

Standard-FantasticFeasts-REVIEWME-12

Standardisierungsdokument zum Softwaregrundprojekt 18/19

sopra teams

24. April 2019

Contents

1	Changelog	4
1.1	Changelog REVIEWME-11 -> REVIEWME-12	4
1.2	Changelog REVIEWME-10 -> REVIEWME-11	5
1.3	Changelog REVIEWME-9 -> REVIEWME-10	5
1.4	Changelog REVIEWME-8 -> REVIEWME-9	5
1.5	Changelog REVIEWME-7 -> REVIEWME-8	5
1.6	Changelog REVIEWME-4 -> REVIEWME-7	6
2	About this document	6
2.1	Whole Game Sequence in short	6
2.1.1	Definition Sequence	7
3	Kommandozeilenparameter für KI und Server	7
3.1	Type Declaration	7
3.1.1	(int/difficulty)	7
3.1.2	(int/verbosity)	8
3.2	Definition	8
3.2.1	KI	8
3.2.2	Server	8
4	Container	9
4.1	Description	9
4.2	Type Declaration	9
4.2.1	(string/payloadType)	9
4.3	Definition	9
4.3.1	container	9
4.3.2	Example	9
4.3.3	Overview	10
4.4	Debug	10
4.4.1	Description	10
4.4.2	Definition	10
4.4.3	Example	11
4.5	Login	11
4.5.1	Type Declaration	12
4.5.2	Definition	12
4.5.3	Example	12

5	Konfigurationen für Team und Partie	13
5.1	Description	13
5.2	Type Declaration	13
5.2.1	(string/name)	13
5.2.2	(string/role)	13
5.2.3	(string/sex)	13
5.2.4	(string/broom)	14
5.2.5	(string/RRGGBB)	14
5.2.6	(float/prob)	14
5.2.7	(string/base64/png)	14
5.3	Definition	14
5.3.1	matchConfig (file)	14
5.3.2	teamConfig (file)	15
6	Match-Start-Finish	16
6.1	Description	16
6.2	Type Declaration	16
6.2.1	(string/victoryReasonType)	16
6.3	Definition	17
6.3.1	matchStart	17
6.3.2	matchFinish	17
7	TeamFormation	17
7.1	Description	17
7.2	Definition	17
7.2.1	teamFormation	17
8	Snapshot	18
8.1	Description	18
8.2	Type Declaration	18
8.2.1	(string/fanType)	18
8.2.2	(int/posx)	18
8.2.3	(int/posy)	18
8.2.4	(string/phaseType)	19
8.3	Definition	19
8.3.1	teamSnapshot	19
8.3.2	snapshot	20
9	Next	21
9.1	Description	21
9.2	Type Declaration	21
9.2.1	(string/entityID)	21
9.2.2	(string/turnType)	22
9.3	Definition	22
9.3.1	next	22
9.4	Example	22
10	Delta	22
10.1	Description	22
10.2	Type Declaration	23
10.2.1	(int/posx)	23
10.2.2	(int/posy)	23

10.2.3	(string/deltaType)	23
10.2.4	(string/entityID)	23
10.3	General Definition	24
10.3.1	deltaRequest, deltaBroadcast	24
10.4	DeltaType-Specific Definition	25
10.4.1	snitchCatch (deltaBroadcast)	25
10.4.2	bludgerBeating (deltaRequest)	25
10.4.3	bludgerBeating (deltaBroadcast)	25
10.4.4	quaffleThrow (deltaRequest)	25
10.4.5	quaffleThrow (deltaBroadcast)	26
10.4.6	snitchSnatch (deltaRequest)	26
10.4.7	snitchSnatch (deltaBroadcast)	26
10.4.8	trollRoar (deltaRequest)	26
10.4.9	trollRoar (deltaBroadcast)	26
10.4.10	elfTeleportation (deltaRequest)	27
10.4.11	elfTeleportation (deltaBroadcast)	27
10.4.12	goblinShock (deltaRequest)	27
10.4.13	goblinShock (deltaBroadcast)	27
10.4.14	ban (deltaBroadcast)	27
10.4.15	bludgerKnockout (deltaBroadcast)	28
10.4.16	move (deltaRequest)	28
10.4.17	move (deltaBroadcast)	28
10.4.18	wrestQuaffle (deltaRequest)	28
10.4.19	wrestQuaffle (deltaBroadcast)	29
10.4.20	foolAway (deltaBroadcast)	29
10.4.21	phaseChange (deltaBroadcast)	29
10.4.22	goalPointsChange (deltaBroadcast)	29
10.4.23	roundChange (deltaBroadcast)	29
10.4.24	skip (deltaRequest,deltaBroadcast)	30
10.4.25	unban (deltaRequest)	30
10.5	Example	30
10.5.1	Example quaffleThrow	30
10.5.2	Example ban	31
11	Pause	32
11.1	Description	32
11.2	Definition	32
11.2.1	pauseRequest	32
11.2.2	continueRequest	32
11.2.3	pauseResponse	32
12	Reconnect	33
12.1	Description	33
12.2	Definition	33
12.2.1	reconnect	33
13	Replay	33
13.1	Description	33
13.2	Type Declaration	34
13.2.1	(string/replayPayloadType) (subset of payloadType)	34
13.3	Definition	34
13.3.1	getReplay	34

13.3.2	replay (file)	34
14	standardisierte Mods (modifications)	34
14.1	Description	34
14.1.1	Overview	34
14.2	Chat-mod	35
14.2.1	Description	35
14.2.2	Definition	35
14.3	DisableGenderBalance-mod	35
14.3.1	Description	35
14.4	ReplayWithSnapshot-mod	35
14.4.1	Description	35
14.4.2	Type Declaration	36
14.4.3	Definition	36
14.5	Error-mod	36
14.5.1	Description	36
14.5.2	Definition	37
14.5.3	privateError	37
14.6	Warning-mod	37
14.6.1	Description	37
14.6.2	Definition	37
14.6.3	privateWarning	37
14.6.4	Error and Warning List	37
14.6.5	Lobby Mod	37
14.7	Test	38
14.7.1	testFormat	38
14.7.2	testRequest	38
14.7.3	testResponse	38
15	appendix	38

1 Changelog

Um die genauen Änderung zu verfolgen bitte `git diff` verwenden. Changelog könnte unvollständig sein.

1.1 Changelog REVIEWME-11 -> REVIEWME-12

- [x] neue Zeicheneinschränkung (`string/name`) in `teamConfig` für Teamname und Namen.
- [x] neuer `deltaType` `foolAway`
- [x] deltas genauer zwischen `deltaRequest` und `deltaBroadcast` unterschieden
- [x] die `payloadType` `deltaUnicast` und `nextUnicast` entfernt
- [x] `reconnect` neues Feld `next`
- [x] neue `deltaTypes` `phaseChange`, `goalPointsChange`, `roundChange`, `skip`, `unban`
- [x] neue felder in deltas
- [x] container im `reconnect`
- [x] Es werden keine `deltaBroadcast` mehr verschickt. Deltas befinden sich jetzt innerhalb eines `snapshots`.

1.2 Changelog REVIEWME-10 -> REVIEWME-11

- [x] Im Beispiel "success": "false" -> verbessert -> "success": false
- [x] Im Beispiel timeout": "7500" -> verbessert -> "timeout": 7500
- [x] password gehört zum userName (`joinRequest`)
- [x] Reihenfolge in der Json egal angemerkt.
- [x] Eine Partie gleichzeitig pro lobby. Server muss mindestens eine lobby unterstützen.
- [x] Der Server kickt den Client bei falsch eintreffenden `deltaRequest`
- [x] Optionen fuer das Lobbyfeld genauer definiert (`string/lobbyType`)
- [x] Das Replay ist erst nach dem MatchFinish anforderbar
- [x] In einer Lobby können hintereinander mehre Spiele gespielt werden.
- [x] success Feld in deltas genauer beschrieben
- [x] `knockout` feld im Snapshot
- [x] default lobby ist `hogwarts`
- [x] Ein Beispiel zum ban und neuer turnType `removeBan`
- [x] Error and Warning List
- [x] container Example

1.3 Changelog REVIEWME-9 -> REVIEWME-10

- [x] Der Login stellt Identitätsabsicherung
- [x] `payloadType` in Definition vergessen.
- [x] Alle Json Nachrichten werden per Websocket Protokoll verschickt. (<https://tools.ietf.org/html/rfc6455>)
- [x] Im `replay` timestamp statt date und `startTimestamp` Type
- [x] Der neuste Login zählt.
- [x] `userName` für den gesamten Server
- [x] alphanumerisch -> unbenannt -> `userNameType`
- [x] `userNameType` Zeichenlänge definiert
- [x] username und password werden in der KI hardgecoded
- [x] `null` falls der Schnatz noch nicht auf dem Spielfeld ist

1.4 Changelog REVIEWME-8 -> REVIEWME-9

- [x] "matchConfig" und "teamConfig" als File auch im Container Format präzisiert.
- [x] in Next (`int/millisec`)
- [x] Die Farben `primary` und `secondary` müssen unterschiedlich sein.
- [x] `teamFormation` startpositionen an pflichtenheft angepasst (appendix).

1.5 Changelog REVIEWME-7 -> REVIEWME-8

- [x] `elfs` -> unbenannt -> `elves`
- [x] Changelog in Dokument übernommen
- [x] In deltas müssen `null`-Felder mitgeschickt werden
- [x] neuer `deltaType` `wrestQuaffle`
- [x] Fehler in Type Declaration z.B. (`int/ballPhase`)-> unbenannt -> `ballPhase`
- [x] delta json formatierung angepasst
- [x] (`string/fan`) gelöscht da es definiert war aber nicht benutzt wurde

1.6 Changelog REVIEWME-4 -> REVIEWME-7

- [x] (int/Difficulty) -> unbenannt -> (int/difficulty)
- [x] (int/Verbosity) -> unbenannt -> (int/verbosity)
- [x] (int/actionPhase) löschen
- [x] turnType fan hinzufügen
- [x] Beschreibung das man timeout in **next** braucht wegen Pause
- [x] Ersichtlich beschreiben das reconnect für Zuschauer und Spieler gleich abläuft
- [x] Im Login, Server muss die Verbindung schließen wenn er gewünschte mods nicht unterstützt
- [x] kurz Spielablauf beschrieben am Anfang des Dokumentes
- [x] remove probabilities goal aus matchConfig
- [x] propability -> unbenannt -> probability
- [x] thinderblast -> unbenannt -> tinderblast
- [x] passiveEntity in Beispiel und Beschreibung unterschiedlich ausgebessert
- [x] neue payloadType deltaUnicast und nextUnicast
- [x] Server muss beim Reconnect deltaUnicast und nextUnicast payloads schicken.
- [x] quaffleThrow Definition präzisiert.
- [x] teamFormation startpositionen präzisiert (appendix).

2 About this document

Dieses Dokument soll ein strukturiertes Nachschlagewerk sein um beim Implementieren der jeweiligen Komponenten einen roten Faden zu haben. Alle Json Nachrichten werden per Websocket Protokoll verschickt. (<https://tools.ietf.org/html/rfc6455>)

Das Protokoll ist delta und snapshot basiert. Bei Protokollverletzung wird der Client gekickt. Es ist egal in welcher Reihenfolge die einzelnen Felder in der Json stehen.

2.1 Whole Game Sequence in short

Erst einmal interessiert einen der Verlauf des Spiels. Hier ist in sehr kurzer Weise dargestellt wie ein Spiel abläuft. Einigen wird schon auffallen das es sehr viele Fälle gibt die noch nicht abgedeckt sind. Deswegen sind genauere Beschreibungen in den betreffenden Abschnitten behandelt. Auch sind hier die Formate welche für die Züge der Spieler verantwortlich sind nicht beschrieben. Es ist mehr ein Ablauf welcher von einem Zuschauer gesehen wird, wobei auch dieser mehr vom Spiel mitbekommen wird.

- **joinRequest**: Ist in jedem Fall die erste Nachricht, welche ein Client an den Server sendet. Bei einem erfolgreichen Login befindet man sich in einer Lobby und ist vorerst ein Zuschauer.
- **teamConfig**: Ein Zuschauer wird zum Spieler, indem er seine Team-Konfiguration an den Server sendet und diese erfolgreich angenommen wird.
- **matchStart**: Das Spiel startet, sobald es zwei Spieler gibt. Teilt beiden Spielern die Spielfeldseite zu.
- **teamFormation**: Beide Spieler stellen alle ihre Spielfiguren auf die linke bzw. rechte Spielfeldseite auf.
- **snapshot**: Ein Snapshot ist ein valider Zustand des Spiels. Jeder Snapshot wird vom Server erzeugt und anschließend an alle Clients verteilt. Der erste Snapshot wird gesendet, nachdem beide Teams aufgestellt sind, als Bestätigung für beide Teamaufstellungen. Snapshots enthalten den letzten deltaBroadcast. Es werden keine Deltas einzeln verschickt.

- `matchFinish`: Markiert das Ende des Spiels.

2.1.1 Definition Sequence

Damit einfacher der sequenzielle Ablauf definiert werden kann. Ist hier folgende Schreibweise und Bedeutung eingeführt.

- `request`: Client sendet an Server
- `broadcast`: Server sendet an alle, Spieler und Zuschauer
- `unicast`: Server sendet an einen Client
- Der Spielverlauf aus Sicht eines Servers. D.h. alle Verbindungen werden geloggt.
- (... markiert ausgelassene Nachrichten)

2.1.1.1 Example

```
# server start
...
request : joinRequest (peter meldet sich an)
...
request : joinRequest (sandra meldet sich an)
...
request : joinRequest (norbert meldet sich an)
...
request : teamConfig (norbert sendet seine teamConfig)
request : teamConfig (sandra sendet ihre teamConfig)
broadcast: matchStart (norbert ist linker Spieler, sandra ist rechter Spieler)
request : teamFormation (sandra stellt ihr Team auf)
request : teamFormation (norbert stellt sein Team auf)
broadcast: snapshot (initialer Snapshot)
broadcast: snapshot (Klatscher1 wurde bewegt)
broadcast: snapshot (Klatscher2 wurde bewegt)
...
broadcast: snapshot (neuer Snapshot mit neuer Spielerposition)
...
broadcast: snapshot (letzter Snapshot)
broadcast: matchFinish (Spiel ist zu Ende und sandra hat gewonnen)
...
# server end
```

3 Kommandozeilenparameter für KI und Server

3.1 Type Declaration

3.1.1 (int/difficulty)

- 0: Maximale Schwierigkeit
- 1: weniger schwer
- 2: noch weniger schwer
- 3: ...
- n: naiv

3.1.2 (int/verbosity)

- 0: keine Ausgabe
- 1: mehr Ausgaben
- 2: noch mehr Ausgaben
- 3: ...
- n: alles

3.2 Definition

3.2.1 KI

3.2.1.1 Verbindlich

zum Start der KI benötigt

```
-a --address <host> Adresse des Servers  
-t --team <path>    Pfad zur Team-Konfiguration  
-l --lobby <string> Name der Lobby
```

3.2.1.2 Optional

Muss implementiert werden, nicht unbedingt zum starten der KI benötigt. Username und password werden in der KI hardgecoded, kann aber beim Start wahlweise auch angegeben werden.

```
-u --username <string>  userName der KI  
-k --password <string>  Passwort der KI  
-h --help                Hilfe  
-p --port <port>         Server-Port (default --port 4488)  
-d --difficulty <int>    Schwierigkeit  
-v --verbosity <int>     Verbosity (Ausführlichkeit der Log-Information)
```

3.2.2 Server

3.2.2.1 Verbindlich

zum Start des Servers benötigt

```
-m --match <path>  Pfad zur Partie-Konfiguration
```

3.2.2.2 Optional

Muss implementiert werden, nicht unbedingt zum starten des Servers benötigt.

```
-h --help                Hilfe  
-p --port <port>         Server-Port  
-v --verbosity <int>     Verbosity (ausführlichkeit der Log-Information)
```


4 Container

4.1 Description

Der Container ist ein valides Json-Objekt und beschreibt den strukturellen Zusammenhang der definierenten jsonpayload Formate. Falls der payloadType hier nicht definiert ist, wird er einfach ignoriert. Falls es Felder im json gibt, welche hier nicht definiert sind, werden sie ebenfalls ignoriert. Der timestamp welcher vom Client geschickt wurde muss nicht richtig sein und wird vom Server mit der Ankunftszeit des Tcp-packets überschrieben.

4.2 Type Declaration

4.2.1 (string/payloadType)

- joinRequest
- loginGreeting
- joinResponse
- sendDebug
- globalDebug
- privateDebug
- matchConfig
- teamConfig
- matchStart
- teamFormation
- snapshot
- pauseRequest
- continueRequest
- pauseResponse
- next
- deltaRequest
- deltaBroadcast
- matchFinish
- getReplay
- replay
- reconnect

4.3 Definition

4.3.1 container

```
{  
  "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",  
  "payloadType" : "(string/payloadType)",  
  "payload": "(json)"  
}
```

4.3.2 Example

Alle Websocket-Messages befinden sich im Container-Format.

```
{
  "timestamp": "2019-04-18 17:20:12.111",
  "payloadType": "examplepayloadTypethatdoentexist",
  "payload": {
    "that": "is",
    "only": [
      "an",
      "example"
    ],
    "everything": "will",
    "be": 8,
    "json": true
  }
}
```

4.3.3 Overview

- request: Client sendet an Server
- broadcast: Server sendet an alle, Spieler und Zuschauer
- unicast: Server sendet an einen Client

request	broadcast	unicast	siehe
joinRequest	loginGreeting	joinResponse	siehe Login
sendDebug	globalDebug	privateDebug	siehe Debug
	matchConfig		siehe MatchConfig
teamConfig			siehe TeamConfig
	matchStart		siehe MatchStart
teamFormation			siehe TeamFormation
	snapshot		siehe Snapshot
pauseRequest, continueRequest	pauseResponse		siehe Pause
	next		siehe Next
deltaRequest			siehe Delta
	matchFinish		siehe MatchFinish
getReplay	replay		siehe Replay
		reconnect	siehe Reconnect

4.4 Debug

4.4.1 Description

Server und Client können jederzeit debug Information senden. Verhalten ist nicht weiter definiert, d.h. Ignorieren, sich mit anderen unterhalten, Fehlermeldungen an den Client senden, Shakespeare vorlesen. Der Server sollte keine unnötigen Debug-Informationen an Clients senden.

4.4.2 Definition

4.4.2.1 sendDebug

```
{
  "information" : "(string)"
}
```

4.4.2.2 globalDebug

```
{
  "information" : "(string)"
}
```

4.4.2.3 privateDebug

```
{
  "information" : "(string)"
}
```

4.4.3 Example

```
{
  "timestamp": "2019-02-11 11:11:11.111",
  "payloadType" : "sendDebug",
  "payload": {
    "information" : "hagrid, I love Magic"
  }
}

{
  "timestamp": "2019-02-11 11:13:12.111",
  "payloadType" : "globalDebug",
  "payload": {
    "information" : "50 points to gryffindor"
  }
}

{
  "timestamp": "2019-02-11 12:10:12.111",
  "payloadType" : "privateDebug",
  "payload": {
    "information" : "you are wizard harry"
  }
}
```

4.5 Login

Die `joinRequest` ist in jedem Fall die erste Nachricht, welche ein Client an den Server sendet. Bei einem erfolgreichen Login befindet man sich in einer Lobby und ist vorerst ein Zuschauer. Falls der Server die angegebenen mods nicht unterstützt, muss die Verbindung vom Server geschlossen werden. Der Login stellt Identitätsabsicherung. Der neuste Login zählt. Eine alte Verbindung muss vom Server geschlossen werden. Es kann immer nur eine Verbindung für einen `userName` existieren. Das password gehört zum `userName`. Der Server speichert den `userName` plus password für den gesamten Server. Der Server unterstützen immer nur eine

Partie gleichzeitig pro lobby. Ein Server muss mindestens eine lobby zugänglich haben. Die default lobby ist hogwarts.

4.5.1 Type Declaration

4.5.1.1 (string/userNameType)

- Darf nur folgende Zeichen enthalten `regex=[a-zA-Z0-9]`
- muss 3 bis 20 Zeichen lang sein. (Anmerkung Zeichen != Byte)

4.5.1.2 (string/lobbyType)

- Darf nur folgende Zeichen enthalten `regex=[a-zA-Z0-9]`
- muss 3 bis 40 Zeichen lang sein. (Anmerkung Zeichen != Byte)

4.5.2 Definition

4.5.2.1 joinRequest

```
{
  "lobby": "(string/lobbyType)",
  "userName": "(string/userNameType)",
  "password" : "(string)",
  "isArtificialIntelligence":"(boolean)",
  "mods":["(string)"]
}
```

4.5.2.2 loginGreeting

```
{
  "userName" : "(string/userNameType)"
}
```

4.5.2.3 joinResponse

```
{
  "message" : "(string)"
}
```

4.5.3 Example

```
{
  "timestamp": "2019-02-11 11:13:12.111",
  "payloadType" : "joinRequest",
  "payload": {
    "lobby": "MemesOnly",
    "userName": "dealwithit",
    "password" : "1337",
    "isArtificialIntelligence":"false",
    "mods": []
  }
}
```

```
}  
}  
{  
  "timestamp": "2019-02-11 11:13:12.112",  
  "payloadType" : "loginGreeting",  
  "payload": {  
    "userName" : "dealwithit"  
  }  
}  
{  
  "timestamp": "2019-02-11 11:13:12.113",  
  "payloadType" : "joinResponse",  
  "payload": {  
    "message" : "welcome, please enjoy"  
  }  
}
```

5 Konfigurationen für Team und Partie

5.1 Description

Die “matchConfig” und “teamConfig” sind Artefakte welche in einem File gespeichert werden. Abgespeichert werden sie im auch Container-Format.

5.2 Type Declaration

5.2.1 (string/name)

- Darf nur folgende Zeichen enthalten `regex=[a-zA-Z0-9]`
- ja, da ist ein Space im Regex.
- muss 3 bis 40 Zeichen lang sein. (Anmerkung Zeichen != Byte)

5.2.2 (string/role)

- `chaser` (Jäger)
- `beater` (Treiber)
- `keeper` (Hüter)
- `seeker` (Sucher)

5.2.3 (string/sex)

- `m` (Männlich)
- `f` (Weiblich)

5.2.4 (string/broom)

- tinderblast
- cleansweep-11
- comet-260
- nimbus-2001
- firebolt

5.2.5 (string/RRGGBB)

- Hexadezimaler RGB-Wert
- Bsp.: C80010

5.2.6 (float/prob)

- Wahrscheinlichkeit als (Komma-)Zahl von 0 bis 1
- Bsp.: 0.87

5.2.7 (string/base64/png)

- Eine PNG-Datei base64-kodiert als String 256x256 Pixel

5.3 Definition

5.3.1 matchConfig (file)

```
{
  "maxRounds": "(int)",
  "timeouts": {
    "playerTurnTimeout": "(int/millisec)",
    "fanTurnTimeout": "(int/millisec)",
    "playerPhaseTime": "(int/millisec)",
    "fanPhaseTime": "(int/millisec)",
    "ballPhaseTime": "(int/millisec)"
  },
  "probabilities": {
    "throwSuccess": "(float/prob)",
    "knockOut": "(float/prob)",
    "foolAway": "(float/prob)",
    "catchSnitch": "(float/prob)",
    "catchQuaffle": "(float/prob)",
    "wrestQuaffle": "(float/prob)",
    "extraMove": {
      "tinderblast": "(float/prob)",
      "cleansweep11": "(float/prob)",
      "comet260": "(float/prob)",
      "nimbus2001": "(float/prob)",

```

```

    "firebolt": "(float/prob)"
  },
  "foulDetection": {
    "flacking": "(float/prob)",
    "haversacking": "(float/prob)",
    "stooging": "(float/prob)",
    "blatching": "(float/prob)",
    "snitchnip": "(float/prob)"
  },
  "fanFoulDetection": {
    "elfTeleportation": "(float/prob)",
    "goblinShock": "(float/prob)",
    "trollRoar": "(float/prob)",
    "snitchSnatch": "(float/prob)"
  }
}
}

```

5.3.2 teamConfig (file)

Die Farben **primary** und **secondary** müssen unterschiedlich sein.

```

{
  "name": "(string/name)",
  "motto": "(string)",
  "colors": {
    "primary": "(string/RRGGBB)",
    "secondary": "(string/RRGGBB)"
  },
  "image": "(string/base64/png)",
  "fans": {
    "goblins": "(int)",
    "trolls": "(int)",
    "elves": "(int)",
    "nifflers": "(int)"
  },
  "players": {
    "seeker": {
      "name": "(string/name)",
      "broom": "(string/broom)",
      "sex": "(string/sex)"
    },
    "keeper": {
      "name": "(string/name)",
      "broom": "(string/broom)",
      "sex": "(string/sex)"
    },
    "chaser1": {
      "name": "(string/name)",
      "broom": "(string/broom)",
      "sex": "(string/sex)"
    }
  }
}

```

```

    },
    "chaser2": {
        "name": "(string/name)",
        "broom": "(string/broom)",
        "sex": "(string/sex)"
    },
    "chaser3": {
        "name": "(string/name)",
        "broom": "(string/broom)",
        "sex": "(string/sex)"
    },
    "beater1": {
        "name": "(string/name)",
        "broom": "(string/broom)",
        "sex": "(string/sex)"
    },
    "beater2": {
        "name": "(string/name)",
        "broom": "(string/broom)",
        "sex": "(string/sex)"
    }
}
}

```

6 Match-Start-Finish

6.1 Description

Wird am Anfang und am Ende der Partie gesendet. Hiermit wird festgelegt, wer “linkes” bzw. “rechtes” Team ist, indem die Configs bzw. die UserNames auf “left” bzw. “right” gematcht werden. In einer Lobby kann nachdem ein MatchFinish ausgelöst wurde wieder ein MatchStart ausgelöst werden, d.h. Alles geht wieder von vorne los.

6.2 Type Declaration

6.2.1 (string/victoryReasonType)

- disqualification
- bothDisqualificationMostPoints
- bothDisqualificationPointsEqualSnitchCatch
- bothDisqualificationPointsEqualLastDisqualification
- mostPoints
- pointsEqualSnitchCatch
- violationOfProtocol

6.3 Definition

6.3.1 matchStart

```
{
  "matchConfig": "(json/matchConfig)",
  "leftTeamConfig": "(json/teamConfig)",
  "rightTeamConfig": "(json/teamConfig)",
  "leftTeamUserName": "(string/userNameType)",
  "rightTeamUserName": "(string/userNameType)"
}
```

6.3.2 matchFinish

```
{
  "endRound": "(int)",
  "leftPoints": "(int)",
  "rightPoints": "(int)",
  "winnerUserName": "(string/userNameType)",
  "victoryReason": "(string/victoryReasonType)"
}
```

7 TeamFormation

7.1 Description

Die Teamformation ist am Anfang des Spiels. Die verbundenen Spieler rechts und links stellen ihre Spielfiguren entsprechend der Spielregeln auf.

7.2 Definition

7.2.1 teamFormation

```
{
  "players": {
    "seeker": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },
    "keeper": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },
    "chaser1": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },
    "chaser2": {
      "xPos": "(int/posx)",

```

```
        "yPos": "(int/posy)"
    },
    "chaser3": {
        "xPos": "(int/posx)",
        "yPos": "(int/posy)"
    },
    "beater1": {
        "xPos": "(int/posx)",
        "yPos": "(int/posy)"
    },
    "beater2": {
        "xPos": "(int/posx)",
        "yPos": "(int/posy)"
    }
}
}
```

8 Snapshot

8.1 Description

Ein Snapshot ist ein valider Zustand des Spiels. Jeder Snapshot wird vom Server erzeugt und anschließend an alle Clients verteilt. Der erste Snapshot wird gesendet, nachdem beide Teams aufgestellt sind, als Bestätigung für beide Teamaufstellungen. Der Server muss Snapshot schicken. Ist für die Leute die Guis gestallten interessant. Der Snapshot allein reicht aus um ein Spiel als spectator zuzuschauen. Manche Informationen sind nur im Snapshot vorliegend, wie z.B. Rundenanzahl und Punktestand.

8.2 Type Declaration

8.2.1 (string/fanType)

- goblin
- troll
- elf
- niffler

8.2.2 (int/posx)

- [0..16]

8.2.3 (int/posy)

- [0..12]

8.2.4 (string/phaseType)

- ballPhase
- playerPhase
- fanPhase
- gameFinish

8.3 Definition

8.3.1 teamSnapshot

```
{
  "points": "(int)",
  "fans": [
    {
      "fanType": "(string/fanType)",
      "banned": "(boolean)"
    }
  ],
  "players": {
    "seeker": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)",
      "banned": "(boolean)",
      "turnUsed" : "(boolean)",
      "knockout" : "(boolean)"
    },
    "keeper": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)",
      "banned": "(boolean)",
      "holdsQuaffle" : "(boolean)",
      "turnUsed" : "(boolean)",
      "knockout" : "(boolean)"
    },
    "chaser1": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)",
      "banned": "(boolean)",
      "holdsQuaffle" : "(boolean)",
      "turnUsed" : "(boolean)",
      "knockout" : "(boolean)"
    },
    "chaser2": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)",
      "banned": "(boolean)",
      "holdsQuaffle" : "(boolean)",
      "turnUsed" : "(boolean)",
      "knockout" : "(boolean)"
    }
  },
}
```

```

"chaser3": {
  "xPos": "(int/posx)",
  "yPos": "(int/posy)",
  "banned": "(boolean)",
  "holdsQuaffle" : "(boolean)",
  "turnUsed" : "(boolean)",
  "knockout" : "(boolean)"
},
"beater1": {
  "xPos": "(int/posx)",
  "yPos": "(int/posy)",
  "banned": "(boolean)",
  "holdsBludger" : "(boolean)",
  "turnUsed" : "(boolean)",
  "knockout" : "(boolean)"
},
"beater2": {
  "xPos": "(int/posx)",
  "yPos": "(int/posy)",
  "banned": "(boolean)",
  "holdsBludger" : "(boolean)",
  "turnUsed" : "(boolean)",
  "knockout" : "(boolean)"
}
}
}

```

8.3.2 snapshot

```

{
  "lastDeltaBroadcast": "(json/deltaBroadcast)",
  "phase": "(string/phaseType)",
  "spectatorUserName": [
    "(string/userNameType)"
  ],
  "round": "(int)",
  "leftTeam": "(json/teamSnapshot)",
  "rightTeam": "(json/teamSnapshot)",
  "balls": {
    "snitch": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },
    "quaffle": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },
    "bludger1": {
      "xPos": "(int/posx)",
      "yPos": "(int/posy)"
    },

```

```

    "bludger2": {
        "xPos": "(int/posx)",
        "yPos": "(int/posy)"
    }
}

```

Falls der Schnatz noch nicht auf dem Spielfeld ist

```

"snitch": {
    "xPos": null,
    "yPos": null
}

```

9 Next

9.1 Description

Es wird an alle Clients eine Benachrichtigung für den nächsten Zug geschickt. Der Zug habende Spieler kann über die entityID herausgefunden werden. Nach einer Pause muss der Spieler wissen wie viel Zeit er hat um seinen Zug zu tätigen, deswegen existiert das Feld `timeout`. Die Antwort des Clients muss innerhalb des `timeouts` am Server eintreffen, sonst wird diese als eine falsch eingetroffene Antwort gewertet und der Client wird daraufhin gekickt. Falls der Client nicht auf das Next antwort passiert nichts. z.B. Spielfigur bleibt stehen. Keine Aktion wird ausgeführt. Mann kann somit das ganze Spiel im idle verharren ohne gekickt zu werden.

9.2 Type Declaration

9.2.1 (string/entityID)

- leftSeeker
- leftKeeper
- leftChaser1
- leftChaser2
- leftChaser3
- leftBeater1
- leftBeater2
- rightSeeker
- rightKeeper
- rightChaser1
- rightChaser2
- rightChaser3
- rightBeater1
- rightBeater2
- snitch
- bludger1
- bludger2
- quaffle
- leftGoblin
- leftTroll

- leftElf
- leftNiffler
- rightGoblin
- rightTroll
- rightElf
- rightNiffler

9.2.2 (string/turnType)

- move
- action
- fan
- removeBan

9.3 Definition

9.3.1 next

```
{
  "turn": "(string/entityID)",
  "type": "(string/turnType)",
  "timeout": "(int/millisec)"
}
```

9.4 Example

Spieler auf der linken Seite wird aufgefordert seinen 3. Jäger zu bewegen, innerhalb der nächsten 7.5 Sekunden

```
{
  "turn": "leftChaser3",
  "type": "move",
  "timeout": 7500
}
```

10 Delta

10.1 Description

- Deltas werden geschickt um einen Zug zu machen (deltaRequest)
- Der Server kickt den Client bei falsch eintreffenden **deltaRequest**
- deltaBroadcast befinden sich nur im snapshot
- null-Felder müssen mitgeschickt werden. (Falls sie hier in der Defintion nicht angegeben sind werden sie immer auf null gesetzt)

deltaType	deltaRequest	deltaBroadcast
snitchCatch		x
bludgerBeating	x	x

deltaType	deltaRequest	deltaBroadcast
quaffleThrow	x	x
snitchSnatch	x	x
trollRoar	x	x
elfTeleportation	x	x
goblinShock	x	x
ban		x
bludgerKnockout		x
move	x	x
wrestQuaffle	x	x
foolAway		x
phaseChange		x
goalPointsChange		x
roundChange		x
skip	x	x

10.2 Type Declaration

10.2.1 (int/posx)

- [0..16]

10.2.2 (int/posy)

- [0..12]

10.2.3 (string/deltaType)

- snitchCatch
- bludgerBeating
- quaffleThrow
- snitchSnatch
- trollRoar
- elfTeleportation
- goblinShock
- ban
- bludgerKnockout
- move
- wrestQuaffle
- foolAway
- phaseChange
- goalPointsChange
- roundChange
- skip

10.2.4 (string/entityID)

- leftSeeker

- leftKeeper
- leftChaser1
- leftChaser2
- leftChaser3
- leftBeater1
- leftBeater2
- rightSeeker
- rightKeeper
- rightChaser1
- rightChaser2
- rightChaser3
- rightBeater1
- rightBeater2
- snitch
- bludger1
- bludger2
- quaffle
- leftGoblin
- leftTroll
- leftElf
- leftNiffler
- rightGoblin
- rightTroll
- rightElf
- rightNiffler

10.3 General Definition

- Der Client setzt das **success** Feld immer auf **null**. (**deltaRequest**)
- Der Server setzt das **success** Feld auf **true** falls es dem Wunsch des Clients entspricht und **false** wenn nicht. (**deltaBroadcast**)
- Der Server setzt das **success** Feld auf **null** falls es sich um eine Serverentscheidung handelt. (**deltaBroadcast**)

10.3.1 deltaRequest, deltaBroadcast

```
{
  "deltaType": "(string/deltaType)",
  "success": "(boolean)",
  "xPosOld": "(int/posx)",
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)",
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)",
  "passiveEntity": "(string/entityID)",
  "phase": "(string/phaseType)",
  "leftPoints": "(int)",
  "rightPoints": "(int)",
  "round": "(int)"
}
```


10.4 DeltaType-Specific Definition

Hier mit null angegebene Felder müssen immer null sein.

10.4.1 snitchCatch (deltaBroadcast)

Wird nur vom Server gesendet(im Snapshot), da durch eine Bewegung vom Sucher auf das Feld des Schnatzes automatisch einen Versuch, den Schnatz zu fangen, ausgelöst wird. So kann auch kein "böser" Client das Spiel auf diese Art verändern. Diese Nachricht muss gesendet werden, wenn ein Sucher auf das Feld mit dem Schnatz zieht. Der Boolean "success" gibt entsprechend an, ob der Schnatz tatsächlich gefangen wurde.

```
{
  "deltaType": "snitchCatch",
  "success": "(boolean)", //If the catch was successful
  "activeEntity": "(string/entityID)", //Seeker the snitch collides with
  "leftPoints": "(int)",
  "rightPoints": "(int)"
}
```

10.4.2 bludgerBeating (deltaRequest)

```
{
  "deltaType": "bludgerBeating",
  "xPosNew": "(int/posx)", //New position of bludger
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Beater that beats the bludger
  "passiveEntity": "(string/entityID)" //Bludger that gets beaten
}
```

10.4.3 bludgerBeating (deltaBroadcast)

```
{
  "deltaType": "bludgerBeating",
  "xPosOld": "(int/posx)", //Old position of bludger
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of bludger
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Beater that beats the bludger
  "passiveEntity": "(string/entityID)" //Bludger that gets beaten
}
```

10.4.4 quaffleThrow (deltaRequest)

```
{
  "deltaType": "quaffleThrow",
  "xPosNew": "(int/posx)", //New position of the quaffle
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)" //Player that throws the quaffle
}
```

10.4.5 quaffleThrow (deltaBroadcast)

```
{
  "deltaType": "quaffleThrow",
  "success": "(boolean)",
  "xPosOld": "(int/posx)", //Old position of the quaffle
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the quaffle
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Player that throws the quaffle
  "passiveEntity": "(string/entityID)" //Player that catches the quaffle, null if noone caught
}
```

10.4.6 snatchSnatch (deltaRequest)

```
{
  "deltaType": "snatchSnatch",
}
```

10.4.7 snatchSnatch (deltaBroadcast)

```
{
  "deltaType": "snatchSnatch",
  "xPosOld": "(int/posx)", //Old position of the snatch
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the snatch
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Niffler that snatches after the snatch
  "passiveEntity": "snatch"
}
```

10.4.8 trollRoar (deltaRequest)

```
{
  "deltaType": "trollRoar",
}
```

10.4.9 trollRoar (deltaBroadcast)

```
{
  "deltaType": "trollRoar",
  "xPosOld": "(int/posx)", //Old position of the quaffle
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the quaffle
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Troll that roar
  "passiveEntity": "(string/entityID)" // entityID that holds the quaffle, null if quaffle
}
```

10.4.10 elfTeleportation (deltaRequest)

```
{
  "deltaType": "elfTeleportation",
  "passiveEntity": "(string/entityID)" //Entity that gets teleported by the elf
}
```

10.4.11 elfTeleportation (deltaBroadcast)

Das Vertändeln wird vor der elfTeleportation geschickt.

Falls ein Spieler einen Quaffel hält und diesen nicht vertändelt wird dieser mit teleportiert.

```
{
  "deltaType": "elfTeleportation",
  "xPosOld": "(int/posx)", //Old position of the passive entity
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the passive entity
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Elf that does the teleportation
  "passiveEntity": "(string/entityID)" //Entity that gets teleported by the elf
}
```

10.4.12 goblinShock (deltaRequest)

```
{
  "deltaType": "goblinShock",
  "passiveEntity": "(string/entityID)" //Passive entity that gets shocked
}
```

10.4.13 goblinShock (deltaBroadcast)

Falls die Entity den Quaffel hält vertändelt sie diesen zuerst mit einem foolAway dannach kommt der goblinShock.

```
{
  "deltaType": "goblinShock",
  "xPosOld": "(int/posx)", //Old position of the passive entity
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the passive entity
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //Goblin that shocks the passive entity
  "passiveEntity": "(string/entityID)" //Passive entity that gets shocked
}
```

10.4.14 ban (deltaBroadcast)

Sobald ein Tor Fällt werden die gebannten Spieler, nach der Fanphase der entsprechenden Runde wieder auf das Spielfeld gelassen. Dabei fordert der Server jede Spielfigur einzeln auf sich auf das

Spielfeld zu begeben. Falls keine Entscheidung getroffen wird entscheidet der Server und setzt die Spielfigur zufällig auf das Spielfeld.

```
{
  "deltaType": "ban",
  "passiveEntity": "(string/entityID)" //Entity that gets banned
}
```

10.4.15 bludgerKnockout (deltaBroadcast)

```
{
  "deltaType": "bludgerKnockout",
  "success": "(boolean)",
  "xPosOld": "(int/posx)", //Old position where the bludger possibly kocks out the passive e
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the bludger[12] after a successful knockout. nu
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)", //The bludger that knocks the passive entity out
  "passiveEntity": "(string/entityID)" //The passive entity that gets knocked out
}
```

10.4.16 move (deltaRequest)

```
{
  "deltaType": "move",
  "xPosNew": "(int/posx)", //New position of the active entity
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)" //Entity that gets a new position
}
```

10.4.17 move (deltaBroadcast)

```
{
  "deltaType": "move",
  "xPosOld": "(int/posx)", //Old position of the active entity
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the active entity
  "yPosNew": "(int/posy)",
  "activeEntity": "(string/entityID)" //Entity that gets a new position
}
```

10.4.18 wrestQuaffle (deltaRequest)

```
{
  "deltaType": "wrestQuaffle",
  "activeEntity": "(string/entityID)" //Entity that wrest the Quaffle
}
```

10.4.19 wrestQuaffle (deltaBroadcast)

```
{
  "deltaType": "wrestQuaffle",
  "success": "(boolean)",
  "xPosOld": "(int/posx)", //Old position of the quaffle
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the quaffle
  "yPosNew": "(int/posx)",
  "activeEntity": "(string/entityID)", //Entity that wrest the Quaffle
  "passiveEntity": "(string/entityID)" //Entity that loses the Quaffle
}
```

10.4.20 foolAway (deltaBroadcast)

```
{
  "deltaType": "foolAway",
  "xPosOld": "(int/posx)", //Old position of the quaffle
  "yPosOld": "(int/posy)",
  "xPosNew": "(int/posx)", //New position of the quaffle
  "yPosNew": "(int/posy)",
  "activeEntity": "quaffle",
  "passiveEntity": "(string/entityID)" //Entity that loses the Quaffle
}
```

10.4.21 phaseChange (deltaBroadcast)

```
{
  "deltaType": "phaseChange",
  "phase": "(string/phaseType)"
}
```

10.4.22 goalPointsChange (deltaBroadcast)

```
{
  "deltaType": "goalPointsChange",
  "leftPoints": "(int)",
  "rightPoints": "(int)",
}
```

10.4.23 roundChange (deltaBroadcast)

```
{
  "deltaType": "roundChange",
  "round": "(int)"
}
```

10.4.24 skip (deltaRequest,deltaBroadcast)

Beendet einen Zug vorzeitig.

```
{
  "deltaType": "skip",
  "activeEntity": "quaffle" //entity that gets skipped
}
```

10.4.25 unban (deltaRequest)

Setzt die neue position eines Spielers nach einem Next removeBan

```
{
  "deltaType": "unban",
  "xPosNew": "(int/posx)", //New position of the player
  "yPosNew": "(int/posx)",
  "activeEntity": "(string/entityID)", //Entity that gets unbanned
}
```

10.5 Example**10.5.1 Example quaffleThrow**

User sendet seine Entscheidung

```
{
  "deltaType": "quaffleThrow",
  "success": null,
  "xPosOld": null,
  "yPosOld": null,
  "xPosNew": 6,
  "yPosNew": 6,
  "activeEntity": "leftChaser1",
  "passiveEntity": null,
  "phase": null,
  "leftPoints": null,
  "rightPoints": null,
  "round": null
}
```

broadcast: Durch Jäger des linken Spielers auf Feld 4,4 werfen, nicht erfolgreich, abgefangen durch Spieler auf Position 5,5. Durch die passiveEntity ist klar schließbar wer den Ball abgefangen hat. Wenn die passiveEntity kein Chaser oder Keeper ist wird danach ein foolAway geschickt.

```
{
  "deltaType": "quaffleThrow",
  "success": false,
  "xPosOld": 4,
  "yPosOld": 4,
  "xPosNew": 5,
  "yPosNew": 5,
  "activeEntity": "leftChaser1",
}
```

```

    "passiveEntity": "rightChaser3",
    "phase": null,
    "leftPoints": null,
    "rightPoints": null,
    "round": null
}

```

10.5.2 Example ban

Server verbannt Spieler leftChaser1 vom Spielfeld

```

{
    "deltaType": "ban",
    "success": null,
    "xPosOld": null,
    "yPosOld": null,
    "xPosNew": null,
    "yPosNew": null,
    "activeEntity": null,
    "passiveEntity": "leftChaser1",
    "phase": null,
    "leftPoints": null,
    "rightPoints": null,
    "round": null
}

```

Der Server fordert auf einen Zug zu machen

```

{
    "turn": "leftChaser1",
    "type": "removeBan",
    "timeout": 7500
}

```

Der Client sendet seine Entscheidung

```

{
    "deltaType": "unban",
    "success": null,
    "xPosOld": null,
    "yPosOld": null,
    "xPosNew": 5,
    "yPosNew": 5,
    "activeEntity": "leftChaser1",
    "passiveEntity": null,
    "phase": null,
    "leftPoints": null,
    "rightPoints": null,
    "round": null
}

```

11 Pause

11.1 Description

Eine Pause-Request kann von allen spielenden Clients an den Server gesendet werden. Außerdem können alle spielenden Clients eine Continue-Request an den Server senden.

Nach jeder Request, broadcastet der Server eine Pause-Response an alle Clients(auch Zuschauer), die einen boolschen Wert enthält, der den aktuellen Pause-Zustand (Spiel pausiert: `pause = true` oder Spiel nicht pausiert: `pause = false`) enthält. Der Server regelt also die komplette Logik und ändert den Pause-Zustand anhand der eingehenden Requests wie im Lastenheft vorgegeben.

Das Lastenheft definiert folgende Serverlogik: "Falls ein mitspielender Client eine Pausierung der Partie wünscht, unterbricht der Server diese, bis einer der Mitspieler anzeigt, dass er weiterspielen möchte. KI-Clients dürfen keine Pausen verlangen."

Der Server überprüft also bei einer Pause-Request ob die empfangene `userName` mit einem spielenden Client übereinstimmt, wenn ja pausiert er das Spiel, falls es nicht schon pausiert ist. Analog überprüft er die `userName` wenn er eine Continue-Request empfängt, wenn diese mit einem spielenden Client übereinstimmt, beendet er die Pause, falls das Spiel nicht schon läuft. In allen Fällen broadcastet der Server eine Pause-Response an alle Clients.

11.2 Definition

11.2.1 pauseRequest

```
{
  "message" : "(string)"
}
```

11.2.2 continueRequest

```
{
  "message" : "(string)"
}
```

11.2.3 pauseResponse

```
{
  "message": "(string)",
  "userName": "(string/userNameType)",
  "pause": "(boolean)"
}
```


12 Reconnect

12.1 Description

Wird mit unicast an Client gesendet, wenn er sich bei laufendem Spiel einloggt. d.h. Nachdem `matchStart` versendet wurde. So erhält der Zuschauer oder Spieler welcher später sich zu dem Spiel verbindet mit nur einem json die gesamten Informationen.

12.2 Definition

12.2.1 reconnect

```
{
  "matchStart": {
    "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
    "payloadType": "matchStart",
    "payload": "(json/matchStart)"
  },
  "snapshot": {
    "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
    "payloadType": "snapshot",
    "payload": "(json/snapshot)"
  },
  "next": {
    "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
    "payloadType": "next",
    "payload": "(json/next)"
  }
}
```

Falls es kein next gibt wird dies so auf null gesetzt

```
"next": null
```

13 Replay

13.1 Description

Das Replay besteht aus statischen Informationen wie `matchConfig`, `leftTeamConfig`, alle Spieler welche mindesten einmal das Spiel betreten haben und einer log-Liste von Deltas. Der Client kann das Replay nur bekommen wenn er sich noch mit der Lobby befindet in welcher das Spiel statt findet. Das Replay wird in einem file persistent abgespeichert. Das Replay ist erst nach dem `MatchFinish` anforderbar. Das `getReplay` bezieht sich immer auf das letzte `MatchFinish`, falls es noch kein Spiel in dieser lobby gab wird der Client gekickt.

13.2 Type Declaration

13.2.1 (string/replayPayloadType) (subset of payloadType)

- deltaBroadcast
- matchFinish (letztes Objekt in dem log array)

13.3 Definition

13.3.1 getReplay

```
{
}
```

13.3.2 replay (file)

```
{
  "lobby": "(string/lobbyType)",
  "startTimestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
  "matchConfig": "(json/matchConfig)",
  "leftTeamConfig": "(json/teamConfig)",
  "rightTeamConfig": "(json/teamConfig)",
  "leftTeamUserName": "(string/userNameType)",
  "rightTeamUserName": "(string/userNameType)",
  "spectatorUserName": [
    "(string/userNameType)"
  ],
  "firstSnapshot": "(json/snapshot)",
  "log": [
    {
      "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
      "payloadType": "(string/replayPayloadType)",
      "payload": "(json)"
    }
  ]
}
```

14 standardisierte Mods (modifications)

14.1 Description

Es ist nicht verpflichtend Mods zu implementieren. Falls der Server und der userClient die Mods unterstützen können diese im login so aktiviert werden

z.B. {"mods":["chat","error","replayWithSnapshot"]}

14.1.1 Overview

request	broadcast	unicast	siehe
sendChat	globalChat		siehe Chat-mod siehe disableGenderBalance-mod
getReplayWithSnapshot		replayWithSnapshot privateError privateWarning	siehe ReplayWithSnapshot-mod siehe Error-mod siehe Warning-mod

14.2 Chat-mod

14.2.1 Description

Damit man sich mit anderen unterhalten kann. Zuschauer und Spielende können in den Chat schreiben. der Server broadcastet dann an alle gerade verbundenen Spieler. Wird im login so aktiviert {"mods":["chat"]}

14.2.2 Definition

14.2.2.1 sendChat

```
{
  "information" : "(string)"
}
```

14.2.2.2 globalChat

```
{
  "senderUserName": "(string/userNameType)",
  "information" : "(string)"
}
```

14.3 DisableGenderBalance-mod

14.3.1 Description

Schaltet die Regel das Teams nur maximal eine bestimmte Anzahl von männlichen und weiblichen Spielfiguren haben dürfen aus. Wird im login so aktiviert {"mods":["disableGenderBalance"]}

14.4 ReplayWithSnapshot-mod

14.4.1 Description

Wie Replay nur alle snapshot und next sind auch enthalten. Wird im login so aktiviert {"mods":["replayWithSnapshot"]}

14.4.2 Type Declaration

14.4.2.1 (string/replayPayloadType2) (subset of payloadType)

- matchStart
- snapshot
- next
- matchFinish (letztes Objekt in dem log array)

14.4.3 Definition

14.4.3.1 getReplayWithSnapshot

```
{
}
```

14.4.3.2 replayWithSnapshot (file)

```
{
  "lobby": "(string/lobbyType)",
  "startTimestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
  "matchConfig": "(json/matchConfig)",
  "leftTeamConfig": "(json/teamConfig)",
  "rightTeamConfig": "(json/teamConfig)",
  "leftTeamUserName": "(string/userNameType)",
  "rightTeamUserName": "(string/userNameType)",
  "spectatorUserName": [
    "(string/userNameType)"
  ],
  "firstSnapshot": "(json/snapshot)",
  "log": [
    {
      "timestamp": "(string/yyyy-MM-dd HH:mm:ss.SSS)",
      "payloadType": "(string/replayPayloadType2)",
      "payload": "(json)"
    }
  ]
}
```

14.5 Error-mod

14.5.1 Description

Bei einem Fehler welcher es erfordert das der Server die Verbindung zum Client abbricht. Der Server sendet über unicast. Wird im login so aktiviert {"mods":["error"]}

14.5.2 Definition

14.5.3 privateError

```
{
  "errorProducingPayloadType" : "(string/payloadType)",
  "information" : "(string)"
}
```

14.6 Warning-mod

14.6.1 Description

Bei einem Fehler welcher vom Server bemerkt wird, der Client wird informiert, weiter passiert nichts. Der Server sendet ein warning Wird im login so aktiviert {"mods":["warning"]}

14.6.2 Definition

14.6.3 privateWarning

```
{
  "warningProducingPayloadType" : "(string/payloadType)",
  "information" : "(string)"
}
```

14.6.4 Error and Warning List

- Error entspricht Verbindungsabbruch.
 - Bei Warning bleibt Verbindung erhalten.
1. (Warning,Error) joinRequest-doublelogin: Benutzer meldet sich bei zwei Computern gleichzeitig an. Dabei wird immer der letzte Login gewertet. Error geht an die alte Anmeldung, Warning an die neue Anmeldung.
 2. (Warning) deltaRequest-timeoutTolerance: Der Server erkennt das ein deltaRequest in den letzten 10 sec richtig gewesen wäre, muss es allerdings verwerfen da es zu spät eingetroffen ist.
 3. (Error) noJson: Server konnte den json nicht parsen
 4. (Error) jsonButNoContainer: Server konnte einen json parsen aber es ist nicht im Container-Format.
 5. (Error) {alle}-invalidPayload: Es konnte das Container-Format erkannt werden, allerdings entspricht die Payload nicht dem gewünschten Format (z.B. fehlende Felder oder Felder haben falsche Typen z.B. String stat Int)

14.6.5 Lobby Mod

Wird so aktiviert {"mods":["lobbies"]}

Der Server sendet *on Connect* d.h. vor der JoinRequest folgende lobbyList

```
{
  "lobbies" : [
    {
      "name": "(string/lobbyName)",
      "matchStarted": boolean,
      "connectedUsers": int
    }
  ],
}
```

14.7 Test

Wird im login so aktiviert {"mods":["test"]}

14.7.1 testFormat

```
{
  "testType":"snapshot-delta-snapshot", // noch keine Anderen Typen
  "createdByTeamXX":"(string)", // z.b. team99
  "description":"(string)", // z.b. Mein erster Test
  "snapshot" : "(json/snapshot)",
  "delta" : "(json/deltaBroadcast)",
  "expectedSnapshot" : "(json/snapshot)"
}
```

14.7.2 testRequest

```
{
  "testType":"snapshot-delta-snapshot", // noch keine Anderen Typen
  "createdByTeamXX":"(string)", // z.b. team99
  "description":"(string)", // z.b. Mein erster Test
  "snapshot" : "(json/snapshot)",
  "delta" : "(json/deltaBroadcast)",
}
```

14.7.3 testResponse

```
{
  "actualSnapshot" : "(json/snapshot)"
}
```

15 appendix

