\*Using the convention whereby MM = Matchmaking Server.

# Operation Models for Player-System Interactions

## User requests to login

**Operation:** MW::requestLogin(username:String, password:String)

**Description:** A user wants to login using the provided credentials.

**Scope:** User, MM

**Messages:** User::{RequestLogin}

MM::{LoginResult}

**New:**

**Pre:**

**Post:** The user sends the username and password via a RequestLogin message to the MM which verifies the pair.

If the MM outputs login-success as the message, then the user is logged into the MM.

If the MM outputs login-failure\_e as the message, the User is notified and prompted to re-enter their credentials.

## User requests to create account

**Operation:** MW::requestAccount(username:String, password:String)

**Description:** A user wants to create an account in the MM.

**Scope:** User, MM

**Messages:** User::{RequestAccount}

MM::{AccountResult}

**New:** Account

**Pre:**

**Post:** The user sends the username and password via a RequestAccount message to the MM which verifies the pair.

If the MM outputs an account-success message, the account is created in the MM.

If the MM outputs an account-failure\_e message, either the format is incorrect or the username already exists, in which case the user will need to provide another username.

## User requests to join game lobby

**Operation:** MW::joinLobby(lobbyName:String)

**Description:** A user wants to join a game lobby.

**Scope:** User, MM, GameSetup

**Messages:** User::{RequestJoin}

MM::{LobbyInfo}

**New:**

**Pre:**

**Post:** The user sends a RequestJoin message with a lobbyName to the MM.

If the MM outputs a valid-lobby message, then the operation adds the player to the lobby.

If the MM outputs a lobby-full message, the player cannot join the lobby.

If the MM outputs a lobby-does-not-exist\_e message, the player cannot join the lobby.

## User requests to host game lobby

**Operation:** MW::hostLobby(lobbyName:String)

**Description:** A user wants to host a game lobby.

**Scope:** User, MM, GameSetup

**Messages:** User::{RequestHost}

MM::{HostLobbyResult}

**New:** Lobby

**Pre:**

**Post:** The user sends a RequestHost message and a lobby name to the MM.

If the MM outputs a lobby-created message, the MM has created a lobby named lobbyName, and the operation updates the GameSetup, making the user the host.

If the MM outputs a lobby-not-cleared\_e message, then the user cannot create a lobby at this time or lobbyName has incorrect format.

## User requests to configure game

**Operations:** MW::changeSettings(setting: GameSettings)

**Description:** The host wants to change the game's settings (map, etc.).

**Scope:** Host, GameSetup, Players

**Messages:** GameSetup::{SettingsResult}

PlayersDisconnected::{List:String}

**New:**

**Pre:** Host is in the lobby and has not declared himself as ready.

**Post:** The host sends a RequestConfigurationChange message to the GameSetup, to either change settings, change map or clear all configuration settings (including a loaded file, if there is one). All Players are notified of the change.

If the GameSetup outputs a valid-settings message, the operation either changes the settings or clears all settings.

If the GameSetup outputs a game-loaded-from-save\_e message, the host will need to clear the loaded file before changing settings.

If the PlayersDisconnected contains any items, the User is notified of which Player(s) disconnected.

**Operation:** MW::loadGame(gameFile:GameState)

**Description:** A user wants to host a game from a saved game state file.

**Scope:** Host, GameSetup, Player

**Messages:** Host::{LoadFile}

GameSetup::{LoadResult}

PlayerDisconnectedMessage::{List:String}

**New:**

**Pre:** Host is in the lobby.

**Post:** The host sends a LoadFile message to the GameSetup to load the game configuration settings from a save file.

If the GameSetup outputs a load-success message, the GameSetup is successfully updated to the gameFile's game state.

If the GameSetup outputs a load-failed\_e message, then the save file does not exist or it cannot be made into a valid game state.

If the PlayerDisconnectedMessage contains any items, the User is notified of which Player(s) were disconnected.

## User plays a turn

**Operation:** MW::makeMove(move:UserMove)

**Description:** A player wants to make a move.

**Scope:** Player, Host, Game

**Messages:** Player::{RequestValidMove}

Host::{MoveResult}

**New:**

**Pre:** Game is in progess.

**Post:** A player sends a RequestValidMove message to validate a move with the host.

If the host outputs a valid-move message, the operation updates the game state, notifying all players in the game of the game state changes.

If the host outputs an invalid-move\_e message, the player will have to revise his move.

**Operation:** MW::saveGame(fileName:String)

**Description:** A Player wants to save the game state on his local machine.

**Scope:** Player, Host, Game

**Messages:** Player::{RequestSaveGame}

Host::{SaveResult}

**New:** saveFile:File

**Pre:** Game is in progress.

**Post:** A player sends a RequestSaveGame message to the host to save the game state on his local machine.

If the host outputs a save-success message, the operation saves the game state into a file on the player's local machine.

If the host outputs a save-failure\_e message, the player will have to try again another time.

**Operation:** MW::endTurn()

**Description:** A player ends his turn.

**Scope:** Player

**Messages:** Player::{TurnEnded}

**New:**

**Pre:** Player's turn.

**Post:** The operation ends a player's turn, sending a TurnEnded message to all other players so they can update their game state accordingly.