# TechCity Kodų Karai 2016 – Sąsajos specifikacija

## Dokumento tikslas

Šis dokumentas yra skirtas TechCity Kodų Karų 2016 dalyviams ir negali būti paviešintas kol neprasidėjo Kodų Karų turnyras. Pagrindinis dokumento tikslas – dokumentuoti kliento/serverio sąsają, kurios pagalba sukurtos programos keisis duomenimis su centriniu žaidimų serveriu. Dokumentas aprašo programų autentikacijos taisykles, sąsajos metodų kvietimo seką, visus sąsajos metodus bei pateikia užklausų ir atsakymų pavyzdžius.

## Protokolas

Žaidimų serveris palaiko HTTP/JSON sąsajos tipą, kai užklausos yra vykdomos per HTTP POST metodą, užklausos duomenis perduodant ir atsakymą gaunant JSON formatu POST užklausos turinyje.

## Bazinės užklausos ir atsakymai

Visos užklausos ir atsakymai yra paremti bendra struktūra, kurią kiekvienas iš metodų išplečia.

### Bazinė užklausa

Visų sąsajos metodų užklausos (angl. Request) paveldi bazinę užklausą.

class BaseReq

{

ReqAuth Auth;

}

*Auth* objektas yra naudojamas užklausos ir programos autentikacijai užtikrinti.

### Bazinis atsakymas

class BaseResp

{

string Status;

string Message;

}

*Status* laukas parodo, ar sėkmingai įvykdyta užklausa. *Message* laukas nurodo galimą klaidos priežastį.

* OK – metodas įvykdytas sėkmingai
* WAIT – laukite sekančio ėjimo
* AUTH – autentifikavimo klaida
* FAIL – kitos klaidos

*Message* lauko paskirtis yra pateikti papildomą informaciją. Atsakyme lauko informacija gali būti neapibrėžta (null) arba nurodoma klaidos ar žaidimo statuso pasikeitimo informacija (pvz. komanda neregistruota, komanda laimėjo ir t.t.). Lauko informacija neturėtų būti interpretuojama.

## Autentifikacija

Visos užklausos turi būti autentifikuojamos. Siekiant supaprastinti sąlygas, užklausos turinys nėra įtraukiamas į autentikavimo skaičiavimą. Kiekvienai užklausai *Auth* objektas turi būti užpildomas toliau nurodytu būdu.

class ReqAuth

{

string TeamName;

string ClientName;

int SessionId;

int SequenceNumber;

string AuthCode;

}

|  |  |
| --- | --- |
| **Parametras** | **Aprašymas** |
| *TeamName* | Jūsų komandai suteiktas unikalus (bet ne slaptas) kodas. |
| *ClientName* | Jūsų pačių laisvai pasirinktas programos identifikatorius. Kiekviena programa turi turėti savo unikalų vardą, kad galėtų veikti nepriklausomai nuo kitų tos pačios komandos programų. |
| *SessionId* | Kaip nurodyta diagramoje žemiau rekomenduojama generuoti atsitiktinį sesijos identifikatorių (bet koks sveikas skaičius, 32 bitai). Visuose kituose kvietimuose naudoti tą patį id. |
| *SequenceNumber* | Kiekvienai užklausai turi būti unikalus. Rekomenduojama pradėti nuo 1 ir didinti per 1 kiekvieno kvietimo metu. |
| *AuthCode* | Supaprastintas pranešimo autentikacijos kodas, užtikrinantis tiek komandos, tiek pranešimo autentišku |

## AuthCode skaičiavimas

*AuthCode* reikšmė skaičiuojama tokiu būdu:

|  |
| --- |
| *AuthString := TeamName + ‘:’ + ClientName + ‘:’ + SessionId + ‘:’ + SequenceNumber*  *AuthCode := GetAuthCode(AuthString)*  *GetAuthCode(Payload):*  *SignBytes := GetUTF8Bytes(Payload + Secret)*  *Return LowercaseHex(SHA1(SignBytes))* |

|  |  |
| --- | --- |
| **Parametras / Funkcija** | **Aprašymas** |
| *Secret* | Kiekvienai komandai išduotas slaptažodis. |
| *GetUTF8Bytes* | Teksto užkodavimas UTF-8 formatu, be specialių ženklinimų (be BOM). |
| *SHA1* | Suskaičiuojamas hash pagal SHA-1 algoritmą. |
| *LowercaseHex* | Hash reikšmė, išreikšta šešioliktainiu pavidalu, mažosiomis raidėmis. |

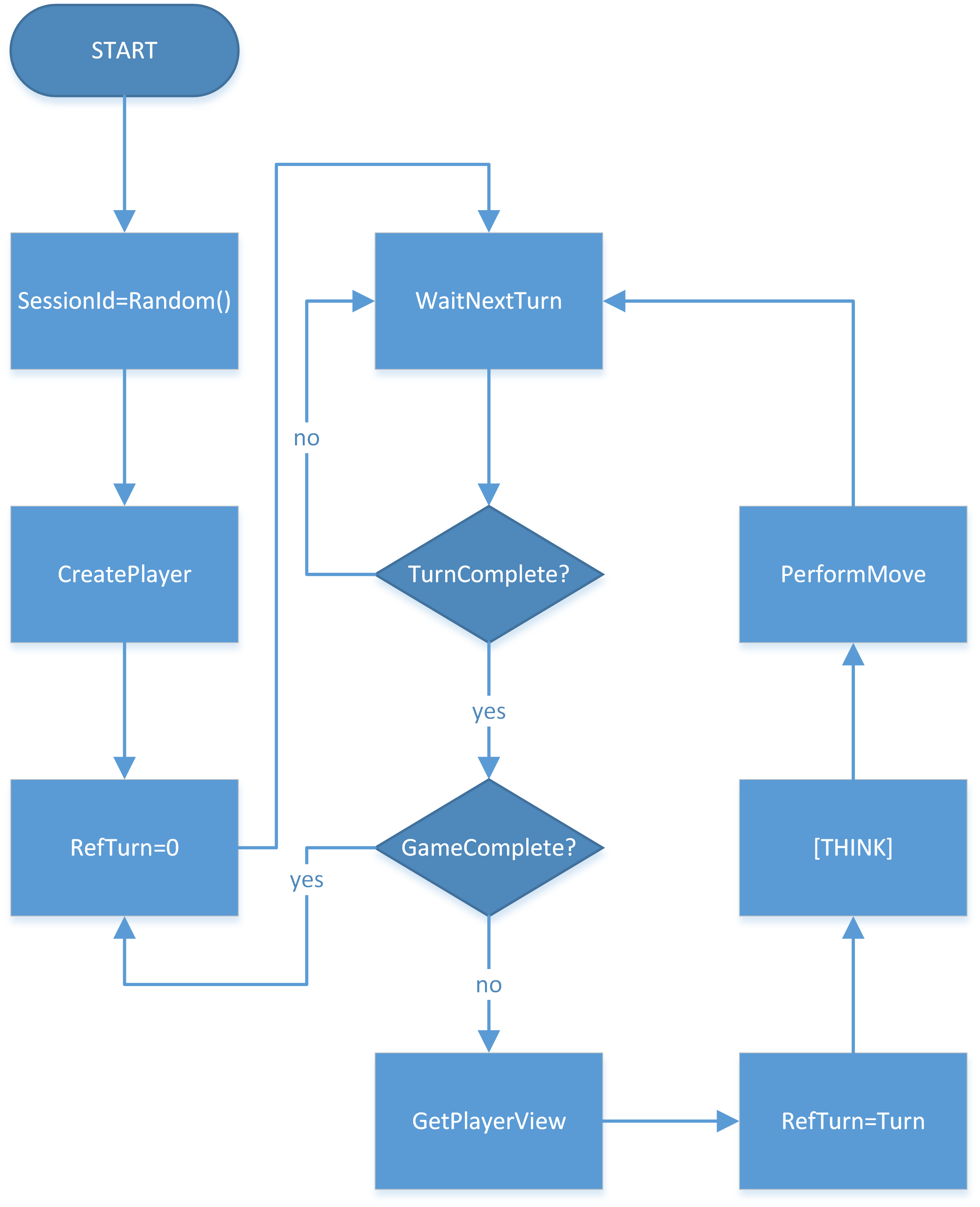
## Neautentikuoti kvietimai

Siekiant palengvinti ir optimizuoti komandų darbą, kiekvienai iš komandų bus suteikiami du prisijungimo vardai (TeamName) – oficialus ir su „-NoAuth“ priesaga. Naudojant „-NoAuth“ priesagą, nebus tikrinamas kvietimų autentiškumas.

**PASTABA**: Varžybų žaidimo etape, kai programos kausis lemiamose dvikovose, kvietimų autentikacija bus privaloma.

## Rekomenduojama sąsajos kvietimo seka

Žemiau yra pateikiama tėkmės diagrama, kuri apibrėžia, kaip turėtų atrodyti pagrindinis programos-žaidėjo sąveikos su žaidimo serveriu algoritmas.



Kiekviena programa turi mokėti prisijungti prie serverio ir veikti nenutrūkstamai iki pat turnyro pabaigos. Pirmiausiai programa turi prisiregistruoti iškviesdama CreatePlayer metodą ir laukti sekančio ėjimo.

Kol komandos programa nedalyvauja dvikovoje, WaitNextTurn metodas grąžins, kad dar negalima atlikti sekančio ėjimo (žiūrėti parametrą TurnComplete?). Kai programa pridedama į dvikovą, tada galima atlikti visus veiksmus: įvertinama situacija žemėlapyje, vykdoma programos skaičiavimo ir sprendimo priėmimo logika, rezultatas perduodamas į serverį. Serveris šiuo atveju laukia kol dvi programos, kurios kovoja tarpusavyje, atsiųs savo sekančius veiksmus, tada atnaujina žaidimo informaciją ir leidžia programoms daryti kitus veiksmus. Jei žaidimas baigiasi, atliekamas nelegalus veiksmas (pvz. už žemėlapio ribų), serveris stabdo žaidimą.

## Sąsajos metodai

### Žaidimo inicializavimas - CreatePlayer

Norint programą užregistruoti žaidime, reikia sukurti žaidėją, kviečiant CreatePlayer metodą.

class CreatePlayerReq : BaseReq

{

// empty

}

class CreatePlayerResp : BaseResp

{

int PlayerId;

}

Serveris įsimena jūsų naudojamą kliento vardą ir visada tam pačiam klientui grąžina tą patį *PlayerId*. Tai leidžia jums pratęsti žaidimą net ir programai „nulūžus“ (su sąlyga, kad ji sugebės pradėti darbą nuo pradžių ir „įsijungti“ į jau vykstantį žaidimą).

|  |  |
| --- | --- |
| **Parametras** | **Aprašymas** |
| Metodo pavadinimas | CreatePlayer |
| Kvietimo nuoroda | /ClientService.svc/json/CreatePlayer |
| **Užklausa (Request)** | |
|  | Užklausos parametrų nėra, tačiau kaip aprašyta aukščiau reikia siųsti autentifikacijos informaciją. |
| **Atsakymas (Response)** | |
| *PlayerId* | Žaidėjui/programai suteikiamas identifikacinis numeris, kurį reikia naudoti visuose žemiau esančiuose metodų kvietimuose. |
| *Status* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |
| *Message* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |

### Sekančio ėjimo laukimas - WaitNextTurn

Kai paaiškėja, kad žaidimas prasidėjo (ar tęsiasi), programos turi pereiti prie ėjimų. Žaidimo pradžioje ir tarp ėjimų reikalinga sinchronizacija, kad visi žaidėjai žaistų žaidimą vienu metu. Tam naudojamas WaitNextTurn metodas.

class WaitNextTurnReq : BaseReq

{

int PlayerId;

int RefTurn;

}

PlayerId lauke reikia nurodyti savo žaidėjo numerį, kurį grąžino CreatePlayer metodas, o RefTurn – ėjimo numerį, kuris buvo įvykdytas paskutinį kartą. Žaidimo pradžioje ar perleidus programą, RefTurn siunčiamas nulis. Tai leidžia programai „susirasti“ savo situaciją žaidime. Kiekvieno kito ėjimo metu RefTurn lauke turi būti siunčiamas ėjimo numeris, grąžintas iš GetPlayerView.

class WaitNextTurnResp : BaseResp

{

bool TurnComplete;

bool GameFinished;

string FinishCondition;

string FinishComment;

bool YourTurn;

}

|  |  |
| --- | --- |
| **Parametras** | **Aprašymas** |
| Metodo pavadinimas | WaitNextTurn |
| Kvietimo nuoroda | /ClientService.svc/json/WaitNextTurn |
| **Užklausa (Request)** | |
| PlayerId | Žaidėjo numeris, kurį grąžino CreatePlayer metodas. |
| RefTurn | Ėjimo numeris, kuris buvo įvykdytas paskutinį kartą. Žaidimo pradžioje ar perleidus programą, RefTurn siunčiamas nulis. Tai leidžia programai „susirasti“ savo situaciją žaidime. Kiekvieno kito ėjimo metu RefTurn lauke turi būti siunčiamas ėjimo numeris, grąžintas iš GetPlayerView. |
| **Atsakymas (Response)** | |
| *TurnComplete* | *WaitNextTurn* metodas laukia iki tol, kol užsibaigia ėjimas arba kol praeina tam tikras laikas. Jei ėjimas neužsibaigia per tam tikrą laiką, grąžinamas *TurnComplete=false*. Tuomet reikia tiesiog kartoti *WaitNextTurn* kvietimą. Visais kitais atvejais *TurnComplete=true* ir kiti laukai įgauna prasmę. |
| *GameFinished* | Jei *true*, žaidimas duotam žaidėjui pasibaigė. Reikia išeiti iš žaidimo ir grįžti prie sekančio žaidimo laukimo. |
| *FinishCondition* | Jei žaidimas pasibaigė, suteikia informaciją apie žaidimo baigtį duotam žaidėjui:   * *WON* – pergalė * *LOST* – pralaimėjimas * *DRAW* – lygiosios |
| *FinishComment* | jei žaidimas pasibaigė pralaimėjimu, nurodoma detalesnė priežastis |
| *Status* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |
| *Message* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |

### Žaidimo informacijos užklausimas - GetPlayerView

Norint gauti žaidimo informaciją – žėmėlapį, žaidėjų pozicijas ir pan. – reikia kviesti GetPlayerView, metodą. Šį metodą rekomenduojama kviesti tada, kai *WaitNextTurn* grąžina *TurnComplete = True,* t.y. visi žaidėjai jau atlikę ėjimus.

**Pastaba:** GetPlayerView ir PerformMove metodai Tecmano ir Vaiduoklių **koordinates skaičiuoja nuo 0**.

class GetPlayerViewReq : BaseReq

{

int PlayerId;

}

class GetPlayerViewResp : BaseResp

{

int Turn;

string Mode;

EnMapData Map;

EnPoint TecmanPosition;

List<EnPoint> GhostPositions;

EnPoint TecmanPosition;

List<EnPoint> PreviousTecmanPosition;

string Status;

string Message;

}

Šio metodo atsakyme naudojamos papildomos klasės visos žemėlapio situacijos aprašymui.

class EnMapData

{

int Width;

int Height;

List<String> Rows;

}

class EnPoint

{

int Row;

int Col;

}

|  |  |
| --- | --- |
| **Parametras** | **Aprašymas** |
| Metodo pavadinimas | GetPlayerView |
| Kvietimo nuoroda | /ClientService.svc/json/GetPlayerView |
| **Užklausa (Request)** | |
| *PlayerId* | Žaidėjo numeris, kurį grąžino CreatePlayer metodas. |
| **Atsakymas (Response)** | |
| *Turn* | Ėjimo numeris, kuris buvo įvykdytas paskutinį kartą. |
| *Mode* | Žaidimo rėžimas, kuriuo dalyvauja programa. Galimos reikšmės:   * GHOSTS - komandos programa žaidime dalyvauja Vaiduoklių rėžime. * TECMAN - komandos programa žaidime dalyvauja Tecman’o rėžime |
| *Map* | Žaidimo žemėlapio informacija, kur:   * Width - žemėlapio plotis * Height - žemėlapio aukštis * Rows - žemėlapio informacija, kuri nusako kiekvienos žemėlapio pozicijos informaciją. Tai yra Width x Height dydžio matrica, kurios kiekvienas elementas nusako atitinkamos žemėlapio pozicijos būseną. Galimi variantai: * # - siena * (tarpas) - praėjimas * . - sausainiukas   **Pastaba**: jei žemėlapio krašte yra ne siena, o praėjimas, tai reiškia, kad toje vietoje galimas simetriškas perėjimas į kitą žemėlapio kraštą. |
| *TecmanPosition* | Grąžinama Tecmano pozicija žemėlapyje, kur:   * Row - y koordinatė (matricos eilutė) * Col - x koordinatė (matricos stulpelis) |
| *PreviousTecmanPosition* | Prieš tai buvusi Tecmano pozicija žemėlapyje. |
| *GhostPositions* | Vaiduoklių pozicijos žemėlapyje. Kadangi vaiduoklių yra daugiau nei vienas (klasiknis variantas 4 vaiduokliai, bet turnyro metu organizatoriai gali naudoti ir kitą vaiduoklių skaičių) atsiunčiamas sąrašas pozicijų (Position tipo objektas), kur:   * Row - y koordinatė (matricos eilutė) * Col - x koordinatė (matricos stulpelis) |
| *PreviousGhostPositions* | Prieš tai buvusios vaiduoklių pozicijos, kur:   * Row - y koordinatė (matricos eilutė) * Col - x koordinatė (matricos stulpelis) |
| *Status* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |
| *Message* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |

### Atlikti ėjimą žaidime - PerformMove

Šis metodas naudojamas norint atlikti ėjimą žaidime. Atlikus PerformMove rekomenduojama iškviesti WaitNextTurn metodą.

class PerformMoveRequest : BaseReq

{

int PlayerId;

List<Position> Positions;

}

class PerformMoveResponse : BaseResp

{

string Status;

string Message;

}

|  |  |
| --- | --- |
| **Parametras** | **Aprašymas** |
| Metodo pavadinimas | PerformMove |
| Kvietimo nuoroda | /ClientService.svc/json/PerformMove |
| **Užklausa (Request)** | |
| *PlayerId* | Žaidėjo numeris, kurį grąžino CreatePlayer metodas. |
| *Positions* | Siunčiamas sąrašas naujų pozicijų. Tecmano atveju tai bus viena pozicija, taigi vienas Position tipo objektas. Vaiduoklių atveju reikia siųsti sąrašą, priklausomai nuo vaiduoklių kiekio žaidime. |
| **Atsakymas (Response)** | |
| *Status* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |
| *Message* | Žiūrėti ankstesnį skyrių „Bazinis atsakymas“. |

# Priedas A - Sąsajos kvietimo pavyzdžiai

## CreatePlayer

**Pavyzdys Nr. 1 - Nesėkmingas CreatePlayer kvietimas**

|  |
| --- |
| **Request:**  {  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 1,  "AuthCode": "33ca3e76f07949c59efbb8c9ffb0fad15d692eba"  }  } |
| **Response:**  {  "PlayerId": 0,  "Status": "AUTH",  "Message": "AuthCode mismatch"  } |

Šiuo atveju tikriausiai neteisingai apskaičiuotas autentifikacijos kodas. To priežastis gali būti neteisingai įgyvendintas autentifikacijos algoritmas, neteisingai perduodamas slaptažodis ir pan.

**Pavyzdys Nr. 2 - Nesėkmingas kvietimas, kai komanda neregistruota**

|  |
| --- |
| **Request:**  {  "Auth": {  "TeamName": "Amadeus",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 2,  "AuthCode": "89088e3b26c39c9bc841b30be727112c249f8dc2"  }  } |
| **Response:**  {  "PlayerId": 0,  "Status": "FAIL",  "Message": "Team is not registered (4 known)"  } |

Šiuo atveju neteisingai perduodama komandos (parametras TeamName) reikšmė ir serveris neatpažįsta tokio pavadinimo.

**Pavyzdys Nr. 3 - Sėkmingas kvietimas**

|  |
| --- |
| **Request:**  {  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 1,  "AuthCode": "fb6cbcdd9954ca752d247785e24ab37212200277"  }  } |
| **Response:**  {  "PlayerId": 1,  "Status": "OK",  "Message": null  } |

Šiuo kvietimu pavyko užregistruoti programą kaip žaidėją ir serveris grąžino PlayerId reikšmę.

## WaitNextTurn

**Pavyzdys Nr. 1 - nesėkmingas WaitNextTurn iškvietimas, kai programa neregistruota**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "RefTurn": 0,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 2,  "AuthCode": "f6950eb3126e831f25248035bf6e709384564023"  }  } |
| **Response:**  {  "TurnComplete": false,  "GameFinished": false,  "FinishCondition": null,  "FinishComment": null,  "Status": "FAIL",  "Message": "Player not found"  } |

Iš pavyzdžio pateikto viršuje matosi, kad programa iškvietė WaitNextTurn dar neprisiregistravus serveryje arba perdavė nekorektišką PalyerId.

**Pavyzdys Nr.2 - sėkmingas WaitNextTurn, bet sekančio ėjimo atlikti dar negalima**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "RefTurn": 0,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 2,  "AuthCode": "f6950eb3126e831f25248035bf6e709384564023"  }  } |
| **Response:**  {  "TurnComplete": false,  "GameFinished": false,  "FinishCondition": "Play",  "FinishComment": null,  "Status": "OK",  "Message": null  } |

**Pavyzdys Nr. 3 - sėkmingas WaitNextTurn kvietimas, galima pradėti sekančio ėjimo skaičiavimus**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "RefTurn": 0,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 7,  "AuthCode": "fe76b503f53af934ae90f487b7f943f5d2bdf97b"  }  } |
| **Response:**  {  "TurnComplete": true,  "GameFinished": false,  "FinishCondition": "Play",  "FinishComment": null,  "Status": "OK",  "Message": null  } |

**Pavyzdys Nr. 4 - Žaidimas baigtas ir jo rezultato aprašymas**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "RefTurn": 8,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 32,  "AuthCode": "2ef0ea9da3a3e0b379e34fe65fb9b50915e70d7d"  }  } |
| **Response:**  {  "TurnComplete": true,  "GameFinished": true,  "FinishCondition": "Lost",  "FinishComment": "Tecman captured by ghosts",  "Status": "OK",  "Message": null  } |

## GetPlayerView

**Pavyzdys Nr. 1 - Sėkmingas GetPlayerView iškvietimas**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 8,  "AuthCode": "700b9464dc73a7b742343e2ba0b5b605a65c4eb7"  }  } |
| **Response:**  {  "Turn": 1,  "Mode": "TECMAN",  "Map": {  "Width": 10,  "Height": 7,  "Rows": [  "##########",  "# #",  "# #",  "# #",  "#.######.#",  "#.... ...#",  "##########"  ]  },  "TecmanPosition": {  "Row": 5,  "Col": 5  },  "PreviousTecmanPosition": {  "Row": 5,  "Col": 4  },  "GhostPositions": [  {  "Row": 2,  "Col": 3  },  {  "Row": 2,  "Col": 4  },  {  "Row": 2,  "Col": 5  },  {  "Row": 2,  "Col": 6  }  ],  "PreviousGhostPositions": [  {  "Row": 2,  "Col": 3  },  {  "Row": 2,  "Col": 4  },  {  "Row": 2,  "Col": 5  },  {  "Row": 2,  "Col": 6  }  ],  "Status": "OK",  "Message": null  } |

**Pavyzdys Nr. 2 - Nesėkmingas GetPlayerView iškvietimas**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 31,  "AuthCode": "edb3f1e04311a888eddd8eda6b9dd66a7ea037dc"  }  } |
| **Response:**  {  "Turn": 0,  "Mode": null,  "Map": null,  "TecmanPosition": null,  "GhostPositions": null,  "PreviousGhostPositions": null,  "Status": "WAIT",  "Message": "Wait for next turn"  } |

## PerformMove

**Pavyzdys Nr. 1 - Sėkmingas TECMAN’o PerformMove iškvietimas**

|  |
| --- |
| **Request:**  {  "PlayerId": 1,  "Positions": [  {  "Row": 5,  "Col": 6  }  ],  "Auth": {  "TeamName": "Auth",  "ClientName": "Test1",  "SessionId": 5656,  "SequenceNumber": 6,  "AuthCode": "894afe4b5ffd73472eed8c7cdf1a605ccd998fd9"  }  } |
| **Response:**  {  "Status": "OK",  "Message": null  } |

**Pavyzdys Nr. 2 - Sėkmingas Vaiduoklių PerformMove iškvietimas**

|  |
| --- |
| **Request:**  {  "PlayerId": 2,  "Positions": [  {  "Row": 3,  "Col": 1  },  {  "Row": 3,  "Col": 6  },  {  "Row": 3,  "Col": 7  },  {  "Row": 3,  "Col": 8  }  ],  "Auth": {  "TeamName": "Auth",  "ClientName": "Test2",  "SessionId": 9819214,  "SequenceNumber": 399,  "AuthCode": "da65c0b181a8bbb0c3573a678427686081a04933"  }  } |
| **Response:**  {  "Status": "OK",  "Message": null  } |