TFA protocol spec

TFA (The Force Awakens) protocol key concepts:

- It a client-server protocol
- It runs over a TCP sessions
- It exchanges message between the client and the server
- All messages have to be ACKed within 3 seconds, or the connection is reset
    - Additional layer of ACKs is required, since TCP have long timeouts
- JSON is used to encode messages
- Messages can not contain new lines and/or carriage returns
- New lines are used for message delimitation

Game workflow

Every match follows the same workflow:

- Auth phase in which players register on the server
- Wait phase in which players wait for the match to start
- Play phase in which the match takes place

The workflow is described from player's points of view.

Auth

The communication between player and server always start by authentication.
The player must send the following JSON object:

```
{
        "cmd": "auth",
        "player": "<playername>"
}
```

The server replies with the following JSON object if auth succeeds:

```
{
        "cmd": "auth-ok"
}
```

With the following if the server is already busy with another match:

```
{
        "cmd": "auth-busy"
}
```

If the player name is already taken by another **active** player (i.e. active TCP session), the server replies with the following:

```
{
        "cmd": "auth-taken"
}
```

If auth phase is successful, the protocol proceeds into Wait phase.
Wait phase is skipped if the game is already going on and the player with this name was already playing but disconnected. In such case server sends world updates right away.

Wait

In the wait phase the players browse the list of current players and their teams.
The players may change the team during this phase.

Server messages

Roster updated

Server say roster update notifications, when a new player enters, exists or changes allegiance.

```
{
	"cmd": "roster-update",
	"players": [
			{
				"name": "Foo Bar Baz",
				"team": "Jedi"
			},
			{
				"name": "Bar Baz Foo",
				"team": "Jedi"
			},
			{
				"name": "Baz Foo Bar",
				"team": "Sith"
			}
	]
}
```

The first notification is sent when the player first goes into Wait state.
Other notifications are sent each time the roster changes.

Game start

Currently we support only 2 vs 2 players matches. The game starts as soon as 2 players for each team are reached. If the player receives a notification with 2 balanced teams in the roster, the game is to be considered started and the protocol goes to Play state.

Player messages

Allegiance change

A player may chose to change his team before the game has started.
This is done with the following message:

```
{
        "cmd": "change-team"
}
```
The server will then notify the change to all the players.

Play

When the player enters a Play phase, the server starts sending world change notifications. World change notifications has the following format:

```
{
        "cmd": "world_change",
        "checkpoints": [
                {
                        "name": "checkpoint_name",
                        "type": "Checkpoint|Base",
                        "team": "Jedi|Sith",
                        "hp": 100
                }
        ],
        "players": [
                {
                        "name": "player name",
                        "team": "Jedi|Sith",
                        "position": {
                                "latitude": 90.0,
                                "longitude": 90.0,
                        },
                        "state": "Roaming|Capturing|Dead"
                        "capturing": "checkpoint_name|empty string"
                }
        ]
}
```
World change notifications are send every second and shall be ACKed as all other messages.

Play phase contains different subphases:

- Roming
- Capturing
- Dead

When the player arrives to a checkpoint, he may request to start capturing the checkpoint, by sending:

```
{
        "cmd": "capture",
        "checkpoint": "<checkpoint_name>"
}
```

The server replies with:

```
{
        "cmd": "capture-started"
}
```

If the capture was started successfully. Or otherwise with:

```
{
        "cmd": "capture-refused"
}
```

If the capture was successfully started, the player passes to Capturing state.
The user will go back to Roaming state either if:

- The checkpoint was captured (notified via world state updates)
- The user gone away from the access point (the presence is verified by AP name and pings every second)

If the user goes away from the server, the server will attempt to send the following message to the user:

```
{
        "cmd": "capture-stop"
}
```

Position reports

Every time a world change report arrive, the player must send a position report, formatted as follows:

```
{
        "cmd": "position-report",
        "position": {
                                "latitude": 90.0,
                                "longitude": 90.0,
        }
}
```

Server must ACK the position report.

Basic combat

When checkpoint is contested between even number of players, the server randomly kills one player (“by the fair roll of a dice”).
The server notifies to the killed player:

```
{
        "cmd": "killed"
}
```

The killed player has to run to the base and “capture” it.
Then player passes back to Roaming state.

Game over

The game is over when the one team captures the “base” checkpoint of another team.
This will be notified by world notification and the client’s TCP session will be closed.

Disconnect

When someone wants to disconnect from server he must send message:

```
{
        "cmd": "goodbye"
}
```

The server replies with:

```
{
        "cmd": "goodbye-ok"
}
```

And connection is closing.