Communication Protocol

When the communication is initialized the server send an *askLogin* message, to which the client answers with a *MessageLogin* (serialized Java Class). Meanwhile the *ClientSocket* sends to the server a *MessagePing* every 30 seconds.

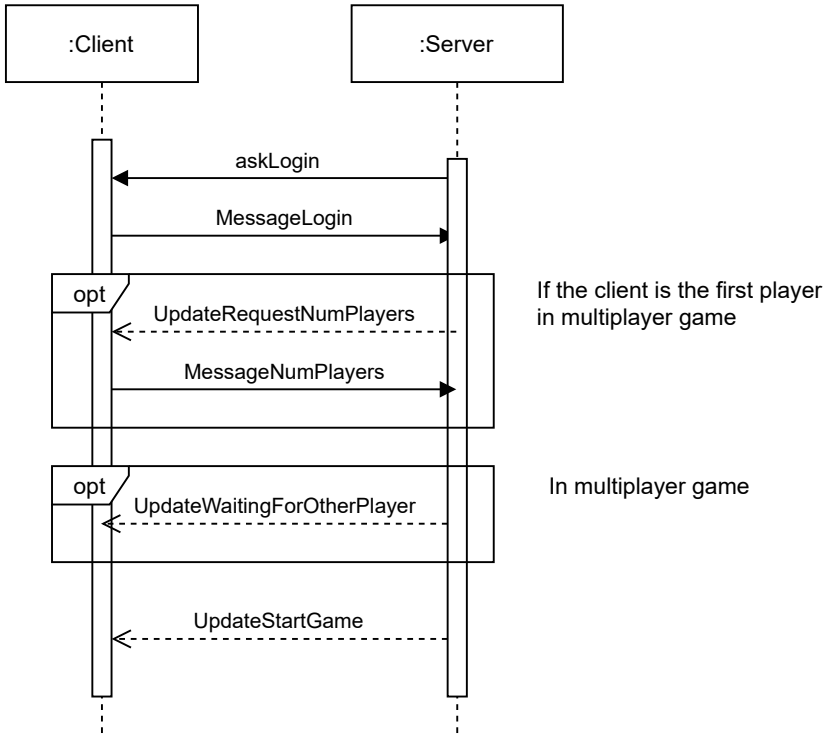
All the message exchanged between client and server are serialized as Java Class (always).

The whole communication can be divided into three phases (described below), each one ending with a *MessageEndTurn* from the client.

- Initialization

Login phase

In this phase the client sends the *server IP address* and the *server port* to connect to the server. If they are correct, then the client inserts his *nickname* and, if the current player is the first one, the server asks him how many players will play. This is the only part of the communication ruled from the server.



MessageLogin (client):
client sends the *nickname* to the server

UpdateRequestNumPlayers (server):
if the client is the first player, server requests to him to indicates the *number of players*

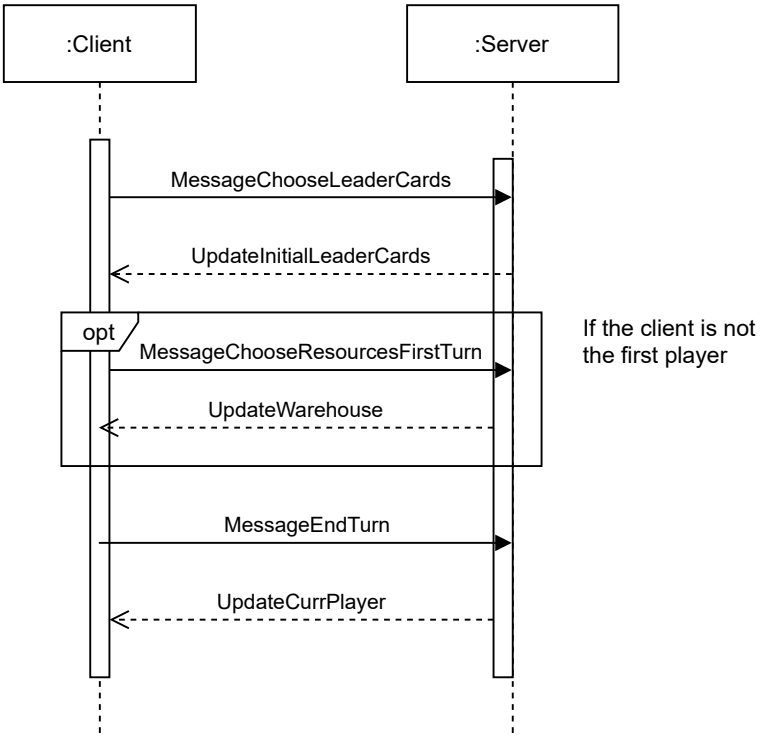
MessageNumPlayers (client):
client sends an *int* with the *number of players*

UpdateWaitingForOtherPlayer (server):
if the client is not the first player, server sends a stand-by message

UpdateStartGame (server):
server sends a semplified version of the Model: the Leader cards to choose, the Development cards deck, the Market tray and the remaining Marble, all converted as String type

Initial turn

In this phase the player chooses two Leader cards from four. If he is the second or the third player, he can choose a resource or, if he is the fourth, he can choose two resources. Eventually he has to sends a *MessageEndTurn* to terminate this initial turn. When all player end this phase, they can begin the *Action phase*.



MessageChooseLeaderCards (client):
client sends the indexes (*int*) of the chosen Leader cards

UpdateInitialLeaderCards (server):
if the previous message is correct, server sends a *list of String* with the chosen cards'ID

MessageChooseResourcesFirstTurn (client):
if the client is not the first player, he can send a *list of String* with the names of the chosen Resource(s)

UpdateWarehouse (server):
if the previous message is correct, server sends the client's Warehouse as a *matrix of String* and,
if any, the Extra chest as a *Map<String ,Integer>* and a *boolean* that, if it's true, indicates if the client has to remove a Marble from its local buffer

MessageEndTurn (client):
client notifies to server that its turn is ended

UpdateCurrPlayer (server):
server updates all players with the name (*String*) of new current player

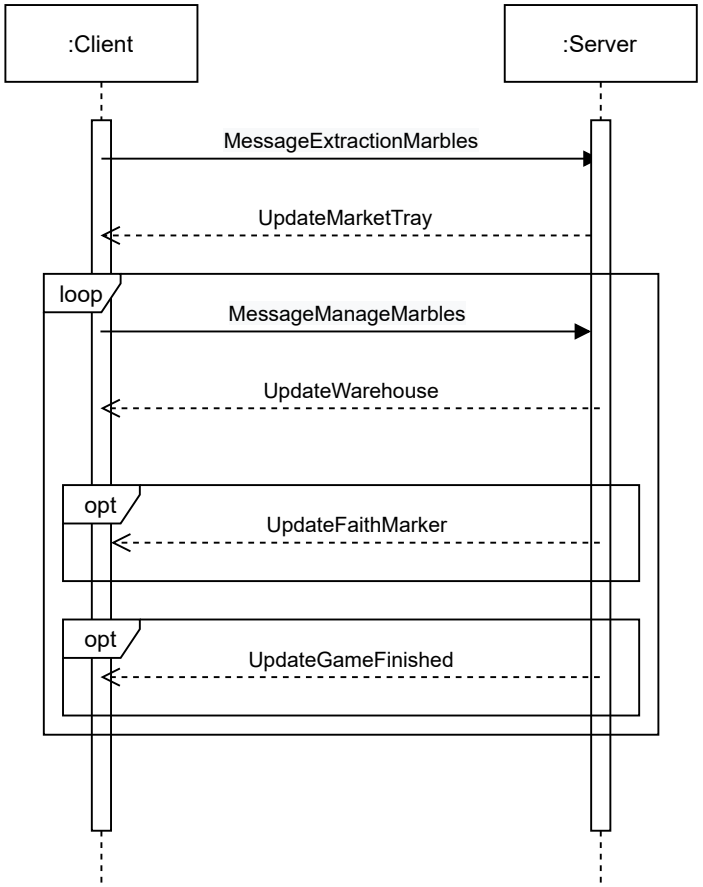
Game phase

After all players have chosen their Leader cards, client can do one of these three actions:

- extract Marbles from the Market tray;
- buy a Development card;
- activate a production (Base production, Development card production or Leader card production);

Sooner or later each of these actions client can activate or discard a Leader card.
Eventually client has to send the *MessageEndTurn* to end his turn.

Extraction Marbles



MessageExtractionMarbles (client):
client sends a *char* that indicates if he wants to extract a column (*c*) or a row (*r*)
and an *int* with the numeber of column or row

UpdateMarketTray (server):
server sends a matrix of String that represents the updated Market tray

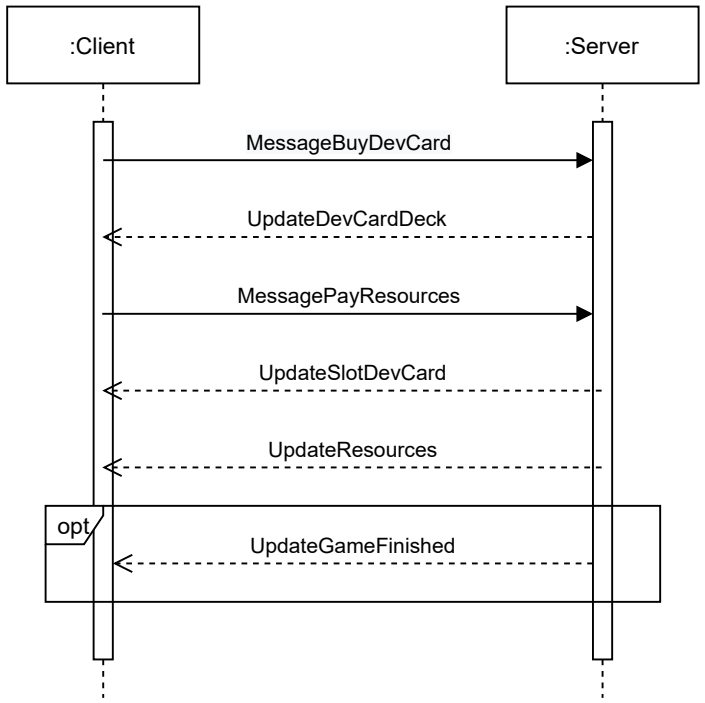
MessageManageMarbles (client):
client sends a *char* that represents the structure in which he wants to add
the converted resource ('W' for Warehouse and 'E' for Extra chest) or 'D' to discard the marble;
an *interger* that represents the row of the Warehouse; a *String* that represents the resource in which
he wants to convert a white marble, if he can

UpdateWarehouse (server):
if the previous message is correct, server sends the client's Warehouse as a *matrix of String* and,
if any, the Extra chest as a *Map<String ,Integer>* and a *boolean* that, if it's true, indicates if the client has
to remove a Marble from its local buffer

UpdateFaithMarker (server):
server sends a Map<String, Integer> that indicates the players' positions; a Map<String, boolean[]>
that indicates for each player the state of his Pop favorite tiles. Instead, in Single Player mode it sends
the position of the Black cross token

UpdateGameFinished (client):
if the client extracts a Red marble or discard a marble the server notifies all clients that the game is finished

Purchase Development card



MessageBuyDevCard (client):
client sends the card's ID (*String*) and the column (*int*) of the Slot dev cards in which he wants to add the card

UpdateDevCardDeck (server):
server sends to all clients the chosen card's ID to update their DevCardsDeck

MessagePayResources (client):
client sends three *Map*<*String, Integer*>: one for the Warehouse, one for the Strongbox and one another for the Extra chest

UpdateSlotDevCard (server):
server sends the card's ID (*String*) and and the column (*int*) of the Slot dev cards to update the local version of the DevCardDeck and SlotDevCard

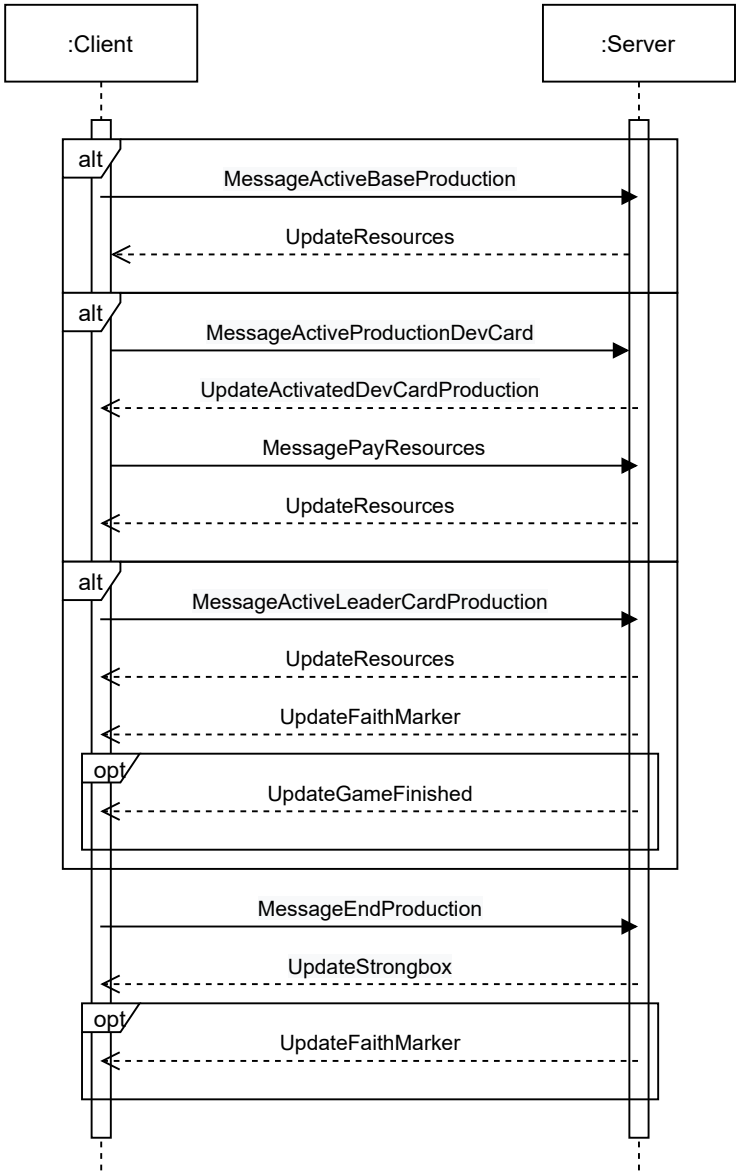
UpdateResources (server):
server sends the updated version of Warehouse, Extra chest and Strongbox converted to String and Integer

UpdateGameFinished (client):
if the client extracts a Red marble or discard a marble the server notifies all clients that the game is finished

Production

The client can do only one of these productions:

- Base production;
- production of Development card;
- production of Leader card



MessageActiveBaseProduction (client):
client sends to server the information about the two resources to exchange with one other and from which he wants to take these resources

UpdateResources (server):
server sends the updated version of Warehouse, Extra chest and Strongbox converted to String and Integer

MessageActiveProductionDevCard (client):
client sends to server the column (*int*) of the Slot dev cards of the chosen Development card

UpdateActivatedDevCardProduction (server):
server sends the card's ID

MessagePayResources (client):
client sends three *Map<String, Integer>*: one for the Warehouse, one for the Strongbox and one another for the Extra chest

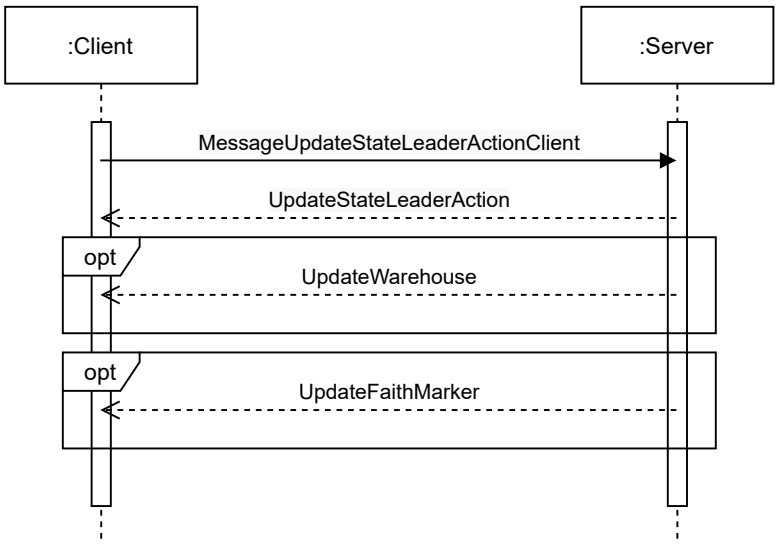
MessageActiveLeaderCardProduction (client):
client sends the card's ID, the structure from which he wants to take the resource indicated on the card and the chosen resource

UpdateFaithMarker (server):
server sends a Map<String, Integer> that indicates the players' positions; a Map<String, boolean[]> that indicates for each player the state of his Pop favorite tiles. Instead, in Single Player mode it sends the position of the Black cross token

UpdateGameFinished (client):
if the client comes to the end of the Faith track

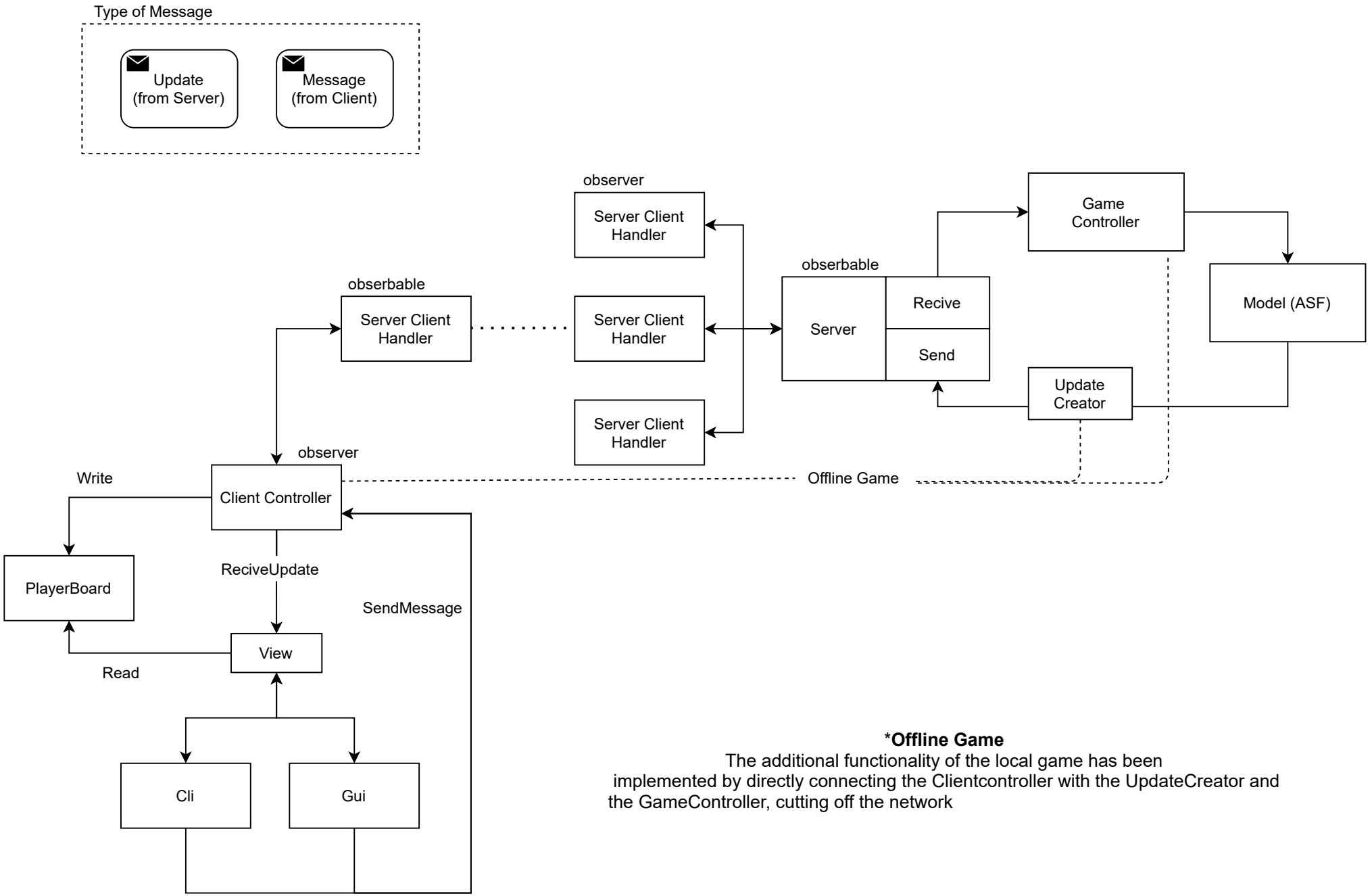
MessageEndProduction (client):
client notifies that he wants to end the Production phase
UpdateStrongbox (server):
server sends a *Map<String, Integer>* that represents the Strongbox

Activate or discard a Leader card



MessageUpdateStateLeaderActionClient (client):
client sends the card's ID and 0 or 1 if he wants to discard or activate the card, respectively
UpdateStateLeaderAction (server):
server sends the card's ID and a boolean that indicates if that card has been activated or discarded
UpdateWarehouse (server):
if the previous message is correct, server sends the client's Warehouse as a *matrix of String* and, if any, the Extra chest as a *Map<String ,Integer>* and a *boolean* that, if it's true, indicates if the client has to remove a Marble from its local buffer
UpdateFaithMarker (server):
server sends a Map<String, Integer> that indicates the players' positions; a Map<String, boolean[]> that indicates for each player the state of his Pop favorite tiles. Instead, in Single Player mode it sends the position of the Black cross token

Network diagram



***Offline Game**

The additional functionality of the local game has been implemented by directly connecting the Clientcontroller with the UpdateCreator and the GameController, cutting off the network