## **PL/SQL Chess Engine Package Documentation**

### **I. PL\_PIG\_CHESS\_ENGINE Package: Core Engine Functionality**

This package contains the core logic of the chess engine, handling move generation, validation, and position manipulation.

#### **A. Position Management**

* **STILLING\_TO\_EPD Function:** Converts the internal position representation to a string in EPD format, allowing for additional EPD operations.
* **FEN\_EPD\_TO\_STR Function:** Converts a position in FEN or EPD format to the internal string representation (POSITIONSTR).
* **still Procedure:** Sets up a position from a POSITIONSTR string, handling FEN, EPD, and English notation.

#### **B. Move Operations**

* **DoMoveOk Function:** Checks if a move is legal and generates possible black/white moves, returning a boolean value.
* **DoMoveC Procedure:** Executes a move without move type validation, assuming the move is already validated.
* **DoMove Procedure:** Executes a move with move type information, validating it before execution.
* **GetNext Procedure:** Finds and returns the next legal move in the position.

#### **C. Position Manipulation**

* **Mirror Procedure:** Mirrors the current position, swapping board layout and colors.

#### **D. Search and Move Retrieval**

* **FindTrk Procedure:** Finds a move at a specified depth within the search tree.
* **GetMove Procedure:** Makes a move based on a given move number, with an option to skip check validation.
* **GetMoveNr Procedure:** Retrieves the move number corresponding to a specific move, with an option to skip check validation.

#### **E. Initialization**

* **Initialize Procedure:** Initializes the chess engine and sets up necessary data structures.

### **II. PL\_PIG\_CHESS\_ENGINE\_EVAL Package: Position Evaluation**

This package focuses on evaluating chess positions, using a data structure called pd to store positional information.

#### **A. Data Access**

* **pdN Function:** Returns the index of a piece in the pd array using its numerical representation and board position.
* **pdX Function:** Returns the index of a piece in the pd array using its character representation and board position.

#### **B. Initialization and Pre-processing**

* **Initialize Procedure:** Allocates and initializes data structures for the evaluation package.
* **PreProcess Procedure:** Sets the pd array to default values, performed once per engine call.
* **PreProcessor Procedure:** Adjusts the pd array and game states based on the current position, performed once per engine call.

#### **C. Position Evaluation**

* **Eval Function:** Evaluates the current position using the pd array and position data, called numerous times per engine call.

### **III. PL\_PIG\_CHESS\_INTERFACE Package: User Interface and Testing**

This package provides an interface for interacting with the chess engine, playing games, and running test suites.

#### **A. Game Management**

* **NEW\_GAME Procedure:** Initializes a new game, setting engine strength, starting position, and game modes.
* **DO\_MOVE Procedure:** Allows a user to input a move in standard algebraic notation.
* **DO\_BOTMOVE Procedure:** Executes a move for the engine, with an option to override the engine's choice.
* **DO\_BOTGAME Procedure:** Plays a full game against the engine up to a specified move limit.

#### **B. Player Control**

* **SET\_White Procedure:** Sets the player type (engine or human) for white.
* **SET\_Black Procedure:** Sets the player type (engine or human) for black.

#### **C. Move Correction (Not Implemented)**

* **TAKEBACK\_MOVE Procedure:** Intended to undo the last move.
* **TAKEBACK\_MOVES Procedure:** Intended to undo the last two moves.

#### **D. Test Suites**

* **Test Suite Procedures (test\_BKtest, test\_MSquickTest, etc.):** Execute various test suites with customizable parameters.
* **test1 and test2 Procedures:** Specific functionality not described in the provided documentation.