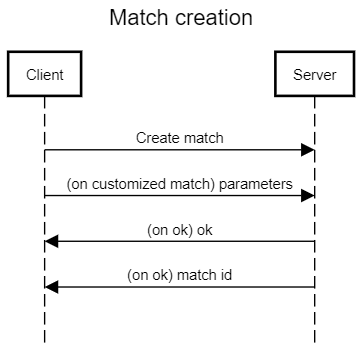
**MATCH CREATION/JOIN**

When a user starts the game, he must choose if he wants to create a new game, or if he wants to join a game of other players.

If he wants to create a new game, he signals it to the server, which replies with “ok\n” and the match id.



**create match request (client)**

create match\n

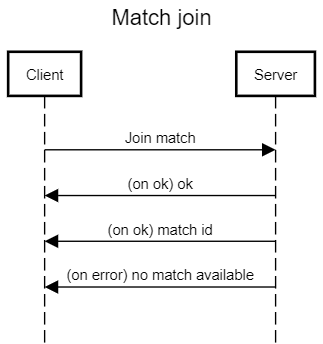
parameters.json\n

**on ok ack (server)**

ok\n

[value1]\n

Instead, if the player wants to join an already existing game, he has 2 possibilities: join a game he knows the ID of, or join a random game.



**Join game request (client)**

Join a random game request \n

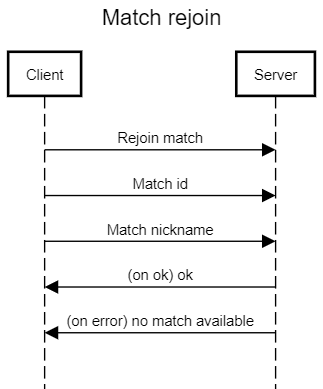
**on available room (server)**

“Game\_ID” \n

**on error (server)**

No rooms available, try again later\n

User also has the possibility to rejoin a match he disconnected from, inserting that game ID and the nickname he was using for that game.



**Rejoin game request (client)**

Rejoin game\n

GameID\n

Nickname in that game\n

**on available room (server)**

ok \n

**on error (server)**

No match with those credentials\n

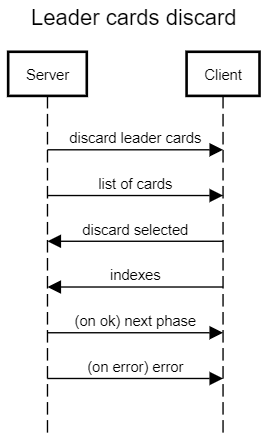
**MATCH INITIALIZATION**

This phase is composed by all the interaction between client and server at the beginning of the game. It includes the login, the resource choice and the leader cards choice.

All the message exchanged between client and server are plain text (always), and everyone of them ends with a new line character "\n".

Leader Cards Choice Phase

Before the effective start of the game, all the players must select 2 of the 4 cards given (this number might change if the creator of the game sets a different amount) , discarding the other 2. The server asks the client to select 2 card to discard, giving the list of the 4 cards the user can choose from, and after the user selected the cards, the server replies with “ok\n” if everything is correct, else with “error\n”, “you selected an incorrect amount of cards”.



**leader cards removal request (server)**

discard 2 of these leader cards\n

[[card1], [card2], [card3], [card4]]\n

**leader cards reply (client)**

[[index1], [index2]]\n

**on ok ack (server)**

ok\n

next phase (resource choice or waiting room)\n

**on error (server)**

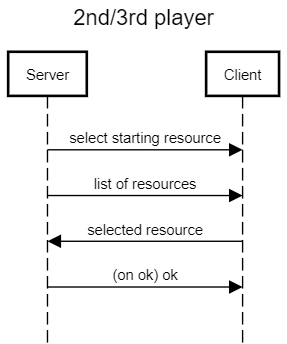
error\n

you selected an incorrect amount of cards

Resource Choice Phase

Since all the player have extra resources at the beginning of the game based on their order, server must request those players to choose resources to add to their warehouse.

If the player is 2nd or 3rd the server asks for a single extra resource, sending the client a list of resources to choose from, the client simply replies with its choice, and server replies again sending “ok\n” just to notify the client that the interaction ended successfully.



**resource request (server)**

select a resource to start with\n

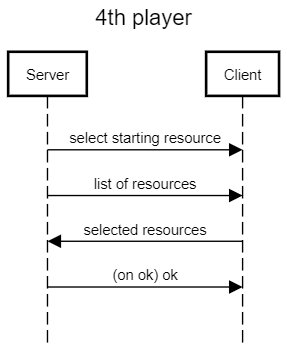
[[coin], [stone], [shield], [servant]]\n

**resource reply (client)**

resource\n

**resource ack (server)**

ok\n

The exact same thing happens when player turn is 4th, with the difference that the client must choose a resource 2 times.

**resource request (server)**

select 2 resources to start with\n

[[coin], [stone], [shield], [servant]]\n

**resource reply (client)**

[[resource1], [resource2]]\n

**resource ack (server)**

ok\n

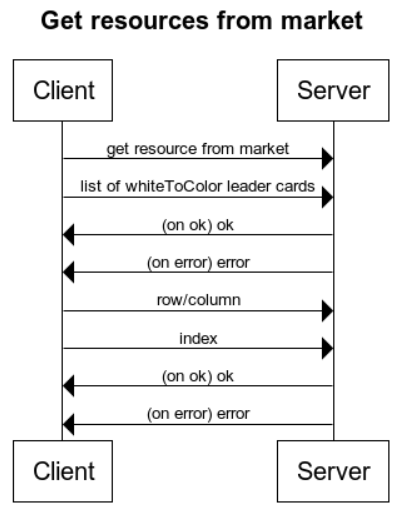
**MAIN OPERATIONS**

This group of actions is composed by the 3 main operations that require an interaction with the gameboard, the component shared among all players. To avoid the risk of bugs, whenever a player interacts with the gameboard (by buying a dev card or using the market), this actions is notified to all players, so that they can change their own copy of the gameboard to stay up to date with the progress of the game.

***Getting resource from Market***

This phase has 2 possible interaction based on the amount and type of leader cards activated by the user. The first one is the standard one, used in 99% of the cases, the second one is used when the player has activated 2 leader cards with WhiteToColor power effect, because he has to select a card for each blank resource in the selected line.

In the standard case client asks server to make a “get resource from market” operation, and server checks if it is possible (user has less than 2 whiteToColor and didn’t make any other operation in this turn). If it is ok client sends an index of column/row and server checks if it chose a possible index.

**get resources from market request (client)**

get resources from marker\n

[[card1], [card2]]\n

**on ok ack (server)**

ok\n

**on error (server)**

error\n

you have 2 white to color\n

you must select another method\n

**on error (server)**

error\n

you have done already a main operation

**get resources from market details (client)**

[[row], [column]]\n

[value1]\n

**on ok ack (server)**

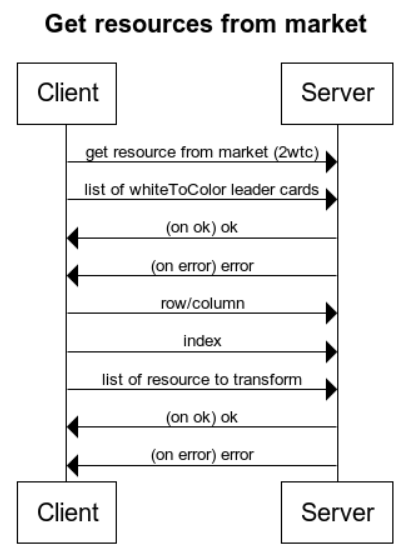
ok\n

**on error (server)**

error\n

incorrect index\n

In the rare case when the player has 2 white to color leader cards, the interaction is very similar with the addition of a list of resources in the details sent by the client.



**get resources from market request (client)**

get resources from marker\n

[[card1], [card2]]\n

**on ok ack (server)**

ok\n

**on error (server)**

error\n

you have less than 2 white to color\n

you must select another method\n

**on error (server)**

error\n

you have done already a main operation

**get resources from market details (client)**

[[row], [column]]\n

[value1]\n

[[resource1], …, [resource4]]\n

**on ok ack (server)**

ok\n

**on error (server)**

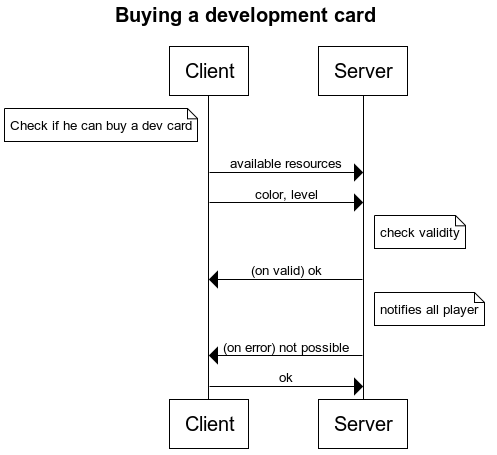
error\n

incorrect index or\n

incorrect amount of resources\n

***Buying a development card***

When a player decides to buy a development card, he must check locally whether he has enough resources to complete the acquisition. If this condition is fulfilled, he communicates to the server his card of choice. The server then proceeds to check himself if the client has in fact the resources needed; if he does, the acquisition is notified to all players; if he doesn’t, the server tells him to try again or to do another action.



**Dev card request (client)**

[[“coin”,“value1”], [“shield”,”value2”], [“servant”,”value3”], [“stone”,”value4”]] \n

[[“color”,”level”]\n

**On ok (server)**

Successful acquisition \n

Notifies all players

**On error (server)**

Acquisition impossible, try something else \n

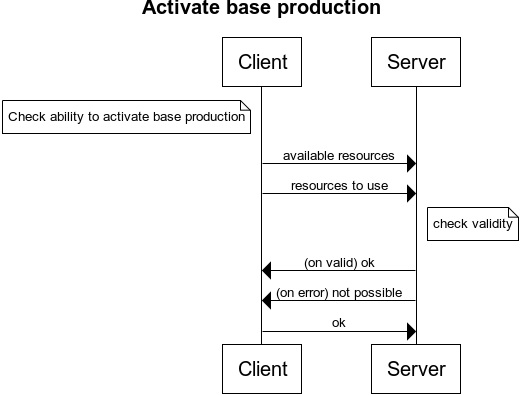
**Client reply (client)**

Ok \n

***Production***

These are the three possible operations a player can use to activate a production. These are very similar; the only difference is the object that gets activated.

Base production activation



**Base prod request (client)**

[[“coin”,“value1”], [“shield”,”value2”], [“servant”,”value3”], [“stone”,”value4”]] \n

[[“resource1”, ”resource2”]\n

**On ok (server)**

Successful production \n

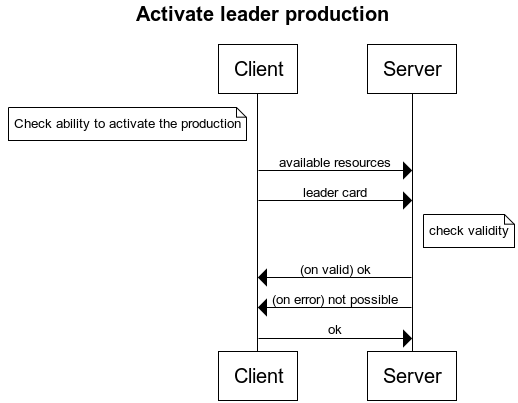
**On error (server)**

Production impossible, try something else \n

**Client reply (client)**

Ok \n

Leader production activation

****

**Base prod request (client)**

[[“coin”,“value1”], [“shield”,”value2”], [“servant”,”value3”], [“stone”,”value4”]] \n

[“LeaderCard”]\n

**On ok (server)**

Successful production \n

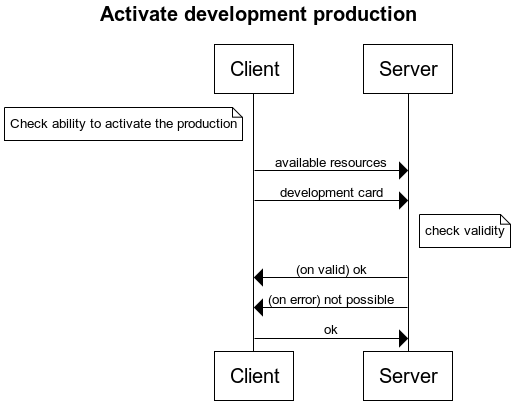
**On error (server)**

Production impossible, try something else \n

**Client reply (client)**

Ok \n

Development production activation

****

**Base prod request (client)**

[[“coin”,“value1”], [“shield”,”value2”], [“servant”,”value3”], [“stone”,”value4”]] \n

[“DevelopmentCard”]\n

**On ok (server)**

Successful production \n

**On error (server)**

Production impossible, try something else \n

**Client reply (client)**

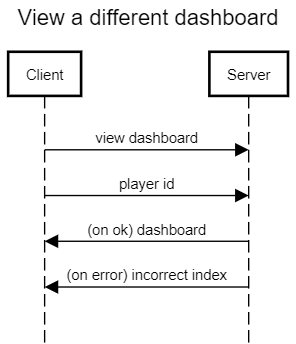
Ok \n

**INTERACTION OPERATIONS**

These are the operations that require an interaction not only with the server, but with other players as well.

Action Choice Phase

Before the choosing what to do, a player might want to get some infos about his opponents. To do so checking their dashboard is essential: by giving the server a player’s turn order (which identifies them univocally), the server asks that player his dashboard, and immediately return it to the caller.



**Dashboard request (client)**

dashboard request\n

[“player order”] \n

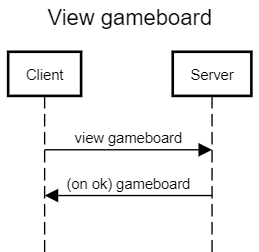
**Dashboard ack (server)**

[Dashboard] \n

**Dashboard error (server)**

[“Incorrect index”] \n

Every time a player wants to buy a development card, he might want to check all the available cards in the gameboard, asking server for them.



**Gameboard request (client)**

gameboard request\n

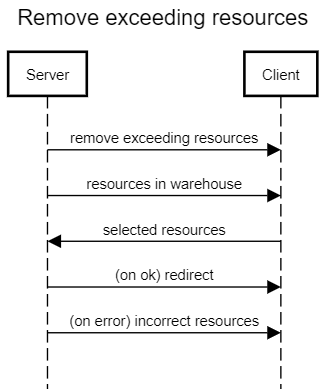
**Gameboard ack (server)**

[Gameboard] \n

**INTERNAL OPERATIONS**

Remove a resource from warehouse

This interaction is used when a player must remove a resource from his warehouse in order to end his turn. His client sends the index of the depot to remove from to the server, which checks if the index is valid.



**Remove exceeding resources request (server)**

remove exceeding resources\n

[warehouse]\n

**Remove selected resources (client)**

[resources]\n

**On ok (server)**

redirect to dashboard\n

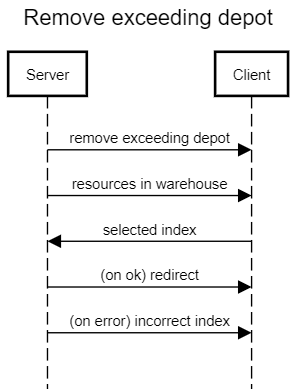
**On error (server)**

error\n

incorrect resources chosen\n

Remove a depot from warehouse

Basically the interaction is identical to the resource removal



**Remove exceeding depot request (server)**

remove exceeding depot\n

[warehouse]\n

**Remove selected depot (client)**

[index]\n

**On ok (server)**

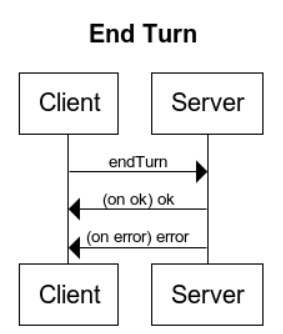
redirect to dashboard\n

**On error (server)**

error\n

incorrect index chosen\n

**END TURN**

****

**end turn request (client)**

remove depot\n

**On ok (server)**

ok \n

**On error (server)**

error\n

you didn’t make any main operation\n