UNIVERSAL CHESS INTERFACE

SEPTEMBER 2024 · DRAFT

Introduction

This specification governs the interaction between two processes named the *client* and the *engine*. ※ Graphical interfaces, terminal emulators, and scripts and utilities are examples of clients. The text of sections 1–5 is normative (except as described in *Conventions* below).

Getting Started

This section needs to be written! It should contain enough for a beginner to add basic UCI support to their engine.

Conventions

An x preceding a number indicates that it is written in hexadecimal; for example, x10 = 16. Sequences of bytes encoded in ASCII are set in teal; for example, `uci` = ⟨x75, x63, x69⟩.

Special terms defined by the specification are set in purple when they are first introduced and inline comments are set ※ after a reference mark.

> A blue box is used to describe a convention that clients and engines are encouraged to follow, usually to do with the interpretation or meaning of the messages that clients and engines send. A blue box is also used to provide a recommendation, usually to do with implementation-defined behavior.

> A grey box is used to provide an explanative note or comment.

> A green box is used to provide an example of conforming behavior.

Text within colored boxes is nonnormative.

1 Definitions

1·1 A violation is any violation, by the client or engine, of the requirements of the specification. When a violation occurs, or when the requirements of the specification are otherwise not met, the specification imposes no further requirements on the behavior of the client or engine.

1·2 The engine's standard input and output must be open file descriptors and the engine's standard error must be an open file descriptor until closed by the engine. The sequence of bytes that the client sends to the engine via the engine's standard input is the client stream. The sequence of bytes that the engine sends to the client via the engine's standard output is the engine stream.

> There are no requirements for an engine's standard error except that it be open. For example, the engine's standard error may be directed to a null file, to the client's standard output or standard error, or to a log file. There are accordingly no restrictions on the sequence of bytes that the engine writes to its standard error.

> No particular encoding is specified: the client and engine streams are not required to be encoded sequences of Unicode scalar values (or code points of any other character set). The specification instead governs the client and engine streams as sequences of bytes *per se*.
>
> However, clients and engines are recommended to use UTF-8 for client–engine communication so that engine, author, and option names are displayed properly. Note that clients and engines may use different encodings for other interfaces, such as for file system paths that are passed as arguments to the operating system. For example, the client and engine may be communicating over a network connexion and the client may be running on Linux (where paths are arbitrary byte sequences that do not contain `x00` or `x2f`) but the engine may be running on Windows (for which NTFS is a common filesystem, which stores file names in UTF-16, but where older libraries or APIs may expect paths encoded in the local code page), and so the path of a tablebase file may require transcoding, implicitly or explicitly.

1·3 A message terminator is the pair ⟨`x0d`, `x0a`⟩ or `x0a` alone when it is not preceded by `x0d`.

1·4 Message terminators divide the client and engine streams into messages; that is, a message is a (possibly empty) subsequence of bytes that does not contain a message terminator and every byte of the client and engine streams is either part of a message or part of a message terminator. The messages within the client stream are client messages and the messages within the engine stream are engine messages.

> This precludes the proper use of some encodings, such as UTF-16 (since in UTF-16, the byte x0a appears in the encodings of various scalar values, whereas in UTF-8, the byte x0a only appears in the encoding for U+000A LINE FEED).

1·5 The byte x20 divides messages into tokens, that is, a token is a non-empty sequence of bytes that are not x20, and every byte of a message is either part of a token or is x20.

> In some cases, only the beginning of a message will be viewed as a collection of tokens and the remainder will be viewed as a contiguous sequence of bytes.

1·6 A position is a tuple with the following fields:

- An 8×8 array of elements, each of which is either NONE or a color–kind pair (where the color is white or black and the kind is king, queen, rook, bishop, knight, or pawn). This array is called the board and is indexed along one axis by the letters “a” through “h” inclusive and along the other axis by the numerals “1” through “8” inclusive. A color–kind pair is called a piece.
- A color, white or black, called the side to move.
- A collection of values, called the rights, which may be empty or include one or more of the following: the white kingside castling right, the white queenside castling right, the black kingside castling right, and the black queenside castling right.
- A value called the en passant target, which is either NONE or one of a3, b3, …, h3, or a6, b6, …, h6.
- A nonnegative integer called the depth from zeroing, which is 100 or less. ※ This is the number of moves that have been played (none of which are captures or pawn moves) since the last capture or pawn move, measured in ply.

The side waiting of a position is the opposite color of the side to move.

1·7 For a given position, a king is in check if it is attacked by one or more pieces of the opposite color, where “attacked” is defined analogously to article 3.1.2 of the 2023 FIDE Laws of Chess.

1·8 A position is valid if the following conditions are all satisfied:

The board contains exactly one white king and one black king.

The board does not contain any pawns at indices a1, b1, …, h1 and does not contain any pawns at indices a8, b8, …, h8.

If the rights include the white kingside castling right, the white king is at index e1 and there is a white rook at index h1. If the rights include the white queenside castling right, the white king is at index e1 and there is a white rook at index a1. If the rights include the black kingside castling right, the black king is at index e8 and there is a black rook at index h8. If the rights include the black queenside castling right, the black king is at index e8 and there is a black rook at index a8.

If the en passant target is *n*3 for some *n*, then the side to move is black, there is a white pawn at index *n*4, and at *n*2 and *n*3 the board is NONE. If the en passant target is *n*6 for some *n*, then the side to move is white, there is a black pawn at index *n*5, and at *n*6 and *n*7 the board is NONE.

The king of the color of the side waiting is not in check.

There is at least one valid move that can be applied (as described in 1·10 below). ※ This means the position is not checkmate or stalemate.

> A valid position (as defined above) need not be reachable from the starting position.
>
> Some engines designed for gameplay rather than analysis may additionally require that the positions they are sent must be reachable from the starting position. Such constraints are specific to engines, and are not constraints of the Universal Chess Interface.

> The 2023 FIDE Laws of Chess state that the game ends immediately when a player cannot checkmate the king by any series of legal moves, but this condition does not cause a position to not be valid (as defined above).

> Engines are recommended to accept positions that are checkmate or stalemate even though they are not required to handle receiving such positions. For such positions, engines should report the null move as the best move (see 1·15 below).

1·9 A move is either a tuple with two fields – a board index called the source and a board index called the destination – or a tuple with three fields: a source, a destination, and a kind called the promotion kind that is either queen, rook, bishop, or knight.

1·10 For a given position *P*, a move *M* is valid and the position *Q* immediately follows when *M* is applied if *P*, *M*, and *Q* fulfill requirements analogous to articles 3.1 through 3.9 inclusive of the 2023 FIDE Laws of Chess, except

> the requirement for a pawn to advance by two is instead that the pawn is white and its index is one of a2, b2, …, h2 or that the pawn is black and its index is one of a7, b7, …, h7, and
>
> the requirement for an en passant capture is instead that the en passant target is not none and the capturing pawn attacks the en passant target.

In particular, when *M* is valid, there is a piece of the color of the side to move in the board of *P* at the source of *M* and there is a piece of the same color and kind (or a piece of the same color and the promotion kind of *M*) in the board of *Q* at the destination of *M*.

The side to move of *Q* is the opposite color of the the side to move of *P*. If *M* is a king or rook move, the rights of *Q* do not include the corresponding castling right or castling rights. If *M* advances a pawn by two, the en passant target of *Q* is the index between the source and destination of *M*. If *M* is a capture or a pawn move, the depth from zeroing of *Q* is zero; otherwise, the depth from zeroing of *Q* is the depth from zeroing of *P* plus one.

1·11 A Forsyth–Edwards Notation (FEN) record is a sequence of six tokens ⟨*board*, *side to move*, *rights*, *EP target*, *DFZ*, *move number*⟩ of the form

```
        board = row / row / row / row / row / row / row / row
          row = 8 | [1–7KQRBNPkqrbnp]+
side to move = w | b
        rights = - | K?Q?k?q?
    EP target = - | [a–h][36]
          DFZ = 0 | [1–9][0–9]*
  move number = [1–9][0–9]*
```

※ The dash "-" is x2d.

with the following additional constraints:

In each *row*, numerals must not be adjacent; that is, there must not be two or more immediately consecutive numerals.

For each *row*, map `K`, `Q`, …, `p` to 1, map `1`, `2`, …, `8` to 1, 2, …, 8, and then define the width as the sum of these numbers. For each *row*, the width must be 8.

The bytes `K` and `k` must each occur exactly once in *board*.

The byte `P` must not occur in the first *row* and must not occur in the last *row*, and likewise for the byte `p`.

The *DFZ* token interpreted as a decimal integer must be 100 or less.

The *move number* token interpreted as a decimal integer must be less than 10000.

1·12 An FEN record describes a position if it maps to the position in a manner analogous to that described in article 16.1.3 of the Portable Game Notation Specification.

1·13 The starting position is the position described by the FEN record `rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1`.

1·14 A move token is a token of the form [a–h][1–8][a–h][1–8] or [a–h]7[a–h]8[qrbn] or [a–h]2[a–h]1[qrbn].

1·15 A null move token is a token of the form `0000`.

2 **Client Messages**

3 **Engine Messages**

4 **States and Transitions**

5 **Notation**

6 Examples

```
let read byte! : File Descriptor → Byte | EOF

def trimCR(seq : List(Byte)) : List(Byte)
  if empty?(seq) or last(seq) ≠ x0d then return seq
  return withoutLast(seq)

def read message!(fd : File Descriptor) : List(Byte) | EOF
  seq ← empty
  repeat
    match read byte!(fd)
      eof ⇒ return (if empty?(seq) then eof else seq)
      x0a ⇒ return trimCR(seq)
      val ⇒ append!(seq, val)

def get token!(seq : List(Byte)) → List(Byte) | None
  repeat
    if empty?(seq) then return none
    if first(seq) ≠ x20 then break
    removeFirst!(seq)
  tok ← empty
  repeat
    append!(tok, first(seq))
    removeFirst!(seq)
    if empty?(seq) then return tok
    if first(seq) = x20 then break
  repeat
    removeFirst!(seq)
    if empty?(seq) then return tok
    if first(seq) ≠ x20 then break
  return tok
```