Network protocol

this file show how the network protocol works and the semantic of the packet sent from server to client and viceversa.

Server-sent packets

- **CONNEST**: this packet is sent by the server to communicate to the client that he has been succesfully registered into the server registry. A CONNEST message has this structure:

  ```
  "PLAYERID" : UUID_OF_THE_PLAYER
  ```

- **NAMECHG**: this packet his sent in broadcast to all clients connected to one specific game following the recived NAMECHG message sent by the client. It contains the name, choosen by the client, which will be show during the game. A NAMECHG message has this structure:

  ```
  "PLAYERID" : UUID_OF_THE_PLAYER
  "NAME" : CHOOSEN_NAME
  ```

- **STATCHNG**: this packet is a json-diff of the player object, i.e. contains all the changes referred to a variation of a player's status (the player's resources has been changed, the player has taken a new card,...). Because client-side initialization of Player's resources set all resources to zero, a first STATCHNG message is sent just after the server-side inizialization of player's resources (this allows a complete configuration of start player's resources simply setting them from the game configuration file, clients will adapt themself to server configuration). STATCHNG message are constructed simply wrapping the status of the client in a JSON packet. According to this, a STATCHNG message will contain the status of player's resources, the card owned by the player and the status of his familymembers (in this last case only a boolean busyFlag is transmitted):

  ```
  "PLAYERID" : UUID_OF_THE_PLAYER,
  "RESOURCE" : {
      "WOOD" : 1,
      "COINS" : 2,
      "MILITARY" : 1,
      ...
  }
  "FAMILYSTATUS" : [false,true,true,false,...],
  "BONUSTILE" : "string representing the personalBonusTile"
  ```

```
"PAYLOAD" : {
    "TERRITORYCARD" : [cardName1, cardName2, ...],
    "BUILDINGCARD" : [cardName1, cardName2, ...],
    ...
}
```

a STATCHNG message is always sent in broadcast.

- **CHGBOARDSTAT**: this message is used to synchronize all the clients on all the changes referred to the board status. For example when cards on board must be refreshed at the end of the period a BOARD-type CHGBOARDSTAT message is sent to communicate, to all client connected, the new card layout. Or when a familiy member has been placed a FAMILY-type CHGBOARDSTAT tells the clients where the familiy memeber has been moved.Changing of single cards (i.e. when a card has been takenf from the board) are communicated by sending a BOARD-type CHGBOARDSTAT. According to this a BOARD-type CHGBOARDSTAT will be structured in this way:

  ```
  "TYPE" : "BOARD",
  "PAYLOAD" : [{"CARD":CARD_NAME, "REGIONID":X, "SPACEID":Y},...]
  ```

  payload will be a 1:1 map of TowerRegions card layout (i.e. the filed Card on the client-side board of the actionSpace Y in the tower X will be set to CARD_NAME).

CHGBOARDSTAT message are also used to update the board after the movement of one pawn, in this case the payload will be structured in this way:

```
"TYPE" : "FAMILY",
"PAYLOAD" : {
                "REGIONID" : X,
                "SPACEID" : Y,
                "PLAYERID" : UUID_OF_THE_PLAYER,
                "FAMILYMEMEBER_ID" : Z
}
```

which tells that the family member number Z of the player indicated has been moved to the action space Y of the region X.

finally CHGBOARDSTAT are used at the end of each round to flush the client board:

```
"TYPE" : "FLUSHFAMILY",
"PAYLOAD" : {
          "TURNENDFLAG" : true
}
```

- **GMSTRT**: this is the first message sent by the server when the Game thread starts. It is aimed to synchronize the clien-side model of the game, i.e. the board configuration and the number of players connected to the specific game, with the server-side model. In this way every configuration setted on the server is automatically adapted on every client. For example if the board must have five towers, a simply change on the server board will change the client-side board, or if action spaces' bonus are customize on server-side board they will be setted in the same way on the client-side board.Every action space will have a structure like this:

```
{
"BONUS" : {
      "COINS" : 1,
      "WOOD" : 1,
      ...
},
"REGIONID" : X,
"SPACEID" : Y,
"ACTIONVALUE" : Z,
"SINGLE" : TRUE/FALSE
}
```

  If not specify, ACTIONVALUE is setted to 1 by default and SINGLE is setted true by default. The special carachter # referred to BONUS indicate that the action space has no bonus. REGIONID, SPACEID and BONUS can't be null. So a GMSTRT packet will have a format like this:

```
"BOARD" : {
  "PRODUCTIONREGION" :  [{ActionSpace},...],
  "HARVESREGION" : [{ActionSpace},...],
  "COUNCILREGION" : [{ActionSpace}],
  "MARKETREGION" : [{ActionSpace},...],
  "TOWERREGION" : [{ActionSpace},...],
  ....
  "TOWERREGION" : [{ActionSpace},...]
},
"PLAYERLIST" : [UUID_OF_THE_PLAYER1, UUID_OF_THE_PLAYER2 ,...]
```

  GMSTRT is sent in broadcast and is automatically generated by the code by simply looking to the server-board configuration and player list status. With GMSTRT packet also a very important information is transmitted, the game UUID, used for all the future packet generated by the client to allow the server to process different packet in a multi-game session.

- **CONTEXT**: this message is the main way to force the client to change his context. It is used when more informations are required to finish the action

the player has start. For example when a player take a PRIVILEGE-instant-effect type card the server must ask the client what resources he want to gain by consuming his privilege. Or if the client perform a PRODUCTION-type action, the server must know what CHANGE-permanent-effect type card must be activated. Those informations can't be known by the server when an ASKACT is sent and so a CONTEXT message is sent. On the client-side this message will be processed and the right context will be open. Every CONTEXT message contains the CONTEXT identifier which will be used by client to open the correct context:

```
"CONTEXTID" : X,
"PAYLOAD" : {
    ...
}
```

are 4 the types of possibile contexts that can be open, every context is characterized by a well-defined payload format:

- *PRIVILEGE*: used when a choise on a council privilege must be performed:

```
"NUMBER" : X,
"COST" : {
  "COINS" : 1,
  ...
}
```

the filed "NUMBER" indicates the number of privilege which can be utilized. A "COST" field is used if the privilege must be payed before it can be spent (this field is used only by cards like card 32, Residence).

- *SERVANT*: used when the server must ask to the client if he want spend any servant to increase the value of the PRODUCTION or HARVEST action he want perform.

```
"NUMBER_SERVANTS" : X,
"ACTIONTYPE" : actionType
```

"NUMBER_SERVANTS" field is used to inform the client on the number of servant which can be spent. The "ACTIONTYPE" field is used only for display what type of action (PRODUCTION/HARVEST) has triggered the context.

- *EXCOMMUNICATION*: used only at the end of each period to ask the client if he has been excommunicated or whether he want or not show his faith to the Church.

```
"PLAYER_FAITH" : X,
"FAITH_NEEDED" : Y
"PLAYERID" : UUID_OF_THE_PLAYER
```

- *CHANGE*: used to open a CHANGE-context when a card with a CHANGE permanent effect has been activated. When a PRODUCTION action has been triggered, the server put all the information needed by the CHANGE context to display the possible choises into a CHANGE-CONTEXT packet.

```
"NAME" : [CardName1, cardName2, ...],
"RESOURCE" : [
  [
      {
          "RESOURCEIN": {
              "STONE": 1
          },
          "RESOURCEOUT": {
              "VICTORY_POINTS": 3
          }
      },
      {
          "RESOURCEIN": {
              "STONE": 3
          },
          "RESOURCEOUT": {
              "VICTORY_POINTS": 7
          }
      }
  ],
  {
      "RESOURCEIN": {
            "COINS" : 1
      }
      "RESOURCEOUT": {
            "MILITARY_POINTS" : 2
      }
  },
  ....
]
```

between the two arrays must there be a 1:1 mapping, i.e. the first element of the RESOURCE array must be the payload of the CHANGE effect of the first CarnName which appear in the NAME array. Note that when a CHANGE payload has been formatted as Array this means that the effect is characterized by more than one choise and that the choises are

exclusive. Instead when a CHANGE payload has been formatted as Object this means that the effect has only one possibly choise.

- *LEADERSET* : only used during the leader card distribuition phase at the start of the game, it is sent to allow the clients to chose the leader card they want take.

  ```
  "LIST" : ["LEADERCARD1" , "LEADERCARD2" , ...]
  ```

- *ACTION* : open an action context when a bonus action must be performed.

  ```
  "PAYLOAD" : {
          "BONUSACTIONVALUE": 2,
          "REGIONID": 7,
          "EXCLUSIVEBONUS": null,
          "TYPE": "TOWER",
          "BONUSRESOURCE": null,
          "FLAGREGION": null
      }
  ```

  the payload contains all the information needed to perform the specified action.

ASKLDRACK: this message is sent as reply after the reception of a ASKLDRACT message and the following appllication of the leader action. The message has a payload format structured in this way:

```
{
  "LEADERCARD" : selected LEADERCARD,
  "DECISION" : type of leader action performed
}
```

according to game rules, the DECISION field can assume one of this value: "DISCARD" if the leader card has been discarded, "PLAY" if the card has been played and "ACTIVATE" if the effect of the card has been activated.

ACTCHK : this message is sent as reply to an ASKACT message and bring whit itself the result of the action already applied. The result is contained into a "RESULT" field, if it is set as true this means tha action is valid and the server already applied it, otherwise the action was not valid and the client must perform another action.

DICEROLL : this message updates all the client on the dice value after the dice roll phase. The message payload is structured in this way:

```
"BLACKDICE" : X,
"WHITEDICE" : Y,
"ORANGEDICE" : z
```

TRNBGN : this message is sent to all the clients connected to inform them on the identity of the client who as the game lock. Its payload will simply contain the UUID of the player to which the lock has been passed:

```
"PLAYERID" : UUID_OF_THE_PLAYER_WHO_HAS_THE_LOCK
```

ENDGAME : message sent to tells the client the game is end.

Client-sent packets

- **CHGNAME** : this packet contains the name chosed by the client which will be show during the game

  ```
  "PLAYERID" : UUID_OF_THE_PLAYER
  "NAME" : CHOOSEN_NAME
  ```

- **MSG** : this packet is used to send chat messages. The message is structured in this way:

  ```
  "FLAG" : allFlag,
  "RECEIVER" : UUID_OF_THE_RECEIVER,
  "SENDER" : UUID_OF_THE_SENDER,
  "MESSAGE" : message
  ```

  the value contained into the "FLAG" field is used to tells the server if the message must be sent as broadcast or if it is a private message, in this last case the "RECEIVER" field is used to redirect the message to the only interested player.

- **ASKACT** : this message is used by the client to perform an action. It contains all the information needed to check if the move is valid (after this phase contexts can be open). The payload will be structured as follow:

  ```
  "ACTIONTYPE" : PRODUCTION,
  "FAMILYMEMEBER_ID" : 1,
  "REGIONID" : 0,
  "SPACEID" : 0,
  "COSTINDEX" : 0,
  "CARDNAME" : null
  ```

this packet tells that the client want perform an action of type PRODUCTION whit his family member number 1 in the specified action space of the board. Field CARDNAME and COSTINDEX are considered only if CARDNAME is not null. In this case, if the action space is free, the action has a type TOWER and the field CARDNAME can't be null. COSTINDEX will contains the information of the chosen cost (if the card has only one cost 0 is setted by default).

- **ASKLDRACK** : this message is used to perform a leader action. Its payload is in this way structured:

  ```
  "LEADERCARD" : nameOfTheLeaderCard,
  "DECISION" : type of leader action to perform
  ```

  according to game rules, the DECISION field can assume one of this value: "DISCARD" if the leader card has been discarded, "PLAY" if the card has been played and "ACTIVATE" if the effect of the card has been activated.

- **LDRSET** : this message is used only in the leader card distribuitioln phase and allows all the client connected to the game to chose their leader card. LDRSET contains an array of LEADERCARD, each player chose one leader card and then send a LDRSET message to the server which will send the received message (as LEADERSET CONTEXT message) to the next player. When the array is empty the leader card distribuition phase is end. Payload of this message:

  ```
  "LIST" : ["LEADERCARD1","LEADERCARD2", ...],
  "PLAYERID" : UUID_OF_THE_PLAYER_WHO_HAS_TO_CHOISE
  ```

- **TRNEND** : this message is sent when the client has terminated is turn. When the server will receive a TRNEND message it will pass the lock to the next player.

- **SENDPOPE** : this message is used only during the excommunication phase and has the purpose to tell the server what decision the client has done:

  ```
  "ANSWER" : TRUE/FALSE
  "FAITH_NEEDED" : FAITH_POINTS_NEEDED,
  "PLAYERID", UUID_OF_THE_PLAYER
  ```

  the field ANSWER if setted true tells the server that the player hasn't show his faith to the pope and so he will be excommunicated, if setted false he won't be excommunicated.

Connection phase packet flow

when the client connects to the server throught **MsgConnection** interface it recive a **CONNEST** packet (sent by the network threads which manage the client connection). The client will response with a **CHGNAME** message to communicate his name.

When the game start a **GMSTRT** message is sent in broadcast, after that a CHGBOARDSTAT message is sent in broadcast as well to communicate the start card layout. following this, a **DICEROLL** message is sent, then the leader card distribuition phase is handled. When this phase is finished a **STATCHNG** message is sent to initilize the Client status. Finally the lock is given to the first player and the first **TRNBGN** is sent.

action phase packet flow

When **TRNBGN** is received on the client side, client perform his action. When the action is ready to be sent an **ASKACT** message is sent. The server receiving this message will perform all the check on the action. During the application of the action (so only if the action is valid), depending on the action performed by the client, some context may be open. In this case **CONTEXT** message are sent to the client. The client will open the correct context and will reply with a **CONTEXTRESPONSE** message (a message containing the response at the context). The server taken all the information needed to apply the action will finally change the game status. The server will send a final **ACTCHK** message to the client to tell the action was valid and was already applied. At this point client will respond with a **TRNEND** message, and the server can pass the lock and send a new **TRNBGN** message to the next player.