MessageClass:Message

**Message**

-messageType: String
-senderNickName: String

+getMessageType(): String
+getSenderNickname(): String

MESSAGES CONTENT:

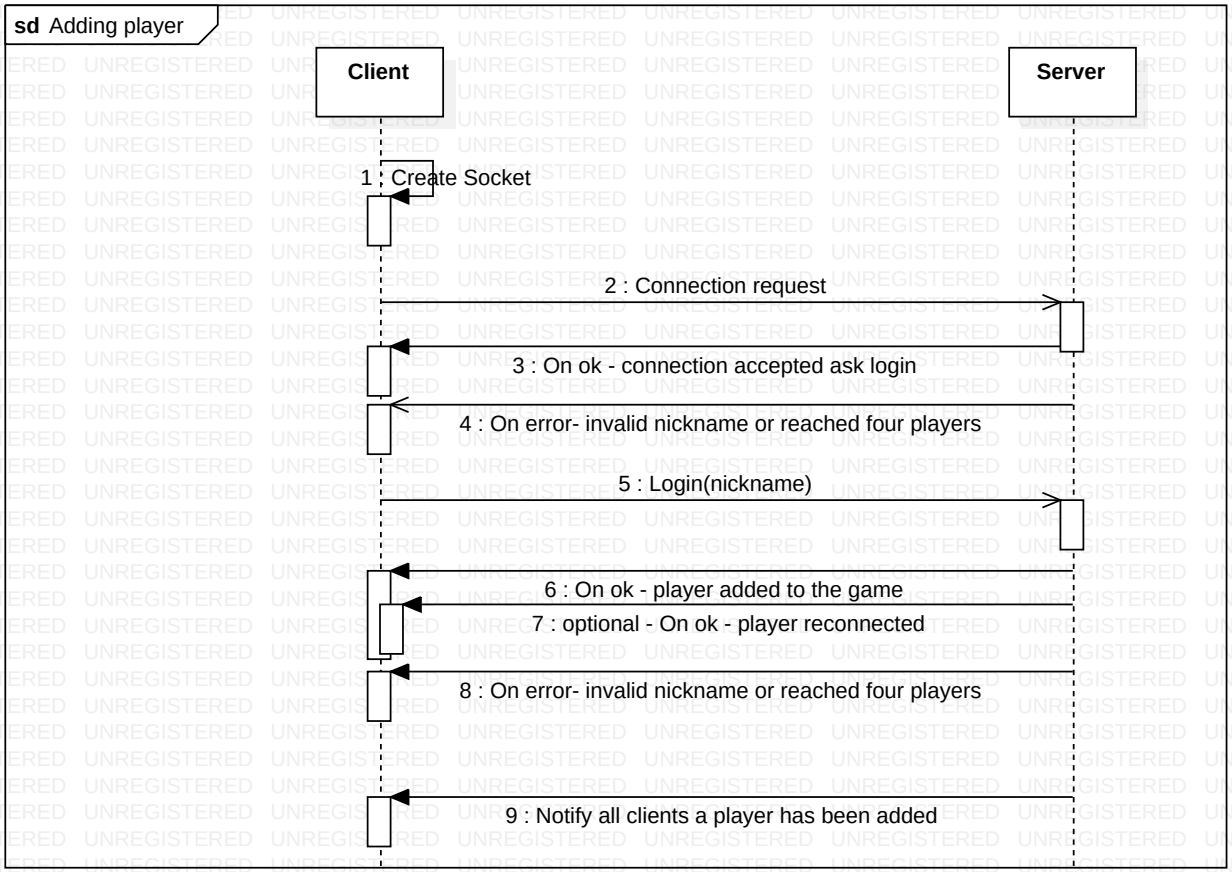
-messageType: String
-senderNickName: String
-warehouseConfiguration: Warehouse
-resourcesToDistribute: ResourceType[2]
-leaderCardPositions: int[2]
-choosableLeaderCards: int[4]
-chosenLeaderCards: int[2]
-actualCurrentPlayer: String
-levelsToSwitch: int[2]
-marketPosition: int[2]
-temporaryResources: Map <ResourceType,Integer>
-resourceToInsert: ResourceType
-quantityToInsert: int
-intoExtraDeposit: boolean
-newFaithPoints: int
-devolpmentCardConfiguration: DevelopmentCard [3]
-extraDepositConfiguration: ExtraDeposit[2]
-strongboxConfiguration: Map <ResourceType,Integer>
-payUsingExtraDeposit: int[2]
-remainingWhiteMarbles: int
-actionCardConfig: ActionCardStack
-deckgrid: Deckgid

NOTE GENERALI:

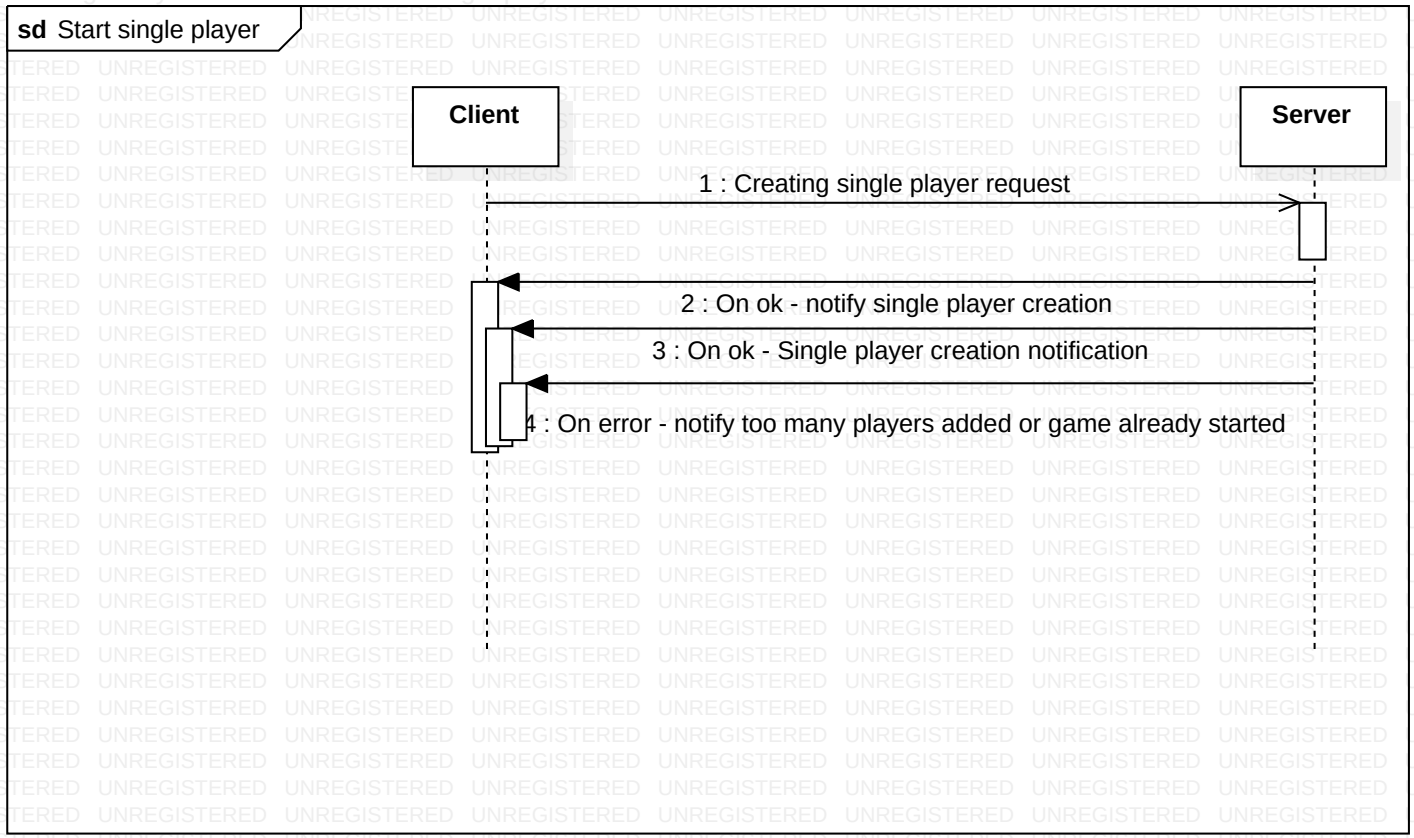
1: Server e Client scambiano messaggi fra loro ad intervalli regolari(ping e pong) attraverso due thread in background per verificare l'effettiva connessione tra Server e Client.

2: Il protocollo si basa sulla classe Message contenente un messageType, specifico per ogni sottoclasse che estende Message. Ogni sottoclasse di Message avrà infatti alcuni degli attributi specificati in "MESSAGE CONTENT", necessari per poter aggiornare i dati e di conseguenza visualizzare la nuova situazione tramite View oppure per invocare l'adeguato metodo sul Model.

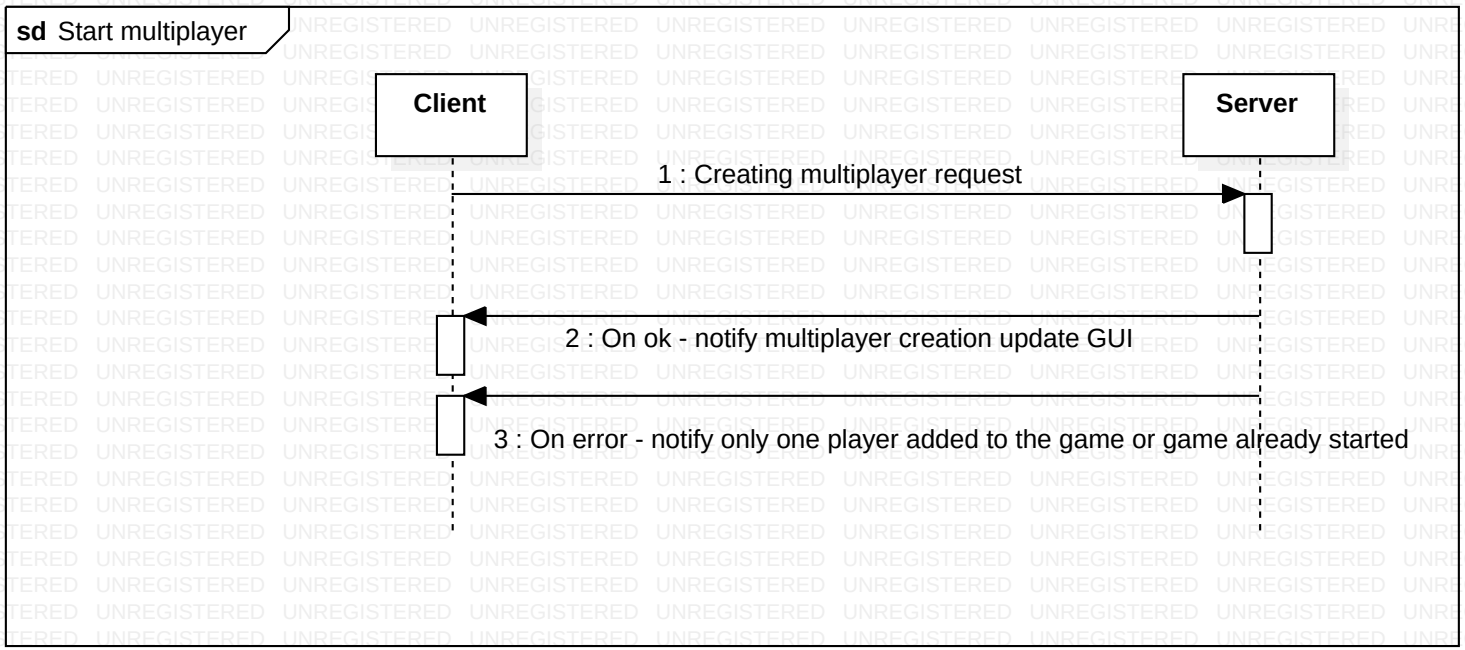
3: Abbiamo deciso di comunicare tra client e server attraverso messaggi di tipo JSON, quindi sia controller che view
avranno delle classi che deserializzeranno le stringhe JSON in un oggetto Message(nello specifico in un oggetto sottoclasse di Message, scelto in base all'attributo messageType). Allo stesso modo avranno un metodo
che serializzerà l'input dell'utente in stringhe JSON ( per esempio per tradurre sulla view i click dell'utente).



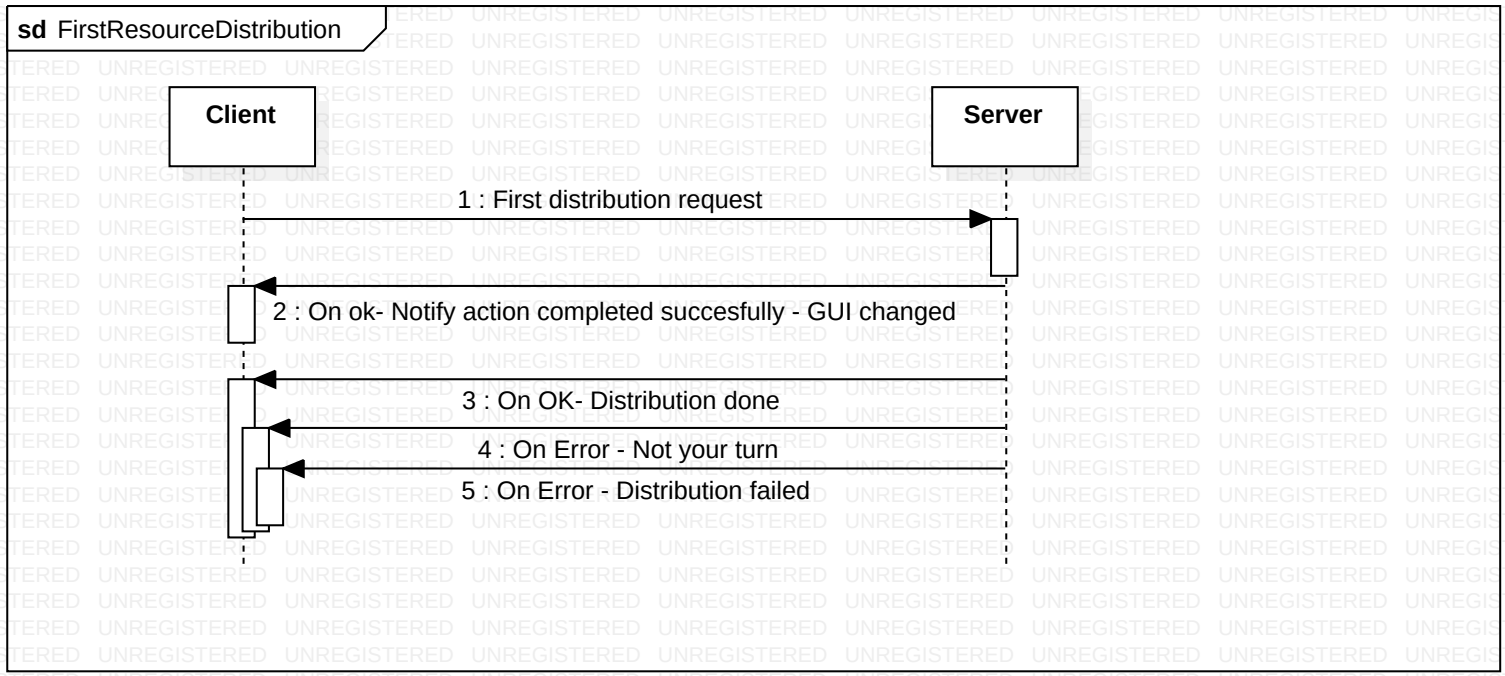

2: ["hostName", "portNumber"]
CONNECTION REPLY(SERVER)
3: {"messageType":"connectionAcceptedPleaseLoginNotification"}
4: {"messageType":"invalidLoginNotification"}
LOGIN REQUEST(CLIENT)
5: {"messageType":"AddPlayer", "senderNickname":"Player's nickname"}
LOGIN REPLY(SERVER)
6: {"messageType":"playeraAddedNotification"}
7: {"messageType":"reconnetedConfigurationMessage"}
8: {"messageType":"invalidPlayerAddedNotification"}
9:{"messageType":"AddPlayerNotificationForEveryone", "senderNickname":"Player's nickname"}



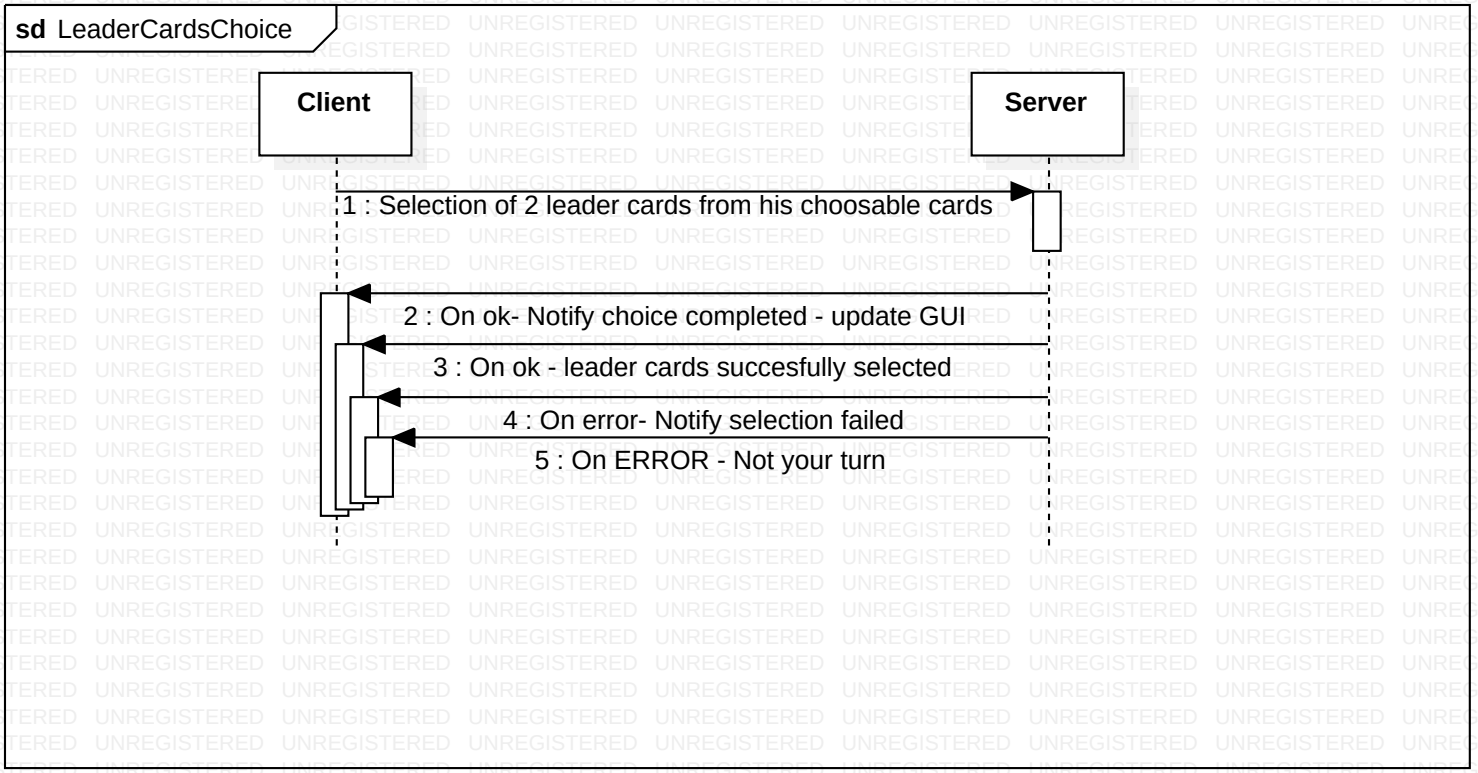

SINGLE PLAYER CREATION(CLIENT)
1: {"messageType" : "StartSinglePlayer"}
SINGLE PLAYER CREATION REPLY(SERVER)
2: {"messageType":"singlePlayerCreationMessage", int[4]: "choosableLeaderCardsNumbers", "Deckgrid": "deckgridConfiguration", String[][]: "marbleGridConfiguration", String: "marbleOut" }
3:{"messageType":"singlePlayerCreationOkNotification"}
4:{"messageType":"singlePlayerFailedCreationNotification"}




MULTI PLAYER CREATION(CLIENT)
1:{"messageType" : "StartMultiplayer"}
MULTI PLAYER CREATION REPLY(SERVER)
FOR EVERY PLAYER:
2: {"messageType" : "multiplayerCreationMessage", int:"playerNumber", String:"nickname", int[4]: "choosableLeaderCardsNumbers", Deckgrid: "deckgridConfiguration", String[][]:"marbleGridConfiguration", String:"marbleOut"}
3: {"messageType" : "multiplayerCreationErrorNotification"}

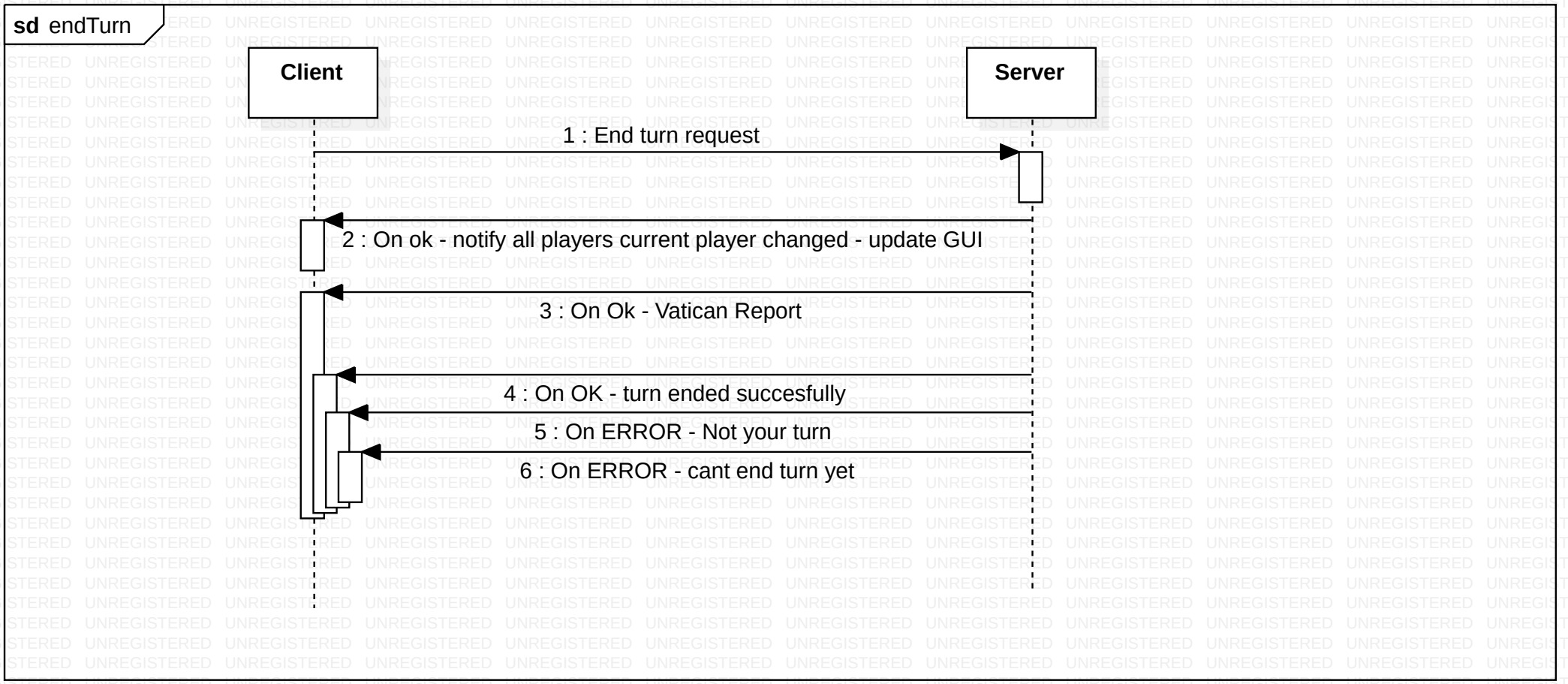



1: {"messageType":"distributionSecondThird", "senderNickname":"Player's nickname", "ResourceToDistribute":"ResourceType"}
**{"messageType":"distributionFourth", "senderNickname":"Player's
nickname","ResourceToDistribute":"ResourceType","SecondResourceToDistribute":"ResourceType"} //The fourth player has to choose 2 resources
2: SENT TO EVERYONE
2: {"messageType": "notifyWareHouseChanged", "warehouseConfiguration": Warehouse }
3: {"messageType": "distributionOkNotification"}
4:{"messageType": "notYourTurnNotification"}
5:{"messageType": "NotRightToDistributionNotification"}



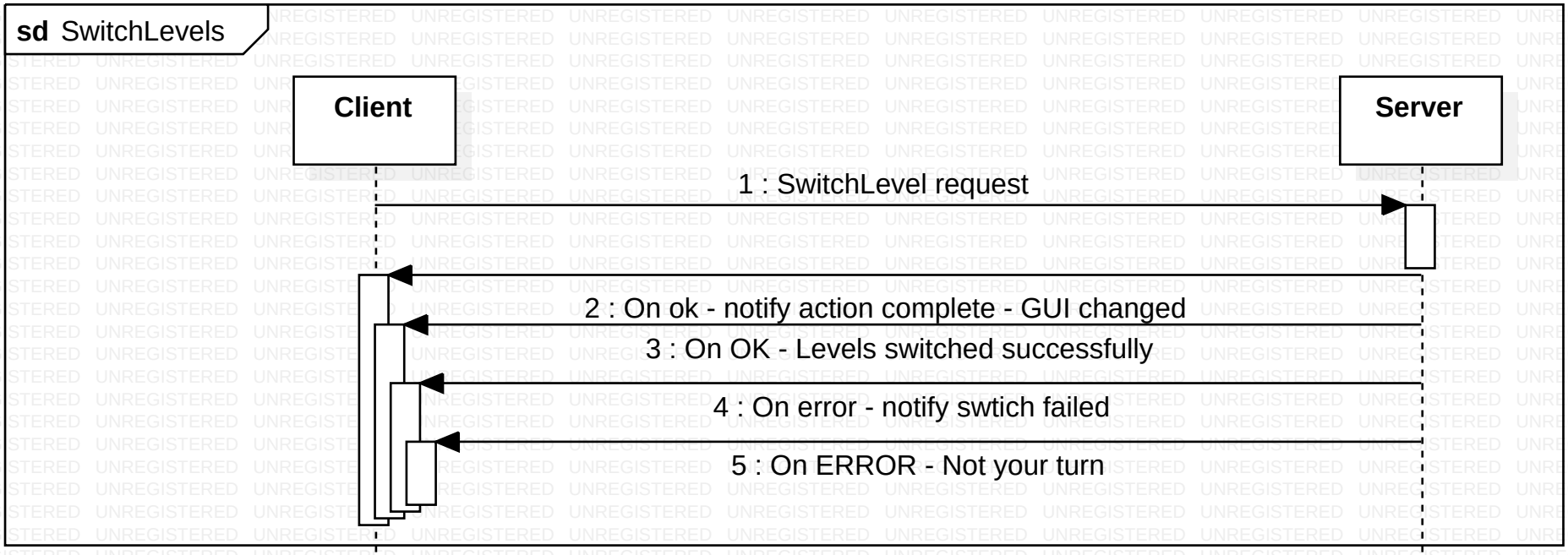

1: {"messageType":"leaderCardSelection",String: "senderNickname", int: leaderCardPosition1, int: leaderCardPosition2}
2: SENT TO EVERYONE
2:{"messageType":"chosenLeaderCardMessage", int : "firstChosenLeaderCardNumber", int : "seondChosenLeaderCardNumber"}
3:{"messageType":"leaderCardSelectionOkNotification"}
4:{"messageType":"notRightToLeaderCardSeletionNotification"}
5:"{"messageType": "notYourTurnNotification"}

endTurn::Interaction::endTurn

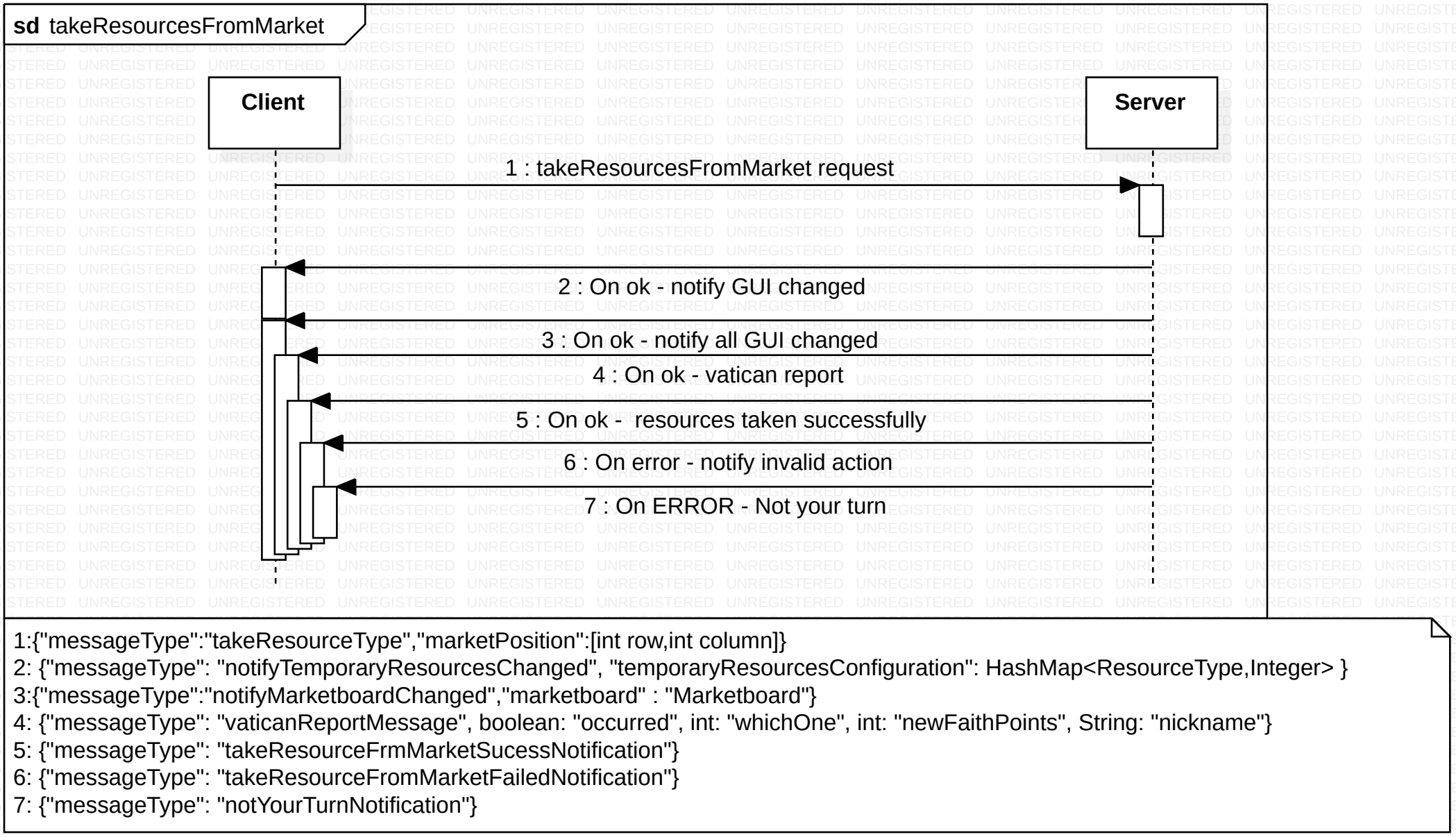


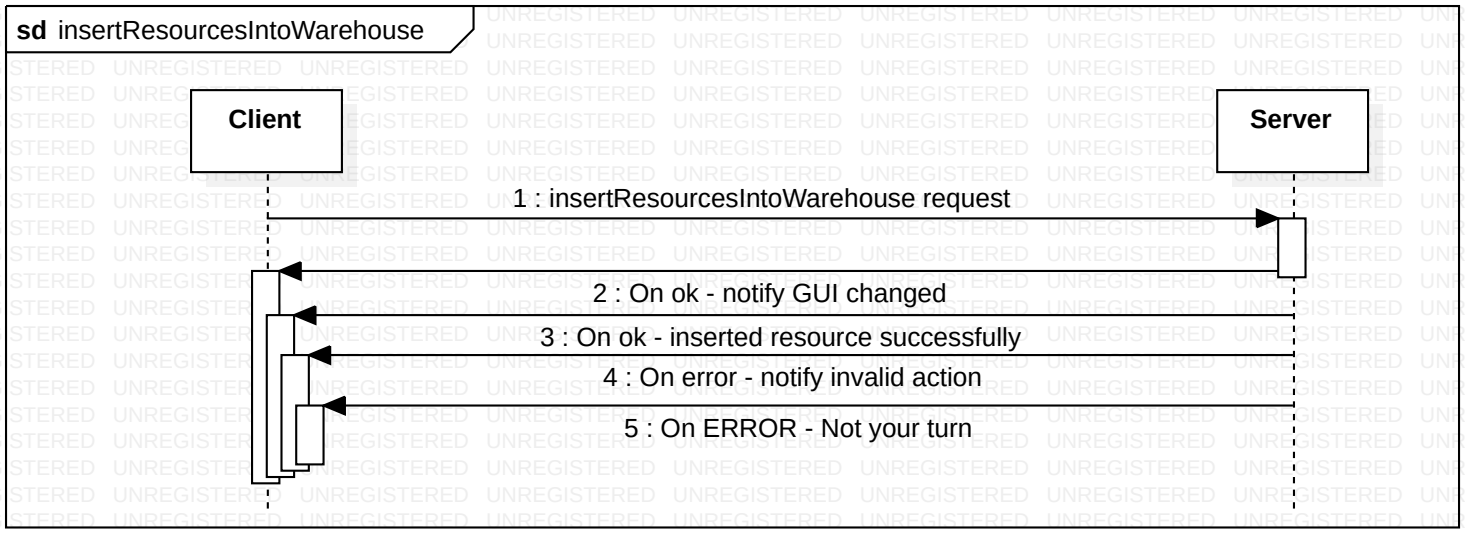

1: {"messageType": "EndTurnRequestMessage", String: "senderNickname"}
2: SENT TO EVERYONE
2: {"messageType": "endTurnNotificationMessage", String: "actualCurrentPlayer", int: "numResourcesDiscarded", String: "winnerPlayerNickname", int. "winnerPlayerNumber", int: "winnerPoints", boolean: "gameEnding", "Map":"temporaryResources", int: "blackFaithPoints"}
3: SENT TO EVERYONE(FOR EACH PLAYER DATA)
3: {"messageType": "vaticanReportMessage", boolean: "occurred", int: "whichOne", int: "newFaithPoints", String: "nickname"}
4: {"messageType": "endTurnOkNotification"}
5: {"messageType": "notYourTurnNotification"}
6: {"messageType": "notRightToEndTurnNotification"}

SwitchLevels.Interaction1.SwitchLevels

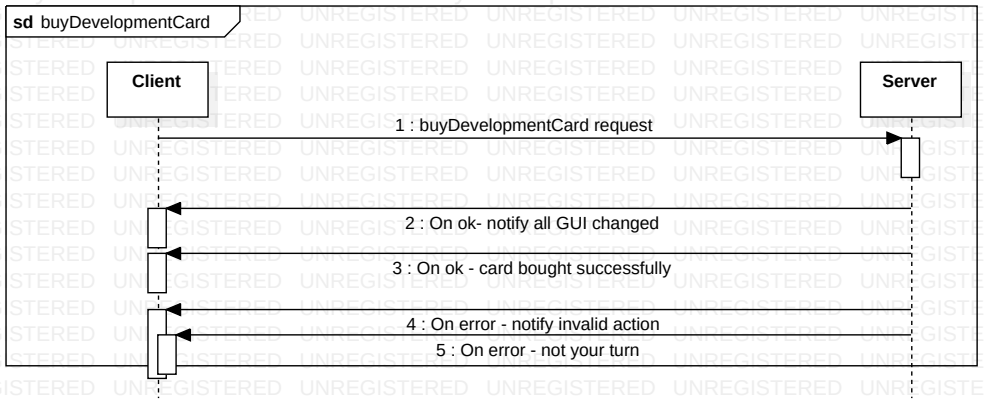



1:{"messageType":"SwitchLevel",String: "senderNickname",int[2]: "levelsToSwitch"}
2: SENT TO EVERYONE
2: {"messageType": "notifyWareHouseChanged", Warehouse: "warehouseConfiguration", String: "nickname" }
3: {"messageType": "SwitchLevelsSuccessNotification"}
4: {"messageType": "SwitchLevelsFailureNotification"}
5: {"messageType": "NotYourTurnNotification"}

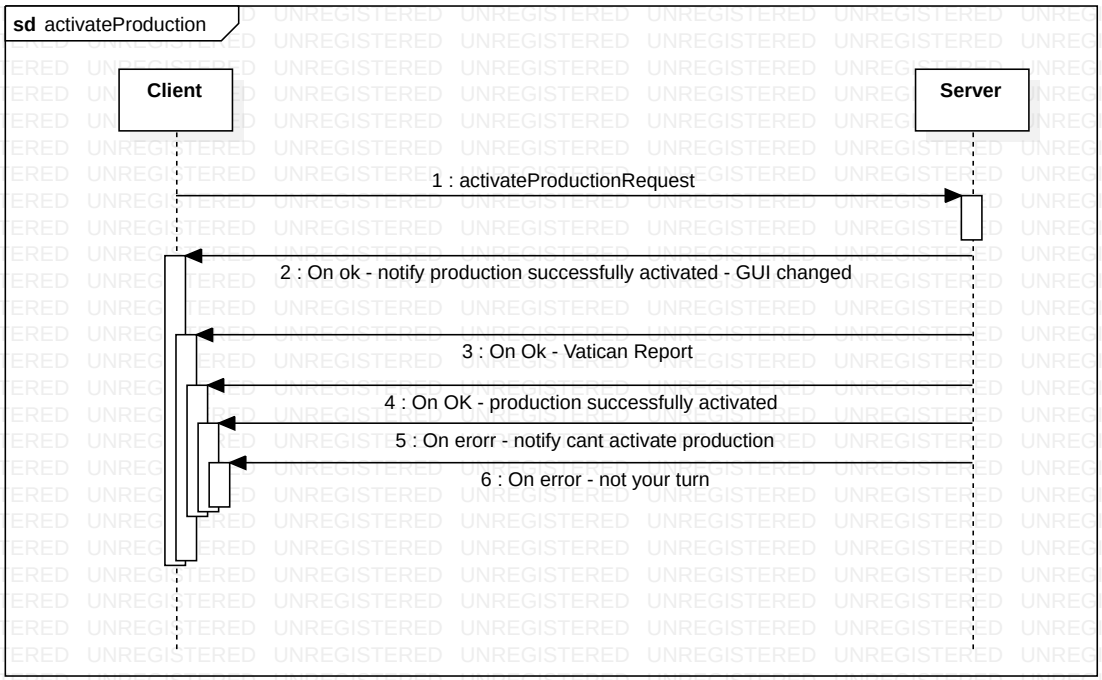

sd takeResourcesFromMarket
Client
Server
1 : takeResourcesFromMarket request
2 : On ok - notify GUI changed
3 : On ok - notify all GUI changed
4 : On ok - vatican report
5 : On ok - resources taken successfully
6 : On error - notify invalid action
7 : On ERROR - Not your turn
1:{"messageType":"takeResourceType","marketPosition":[int row,int column]}
2: {"messageType": "notifyTemporaryResourcesChanged", "temporaryResourcesConfiguration": HashMap<ResourceType,Integer> }
3:{"messageType":"notifyMarketboardChanged","marketboard" : "Marketboard"}
4: {"messageType": "vaticanReportMessage", boolean: "occurred", int: "whichOne", int: "newFaithPoints", String: "nickname"}
5: {"messageType": "takeResourceFrmMarketSucessNotification"}
6: {"messageType": "takeResourceFromMarketFailedNotification"}
7: {"messageType": "notYourTurnNotification"}




1:{"messageType":"insertResourceMessage","ResourceType": "resourceToInsert",int: "quantityToInsert",boolean: "intoExtraDeposit", String: "senderNickname"}
2:SENT TO EVERYONE
2: {"messageType": "insertedResourceChanged", Map<ResourceType,Integer>: "temporaryResourcesConfiguration" ,"Warehouse": "warehouseConfiguration", "ExtraDeposit[]": "extraDepositsConfiguration", Stirng: "senderNickname"}
3:{"messageType": "insertedResourcesSuccessNotification"}
4: {"messageType": "insertedResourceFilureNotification"}
5:{"messageType": "notYourTurnNotification"}

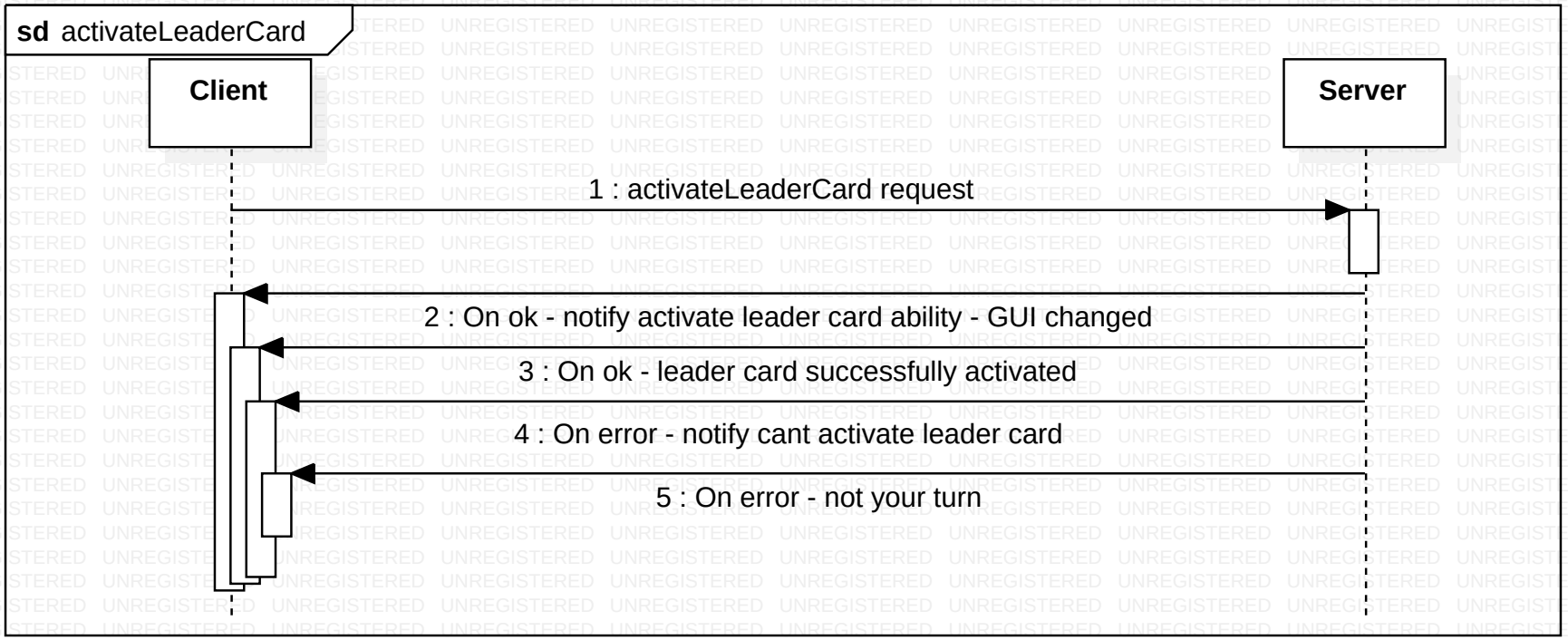



1:{"messageType":"buyDevelopmentCard",int: "level", "Colour": "colour", int: "slot", int[2]: "payUsingExtraDeposit", String: "senderNickname"}
2: SENT TO EVERYONE
2:{"messageType":developmentCardBought","developmentCard[3]": "developmentCardConfiguration", "Warehouse": "warehouseConfiguration", "ExtraDeposit[2]": "depositConfiguration",
"Map<ResourceType,Integer>": "strongboxConfiguration", "Deckgrid":"deckgridConfiguration", String: "senderNickname", ArrayList<DevelopmentCard>: "insertedDevelopmentCards"}
3:{"messageType": "buyDevelopmentcardSuccessNotfication"}
4:{"messageType": "buyDevelopmentFailureNotification"}
5:{"messageType": "notYourTurnNotification"}



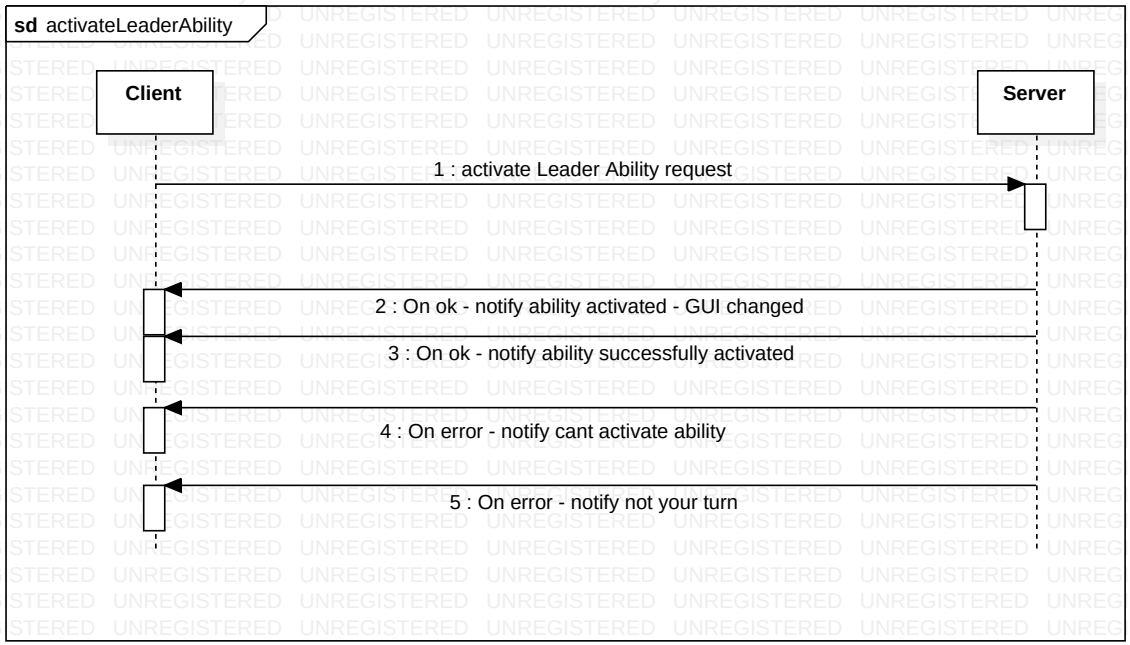

Il controller guarderà l'array di faith cards del player per vedere quando viene chiamata la prima, seconda o terza udienza salvando su un array di booleani quando una faith card viene posta a zero ( corrispondenza 1 a 1 posizioni faithCards-booleano)

1:{"messageType":"activateProduction", String: "senderNickname",boolean[3]: "whichDevCardSlot", boolean: "fromPersonalBoard", boolean[3]: "whichLeaderCard"
,"ResourceType[3]": "resourceBaseProduction", "ResourceType[2]": "resourceFromLeader", int[2]: "payUsingExtraDeposit"}
2: SENT TO EVERYONE
2:{"messageType":notifyActivateProductionMessage", "Warehouse": "warehouseConfiguration, "ExtraDeposit[2]": "extraDepositConfiguration","Map<ResourceType,Integer>": "strongboxConfiguration", int: "newFaithPoints",
String: "whoActivatedProduction"}
3: SENT TO EVERYONE(FOR EACH PLAYER DATA)
3: {"messageType": "vaticanReportMessage", boolean: "occurred", int: "whichOne", int: "newFaithPoints", String: "nickname"}
4:{"messageType": "activateProductionSuccessNotification"}
5:{"messageType": "activateProductionFailureNotification"}
6:"messageType": "NotYourTurnNotification"}



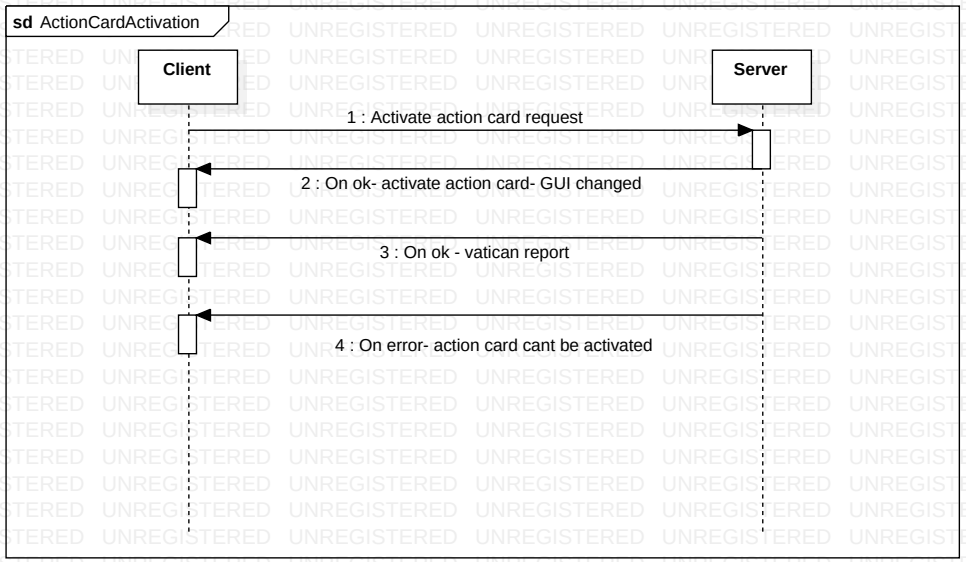

1:{"messageType":"activateLeaderCardMessage",String: "senderNickname", int: "position"}
2: SENT TO EVERYONE
2:{"messageType":notifyActivateLeaderCard", int: "activatedLeaderCardPosition", String: "whoActivatedLeaderCard"}
3:{"messageType": "activatedLeaderCardSuccessNotification"}
4:{"messageType": "activatedLeaderCardFailureNotification"}
5:{"messageType": "notYourTurnNotification", }

Activate Leader Ability interaction overview activateLeaderAbility




1:{"messageType":"activateLeaderAbilityMessage","senderNickname":"playerNickname","position","int"}
2:SENT TO ALL
2:
-(case deposit){"messageType":"activateLeaderAbilityDeposit","position","int", "extraDepositConfiguration":["extraDeposit1", "extraDeposit2"], nickname: String}
-(case discount){"messageType":"activateLeaderAbilityDiscount","position","int, nickname: String" }
-(case whiteTransformation){"messageType":activateLeaderAbilityWhiteTransformation","position","int", "temporaryResources": Map <ResourceType,Integer>,
remainingWhiteMarbles: int, nickname: String}
-(case production){"messageType":notifyActivateLeaderAbilityProduction","position","int", nickname: String}
3:{"messageType": "ActivateLeaderAbilitySuccessNotification"}
4:{"messageType": "ActivateLeaderAbilityFailureNotification"}
5:{"messageType": "NotYourTurnNotification"}

Action Card Activation Interaction ActionCardActivation




1:{"messageType":"actionCardActivation"}
2:
-(case move){"messageType":"moveLorenzo","newFaithPoints":"int"}
-(case moveAndShuffle){"messageType":"moveAndShuffle","newFaithPoints":"int", "actionCardConfig":ArrayList<String>}
-(case RemoveDevCard){"messageType":"notifyDeckgridChanged","deckgrid":"Deckgrid"}
3{"messageType": "vaticanReportMessage", boolean: "occurred", int: "whichOne", int: "newFaithPoints", String: "nickname"}
4:{"messageType": "ActionCardActivationFailureNotification"}