ip=192.168.0.53 port=3100**” // explicit IP and port
4. [remote] **ChessDisconnect [reason="<explanation>"]**: Send this command remotely to the connected computer, then disconnect yourself after sending. If the command includes a “remote=true” argument then it was sent to you from the remote host; disconnect your connection but do not re-send it back.
   1. Example: “**ChessDisconnect**”
   2. Example: “**ChessDisconnect reason=Goodbye!**”
   3. Example: “**ChessDisconnect reason="I have to go cook dinner."**”
5. [remote] **ChessPlayerInfo [name=<myName>]**: Sets your local player name. If received from remote host with the “remote=true” argument, sets your opponent’s player name instead.
   1. Example: “**ChessPlayerInfo name=Squirrel**” // My name is Squirrel
   2. Example: “**ChessPlayerInfo name=Matt remote=true**” // Opponent says he is Matt
6. [remote] **ChessBegin [firstPlayer=<playerName>]**: Start a new match with the named player as the starting player (playing the traditional “white” position), defaults to you if unspecified.
   1. Example: “**ChessBegin**” // start new game with me as first player (adds firstPlayer=<myName> to remote command before sending)
   2. Example: “**ChessBegin firstPlayer=Matt**” // I’m starting a new game; Matt goes first
   3. Example: “**ChessBegin firstPlayer=Matt remote=true**” // Matt started a new game, and wants himself to go first
7. [remote] **ChessValidate [state=<gameState>] [player1=<playerName>] [player2=<playerName>] [move=<moveNumber>] [board=<64characters>]**: Validate the current state of the game, names and seats of the players, current move number (starting with 0 before the opening move is made), and the state of the entire board. Standard piece character abbreviations are used (see above). If you receive this (remote=true) then validate each item against your own game state and, if ***any*** discontinuity is found:
   * 1. Print a detailed report to the DevConsole log, noting all discontinuities;
     2. Send **ChessDisconnect reason="VALIDATION FAILED"** to your opponent;
     3. Disconnect.
   1. Example: “**ChessValidate**” // I’m initiating a validation; all info will be sent to opponent…
   2. …and then “**ChessValidate state=Player1Turn player1=Squirrel player2=Matt move=0 board=RNBKQBNRPPPPPPPP................................pppppppprnbkqbnr remote=true**” // …would be received by my opponent as a result
   3. Valid game state strings include: “Player1Moving”, “Player2Moving”, and “GameOver”
8. [remote] **ChessMove from=<square> to=<square> [promoteTo=<pieceType>]**: Move the current player’s piece from one square to another; the move must be legal; it is considered a network protocol error to send an illegal move. If this is a pawn moving into the final (furthest) rank (row), the promoteTo= argument is required; otherwise, it is forbidden.
   1. Example: “**ChessMove from=e2 to=e4**” // you are making a move
   2. Example: “**ChessMove from=c7 to=c8 promoteTo=knight**” // promoting a pawn
   3. Example: “**ChessMove from=e2 to=e4 remote=true**” // Opponent says he moved
   4. Example: “**ChessMove from=c7 to=a1 teleport=true**” // ignore move rules
9. [remote] **ChessResign**: You resign (forfeit) the current game. Only valid while a game is in progress; can be used on either player’s turn. If remote=true then opponent resigned.
   1. Example: “**ChessResign**” // you resign
   2. Example: “**ChessResign remote=true**” // Opponent is telling you that he resigned
10. [remote] **ChessOfferDraw**: You offer your opponent a draw (mutually-agreed tie game). Only valid while a game is in progress; can be used on either player’s turn. *You may wish to limit each player’s ability to send this message more than once per turn; this rejection should happen locally.*
    1. Example: “**ChessOfferDraw**” // you are offering a draw (tie game)
    2. Example: “**ChessOfferDraw remote=true**” // Opponent is offering you a draw (tie)
11. [remote] **ChessAcceptDraw** and **ChessRejectDraw**: You accept (or reject) your opponent’s offer of a draw (mutually-agreed tie game). Only valid while a game is in progress, and after your opponent has just sent you a ChessOfferDraw message (with remote=true).
    1. Example: “**ChessAcceptDraw**” // I accept my opponent’s offer to draw (tie) the game
    2. Example: “**ChessRejectDraw**” // I reject their offer
    3. Example: “**ChessAcceptDraw remote=true**” // My opponent accepted my offer to draw
12. [local] **RemoteCmd cmd=<commandName> [<key1>=<value1>] [key2=<value2>]…** : **Builds a DevConsole command string** to send to the remote computer for execution, where <commandName> is the actual DevConsole command (and EventSystem event name) to be executed remotely. “RemoteCmd” is not sent as part of the command string, nor is “cmd=”. When your Chess game receives a remote command string from the network, you should (a) append the text “remote=true” onto the end of the string, and then (b) Execute that string in your dev console (hopefully firing an Event, if the command name and arguments are correct).
    1. Example: “**RemoteCmd cmd=Echo text=Hello**” would send “Echo text=Hello” over the network; the receiving computer would add “ remote=true” on the end of it, and then Execute the final command string “Echo text=Hello remote=true” in its DevConsole.
    2. Example: “**RemoteCmd cmd=ChessMove from=e2 to=e4**” would send the command string “ChessMove from=e2 to=e4” over the network, and the remote computer would append to it and then execute the string “ChessMove from=e2 to=e4 remote=true” in its DevConsole (as if it had been typed there).