This document is based on the Universal Chess Interface (UCI) document of April 2006 which was designed by Stefan Meyer-Kahlen the author of the Shredder chess program.

In October 2011 it was adapted to Khet 2 by Ruben Perez and Don Dailey.

Description of the Universal Khet Interface for Khet (UCI) Nov 2011
===================================================================

* The specification is independent of the operating system. For Windows, the engine is a normal exe file, either a console or "real" windows application.

* all communication is done via standard input and output with text commands,

* The engine should boot and wait for input from the GUI, the engine should wait for the "isready" or "setoption" command to set up its internal parameters as the boot process should be as quick as possible.

* the engine must always be able to process input from stdin, even while thinking.

* all command strings the engine receives will end with '\n', also all commands the GUI receives should end with '\n', Note: '\n' can be 0x0d or 0x0a0d or any combination depending on your OS. If you use Engine and GUI in the same OS this should be no problem if you communicate in text mode, but be aware of this when for example running a Linux engine in a Windows GUI.

* arbitrary white space between tokens is allowed

* The engine will always be in forced mode which means it should never start calculating or pondering without receiving a "go" command first.

* Before the engine is asked to search on a position, there will always be a position command to tell the engine about the current position.

* By default all the opening book handling is done by the GUI, but there is an option for the engine to use its own book ("OwnBook" option, see below)

* If the engine or the GUI receives an unknown command or token it should just ignore it and try to parse the rest of the string in this line. Examples: "joho debug on\n" should switch the debug mode on given that joho is not defined, "debug joho on\n" will be undefined however.

* If the engine receives a command which is not supposed to come, for example "stop" when the engine is not calculating, it should also just ignore it.

Move format:
The move format is in lower case algebraic notation, 'from' square followed by the 'to' square. If the move is a rotation, it will be just the square followed by upper case L for Left and R for right. Right rotation is clockwise.

Examples:  d3e4 e5f5 f5L c3R  etc ...

The squares for Khet are labeled very much as in chess, where the a-j are used to specify the files (from left to right) and (1-8) are used to specify ranks starting from the silver (first player) side of the board.

GUI to engine:
-------------

These are all the command the engine gets from the interface.

* uki

  tell engine to use the uki (universal khet interface), this will be sent once as a first command after program boot to tell the engine to switch to uki mode. After receiving the uki command the engine must identify itself with the "id" command and send the "option" commands to tell the GUI which engine settings the engine supports if any. After that the engine should send "ukiok" to acknowledge the uki mode. If no ukiok is sent within a certain time period, the engine task will be killed by the GUI.

* isready

  this is used to synchronize the engine with the GUI. When the GUI has sent a command or multiple commands that can take some time to complete, this command can be used to wait for the engine to be ready again or to ping the engine to find out if it is still alive. This command is also required once before the engine is asked to do any search to wait for the engine to finish initializing. This command must always be answered with "readyok" and can be sent also when the engine is calculating in which case the engine should also immediately answer with "readyok" without stopping the search.

* setoption name <id> [value <x>]

  this is sent to the engine when the user wants to change the internal parameters of the engine. For the "button" type no value is needed. One string will be sent for each parameter and this will only be sent when the engine is waiting. The name and value of the option in <id> should not be case sensitive and can include spaces. The substrings "value" and "name" should be avoided in <id> and <x> to allow unambiguous parsing, for example do not use <name> = "draw value".

Here are some examples:
  "setoption name Selectivity value 3\n"
  "setoption name Style value Risky\n"
  "setoption name Hash value 64"
* ukinewgame

this is sent to the engine when the next search (started with
"position" and "go") will be from a different game. This can be a
new game the engine should play or a new game it should analyse but
also the next position from a testsuite with positions only. If
the GUI hasn't sent a "ukinewgame" before the first "position"
command, the engine shouldn't expect any further ukinewgame
commands as the GUI is probably not supporting the ukinewgame
command. So the engine should not rely on this command even though
all new GUIs should support it. As the engine's reaction to
"ukinewgame" can take some time the GUI should always send
"isready" after "ukinewgame" to wait for the engine to finish its
operation.

* position [fen <fenstring> | classic | dynasty | imhotep ] moves <move1> .... <movei>

NOTE: fen is different for Khet and this is described later in this
documentation

set up the position described in fenstring on the internal
board and play the moves on the internal chess board. if the
game was played from one of the standard starting position of
t the string "classic", "dynasty" or "imhotep" will be sent

Note: no "new" command is needed. However, if this position is
from a different game than the last position sent to the
engine, the GUI should have sent a "ukinewgame" inbetween.

* go

start calculating on the current position set up with the "position" command.
There are a number of commands that can follow this command, all will be sent
in the same string.
If one command is not sent its value should be interpreted as it would not
influence the search.

* ponder

start searching in pondering mode. Do not exit the
search in pondering mode, even if it's mate! This means
that the last move sent in in the position string is
the ponder move. The engine can do what it wants to
do, but after a "ponderhit" command it should execute
the suggested move to ponder on. This means that the
ponder move sent by the GUI can be interpreted as a
recommendation about which move to ponder. However, if
the engine decides to ponder on a different move, it
should not display any mainlines as they are likely to
be misinterpreted by the GUI because the GUI expects
the engine to ponder on the suggested move.

* wtime <x>
white has x msec left on the clock
* btime <x>
black has x msec left on the clock
* winc <x>
white increment per move in mseconds if x > 0
* binc <x>
black increment per move in mseconds if x > 0

* movestogo <x>
  there are x moves to the next time control, this will only be sent if x > 0, if you don't get this and get the wtime and btime it's sudden death

* depth <x>
  search x plies only.

* nodes <x>
  search x nodes only,

* infinite
  search until the "stop" command. Do not exit the search without being told so in this mode!

* stop
  stop calculating as soon as possible, don't forget the "bestmove" and possibly the "ponder" token when finishing the search

* ponderhit
  the user has played the expected move. This will be sent if the engine was told to ponder on the same move the user has played. The engine should continue searching but switch from pondering to normal search.

* quit
  quit the program as soon as possible

Engine to GUI:
--------------
* id
  * name <x>
    this must be sent after receiving the "uki" command to identify the engine, e.g. "id name MyKet 3.1.1"

  * author <x>
    this must be sent after receiving the "uki" command to identify the engine, e.g. "id author Ozzie Puggermiller"

* ukiok
  Must be sent after the id and optional options to tell the GUI that the engine has sent all infos and is ready in uki mode.

* readyok
  This must be sent when the engine has received an "isready" command and has processed all input and is ready to accept new commands now. It is usually sent after a command that can take some time to be able to wait for the engine, but it can be used anytime, even when the engine is searching, and must always be answered with "isready".

* bestmove <move1> [ ponder <move2> ]
the engine has stopped searching and found the move <move> best in this position. the engine can send the move it likes to ponder on. The engine must not start pondering automatically. this command must always be sent if the engine stops searching, also in pondering mode if there is a "stop" command, so for every "go" command a "bestmove" command is needed! directly before that the engine should send a final "info" command with the final search information, the the GUI has the complete statistics about the last search.

* info
  the engine wants to send information to the GUI. This should be done whenever one of the info has changed.
  
  The engine can send only selected infos or multiple infos with one info command,
  e.g. "info currmove e2e4 currmovenumber 1" or
  "info depth 12 nodes 123456 nps 100000".
  Also all infos belonging to the pv should be sent together
  e.g. "info depth 2 score cp 214 time 1242 nodes 2124 nps 34928 pv e4R j4f5 b3L"

  I suggest to start sending "currmove", "currmovenumber", "currline" and "refutation" only after one second
to avoid too much traffic.

Additional info:
* depth <x>
  search depth in plies

* time <x>
  the time searched in ms, this should be sent together with the pv.
* nodes <x>
  x nodes searched, the engine should send this info regularly
* pv <move1> ... <movei>
  the best line found
* multipv <num>
  this for the multi pv mode.
  for the best move/pv add "multipv 1" in the string when you send the pv.
  in k-best mode always send all k variants in k strings together.
* score
  * cp <x>
    the score from the engine's point of view in centipawns.
  * mate <y>
    mate in y moves, not plies.
    If the engine is getting mated use negative values for y.
    Note: mate 1 means THIS move is a mate
    Note: mate -1 means we expect to get mated after this move.

* currmove <move>
  currently searching this move

* currmovenumber <x>
  currently searching move number x, for the first move x should be 1 not 0.

* hashfull <x>
the hash is x permill full, the engine should send this info regularly

* nps <x>
  x nodes per second searched, the engine should send this info regularly

* string <str>
  any string str which will be displayed be the engine,
  if there is a string command the rest of the line will be interpreted
  as <str>.

* option

  This command tells the GUI which parameters can be changed in
  the engine. This should be sent once at engine startup after
  the "uki" and the "id" commands if any parameter can be
  changed in the engine.

  The GUI should parse this and build a dialog for the user to
  change the settings.

  Note a few optikons are strictly defined but need not be
  honored by the engine such as "Hash." or "Ponder."

If the user wants to change some settings, the GUI will send a
"setoption" command to the engine. Note that the GUI need not
send the setoption command when starting the engine for every
option if it doesn't want to change the default value.

For all allowed combinations see the examples below,
as some combinations of this tokens don't make sense.
One string will be sent for each parameter.

* name <id>
  The option has the name id.

  The following two should always be implemented by your engine:

  * <id> = Hash, type is spin
    the value in MB for memory for hash tables can be changed,
    this should be answered with the first "setoptions" command at
    program boot
    if the engine has sent the appropriate "option name Hash"
    command,
    which should be supported by all engines!
    So the engine should use a very small hash first as default.

  * <id> = Ponder, type check
    this means that the engine is able to ponder.
    The GUI will send this whenever pondering is possible or not.
    Note: The engine should not start pondering on its own if this is
    enabled, this option is only
    needed because the engine might change its time management
    algorithm when pondering is allowed.

  * type <t>
    The option has type t.
    There are 5 different types of options the engine can send
    * check
      a checkbox that can either be true or false
    * spin
      a spin wheel that can be an integer in a certain range
* combo
  a combo box that can have different predefined strings as a value

* button
  a button that can be pressed to send a command to the engine

* string
  a text field that has a string as a value,
  an empty string has the value "<empty>"

* default <x>
  the default value of this parameter is x

* min <x>
  the minimum value of this parameter is x

* max <x>
  the maximum value of this parameter is x

* var <x>
  a predefined value of this parameter is x

Examples:

Here are 6 strings for each of the 5 possible types of options

"option name use nullmove type check default true\n"
"option name selectivity R factor type spin default 2 min 0 max 4\n"
"option name Style type combo default Normal var Solid var Normal var
Risky\n"
"option name Book Path type string default c:\n"
"option name Clear Hash type button\n"

FEN notation for Khet
---------------------

For Khet we will define a simple protocol for defining a Khet board position. For simplicity of parsing we will use 2 character fields per square, 80 squares starting from a8 in upper left corner to j1 in lower right corner. The first character specifies the piece type, and the second field describes its orientation.

Furthermore, an uppercase letter is used to describe white (silver) pieces and the red or "black" pieces are described with lower case letters.

  h -> Sphinx
  a -> Anubis
  p -> Pharoah
  y -> Pyramid
  s -> scarab

Orientations. In Khet all pieces have an orientation. We define them from the viewpoint of white, as if looking at them from the white (or silver) side of the board.

  u - up
  d - down
  l - left
  r - right

Note: orientation always in lower case.
In some cases the orientation is ambiguous. The scarab when locked into position is oriented diagonally and likewise the pyramid does not have a clearly defined "front."

So we define "up" for these two pieces to be the orientation that "aims" (one of) the mirrors in the Northwestern or upper left corner of the board. For the Scarab the "head" of the bug would be actually facing the upper right corner of the board but one side of the mirror would be facing the upper left corner. Also, for the scarab there is effectively only 2 relevant orientations.

So a fen string is 2 field string, 160 characters describing the position (2 characters per square) followed by exactly 1 space and then the lower case 'w' or 'b' character to specify color to move. 'w' is equivalent to Silver and 'b' correspods to the Red pieces. The orientation

There are no line breaks in a fen string but for the purpose of presentation we show how it might be hard coded into a java program. Note that spaces are designated with double hyphen characters.

Here are the 3 standard khet starting positions in Java Srting format:

```java
String classicFen = "hd------adpdadyd----" + 
"----yl--------------" + 
"------Yu------------" + 
"yr--Yl--srsd--yd--Yu" + 
"yd--Yu--SuSl--yr--Yl" + 
""""y-------yd-------" + 
""""--------------Yr----" + 
""""----YuAuPuAu------Hu" + " w";

String char dynastyFen = 
"hd------yladyd------" + 
"----------pd--------" + 
"yr------yladsd------" + 
"yd--sr--Yu--Yd------" + 
"------yu--yd--Sl--Yu" + 
"------SuAuYr------Yl" + 
"--------Pu----------" + 
"------YuAuYr------Hu w";

String imhotepFen = 
"hd------adpdadsu----" + 
"--------------------" + 
"------Yu----yr------" + 
"yrYl----Ydsu----ydYu" + 
"ydYu----Sdyu----yrYl" + 
"------Yl----yd------" + 
"--------------------" + 
"----SdAuPuAu------Hu w";

Examples:
Here is an example where indented lines are message from the engine and blank lines are just for clarity. I have added comments after the semicolon.

uki
    id name KhetMaster 5.3.2
    id author John Doe
    option name Hash type spin default 64 min 1 max 1024
    option name use selectivity type check default true
    option name Ponder type check default false
ukiok

ukinewgame   ; no response expected

ukisread     ; like a ping command
    ukiock   ; engine responds

position startpos classic moves c1R a4L    ; no response required

go depth 5   ; begin a 5 ply search (but don't execute move on board)
    info depth 1 time 0 nodes 34 score cp 40 nps 34002 pv b1c3
    info depth 2
    info depth 3
    info depth 4
    info depth 5
    info depth 5 time 4 nodes 1832 score cp 9 nps 366409 pv d2d4 g8f6 b1c3 d7d5 g1f3 b8c6
    info depth 5 time 6 nodes 2550 score cp 33 nps 424997 pv b1c3
    info depth 5 time 6 nodes 2779 score cp 35 nps 463163 pv b1c3 e7e6 d2d4 g8f6 e2e4 b8c6
    info depth 5 time 6 nodes 2842 score cp 35 nps 473663 pv b1c3 e7e6 d2d4 g8f6 e2e4 b8c6
    info nodes 2842
    bestmove b1c3     ; OR "bestmove b1c3 ponder e7e6"

NOTE: I used chess output here for illustration purpose, but the notation will look different for Khet