API Spec

2024-07-15

realCity Query Server API

Overview

Specification for the realCity Query Server API.

Version

1.0.1

GET /{dialect}/api/where/alert-search

Request Parameters:

query: string; // A keresési feltétel, amit a zavar fejlécével, leírásával, vagy azonosítójával illesztünk.
start: integer; // A keresés id –çFW vallumának eleje epoch másodpercben.
end: integer; // A keresési id –çFW vallum vége epoch másodpercben.
minResult: integer; // A visszaadott elemek minimális száma.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszatér a `query` paraméterben megadott keresési feltételnek megfelel ¦ varokkal, amik aktívak a megadott id –çFW vallumnban.

Content: application/json | AlertSearchMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesTransitSearch;
}
```

GET /{dialect}/api/where/arrivals-and-departures-for-location

Request Parameters:

groupLimit: integer; // Menetrendi adatok maximális száma egy csoportban.
clientLon: number; // A kliens hosszúsági koordinátája.
clientLat: number; // A kliens szélességi koordinátája.
minutesBefore: integer; // A lekérdezési id lak a `time` paraméter el GB Væàyi perccel indul.
minutesAfter: integer; // A lekérdezési id lak a `time` paraméter után ennyi perccel ér véget.
stopId: Array<string>; // A megállók azonosítóinak listája, amelyekhez a lekérést végezzük.
includeRouteId: Array<string>; // A válasz sz .—>—&R † 7¦î ÇB ® atok azonosítóinak listája.
time: integer; // A lekérdezés kiértékeléséhez használt id öçBà Alapértelmezetten az aktuális szerverid à
onlyDepartures: boolean; // Ha igaz akkor csak az érkezési (és el &V¦VÇ ett érkezési) id ² æVÒ szerepelnek a válaszban.

limit: integer; // A visszaadott indulási és érkezési id ² Æ—7N ® æ ² Ö †–Þ Æ—2 †÷77¦ à
lat: number; // A helyszín középpontjának szélessgéi koordinátája.
lon: number; // A helyszín középpontjának hosszúsági koordinátája.
latSpan: number; // A lekérési terület szélességi íve. (Terület szélessége: `lat +/- latspan`).
lonSpan: number; // A lekérési terület hosszúsági íve. (Terület hosszúsága: `lon +/- lonSpan`).
radius: integer; // Ha a `latSpan` vagy a `longSpan` nincs kitöltve, a keresési terület a középpontól számított `radius` méter távolság mind a négy irányban.
query: string; // A válasz sz .—>—&R † 7¦î ÇB °eresési kifejezés.
minResult: integer; // A visszaadott elemek minimális száma.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja egy területhez tartozó érkezési és indulási id °et járatazonosítók és célállomások szerint csoportosítva.

Content: application/json | ArrivalsAndDeparturesForLocationOTPMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitListEntryWithReferencesTransitDepartureGroup;
}
```

GET /{dialect}/api/where/arrivals-and-departures-for-stop

Request Parameters:

minutesBefore: integer; // A lekérdezési id lak a `time` paraméter el GB Væàyi perccel indul.
minutesAfter: integer; // A lekérdezési id lak a `time` paraméter után ennyi perccel ér véget.
stopId: Array<string>; // A megállók azonosítóinak listája, amelyekhez a lekérést végezzük.
includeRouteId: Array<string>; // A válasz sz .—>—&R † 7¦î ÇB ® atok azonosítóinak listája.
time: integer; // A lekérdezés kiértékeléséhez használt id öçBà Alapértelmezetten az aktuális szerverid à
onlyDepartures: boolean; // Ha igaz akkor csak az érkezési (és el &V¦VÇ ett érkezési) id ² æVÒ szerepelnek a válaszban.
limit: integer; // A visszaadott indulási és érkezési id ² Æ—7N ® æ ² Ö †–Þ Æ—2 †÷77¦ à
lat: number; // A helyszín középpontjának szélessgéi koordinátája.
lon: number; // A helyszín középpontjának hosszúsági koordinátája.
latSpan: number; // A lekérési terület szélességi íve. (Terület szélessége: `lat +/- latspan`).
lonSpan: number; // A lekérési terület hosszúsági íve. (Terület hosszúsága: `lon +/- lonSpan`).
radius: integer; // Ha a `latSpan` vagy a `longSpan` nincs kitöltve, a keresési terület a középpontól számított `radius` méter távolság mind a négy irányban.
query: string; // A válasz sz .—>—&R † 7¦î ÇB °eresési kifejezés.
minResult: integer; // A visszaadott elemek minimális száma.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Lekéri a megállóhoz tartozó érkezési és indulási id °et.

Content: application/json | ArrivalsAndDeparturesForStopOTPMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitEntryWithReferencesTransitArrivalsAndDepartures;
}
```

GET /{dialect}/api/where/bicycle-rental

Request Parameters:

appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszatér az összes kerékpárkölcsönz ÆÆöÞ 76 Âà

Content: application/json | BicycleRentalMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitListEntryWithReferencesTransitBikeRentalStation;
}
```

GET /{dialect}/api/where/booking-redirect

Request Parameters:

routeId: string; // A járat azonosítója.
directionId: string; // A járat iránya.
tripId: string; // A menet azonosítója.
serviceDate: string; // A menet üzem napja.
boardStopId: string; // A felszállási megálló azonosítója.
alightStopId: string; // A leszállási megálló azonosítója.
version: ApiVersion; // Az API verziója.
appVersion: string; // A kliensalkalmazás verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.

Response 302:

Átírányit a foglalási oldalra.

GET /{dialect}/api/where/metadata

Request Parameters:

time: integer; // A lekérés kiértékelésének id öçF¦ W ö6, Þ 6öG W&6&Vâ † ¢ ·N×b ¦ varok lekéréséhez szükséges).A szerver ideje az alapértelmezett értéke.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

A szerver metaadataival tér vissza.

Content: application/json | MetadataResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesTransitMetadata;
}
```

GET /{dialect}/api/where/multi-route-details

Request Parameters:

routeId: Array<string>; // A lekért járatok azonosítói.
date: string; // Az aktív zavarok lekéréséhez használt dátum. Alapértelmezett értéke az aktuális nap.
related: boolean; // Ha igaz, akkor az útvonalhoz tartozó kapcsolódó útvonalak is szerepelnek a válaszban.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Az összes járatot visszaadja a megadott azonosítókhoz.

Content: application/json | MultiRouteDetailsMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitListEntryWithReferencesTransitRouteDetails;
}
```

Response 400:

A `date` paramétert nem sikerült beolvasni.

Response 404:

Ismeretlen járat azonosító.

POST /{dialect}/api/where/onboard-depart-search

Request Parameters:

appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Request Body:

Content: application/json | Array<OnboardDepartPosition>

```
{
  lat: number; // A szélességi koordinátája egy pozició pontnak.
  lon: number; // A hosszúsági koordinátája egy pozició pontnak.
  timestamp: integer; // Az id öçF¦ Vw' ÷¦–6Ÿ2 öçFæ ² W ö6, Þ 6öG W&6&Vâà
  accuracy?: number; // A pontossága egy pozició pontnak méterben.
  speed?: number; // A sebesség egy pozició pontban m/s-ban.
}
```

Response 200:

Visszatér az utas útvonalára illeszked ® m `ekkel.

Content: application/json | OnboardDepartSearchMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitListEntryWithReferencesTransitVehicle;
}
```

GET /{dialect}/api/where/plan-trip

Request Parameters:

version: ApiVersion; // Az API verziója.
appVersion: string; // A kliensalkalmazás verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.
date: string; // A dátum, amikor a tervezés indul (`arriveBy` esetén érkezik).
time: string; // Az id Â Ö–°or a tervezés indul (`arriveBy` esetén érkezik).
fromPlace: string; // Az indulási hely adatai `név::hely` formában. A `név` az adott hely tetsz ÆVvW2 neve, amely az útiterv kiindulási pontjának lesz a neve (`leg.from.name`). A `hely` lehet koordináta

`lat,lon` formában, vagy a geocoder API által visszaadott `vertex` azonosítója.
toPlace: string; // Az érkezési hely adatai `név::hely` formában. A `név` az adott hely tetsz ÆVvW2 neve, amely az útiterv érkezési pontjának lesz a neve (`leg.to.name`). A `hely` lehet koordináta `lat,lon` formában, vagy a geocoder API által visszaadott `vertex` azonosítója.
mode: Array<TraverseMode>; // A tervez Ö–Ç•en közlekedési módokat használjon. A kerékpár kölcsönzés a `BICYCLE,WALK` paraméterekkel kapcsolható be.
shouldBuyTickets: boolean; // Az els ® m &R 7® ÆÎ 2 VÁQtt történjen-e jegyvásárlás.
showIntermediateStops: boolean; // A válasz tartalmazza-e a járatok által érintett köztes megállókat is.
arriveBy: boolean; // Érkezési id &R FW vezzünk-e a megadott `date` és `time` paraméterrel.
wheelchair: boolean; // Az összes járat alacsonypadlós legyen-e.
triangleSafetyFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít a biztonság. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
triangleSlopeFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít az útvonal síksága. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
triangleTimeFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít a gyorsaság. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
optimize: string; // A keres Ö—&R ÷ F–Ö Æ—® Æ¦ ¢ §G`onalakat.
walkProfile: WalkProfile; // A séta sebességét meghatározó profil.
numItineraries: integer; // Legfeljebb hány útvonal szerepeljen a válaszban.

Response 200:

A megadott paraméterekkel tervez útvonalat.

Content: application/json | PlanTripResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesResponse;
}
```

GET /{dialect}/api/where/plan-access

Request Parameters:

version: ApiVersion; // Az API verziója.
appVersion: string; // A kliensalkalmazás verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.
date: string; // A dátum, amikor a tervezés indul (`arriveBy` esetén érkezik).
time: string; // Az id Â Ö–°or a tervezés indul (`arriveBy` esetén érkezik).
fromPlace: string; // Az indulási hely adatai `név::hely` formában. A `név` az adott hely tetsz ÆVvW2 neve, amely az útiterv kiindulási pontjának lesz a neve (`leg.from.name`). A `hely` lehet koordináta `lat,lon` formában, vagy a geocoder API által visszaadott `vertex` azonosítója.
toPlace: string; // Az érkezési hely adatai `név::hely` formában. A `név` az adott hely tetsz ÆVvW2 neve, amely az útiterv érkezési pontjának lesz a neve (`leg.to.name`). A `hely` lehet koordináta `lat,lon` formában, vagy a geocoder API által visszaadott `vertex` azonosítója.
mode: Array<TraverseMode>; // A tervez Ö–Ç•en közlekedési módokat használjon. A kerékpár kölcsönzés a `BICYCLE,WALK` paraméterekkel kapcsolható be.
shouldBuyTickets: boolean; // Az els ® m &R 7® ÆÎ 2 VÁQtt történjen-e jegyvásárlás.
showIntermediateStops: boolean; // A válasz tartalmazza-e a járatok által érintett köztes megállókat is.

arriveBy: boolean; // Érkezési id &R FW vezzünk-e a megadott `date` és `time` paraméterrel.
wheelchair: boolean; // Az összes járat alacsonypadlós legyen-e.
triangleSafetyFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít a biztonság. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
triangleSlopeFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít az útvonal síksága. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
triangleTimeFactor: number; // Kerékpáros háromszög módú tervezés esetén mennyire számít a gyorsaság. 0 és 1 közötti szám, és 1-et kell kiadnia a másik két faktor összegével.
optimize: string; // A keres Ö—&R ÷ F–Ö Æ—® Æ¦ ¢ §G`onalakat.
walkProfile: WalkProfile; // A séta sebességét meghatározó profil.
numItineraries: integer; // Legfeljebb hány útvonal szerepeljen a válaszban.

Response 200:

A megadott paraméterekkel tervez megállóba útvonalat.

Content: application/json | PlanTripResponse

{
currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
version: integer; // A válasz API verziója.
status: Status;
code: integer; // A válasz státusz kódja.
text: string; // A válasz szövege.
data: TransitEntryWithReferencesResponse;
}

GET /{dialect}/api/where/references

Request Parameters:

agencyId: Array<string>; // A szolgáltató ID-je.
alertId: Array<string>; // A zavar ID-je.
routeId: Array<string>; // A járat ID-ja.
stopId: Array<string>; // A megálló ID-ja.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

ID alapú referencia lekérdezés.

Content: application/json | ReferencesMethodResponse

{
limitExceeded: boolean; // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és érkezési id N× W?¢ ÆV°éréseknél használjuk.
entry: ReferencesMethodResult;
references: TransitReferences;
class: string; // Az adat típusa. Egy entitás esetén "entryWithReferences".
}

GET /{dialect}/api/where/route-details-for-stop

Request Parameters:

stopId: string; // A lekért megálló azonosítója.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

A megállóhoz és a testvérmegállóihoz tartozó összes járattal tér vissza.

Content: application/json | RouteDetailsForStopMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitListEntryWithReferencesTransitRoute;
}
```

Response 404:

Ismeretlen megálló azonosító.

GET /{dialect}/api/where/route-details

Request Parameters:

routeId: string; // A járat ID-ja.
date: string; // Az aktív zavarok lekéréséhez használt dátum. Alapértelmezett értéke az aktuális nap.
related: boolean; // Ha igaz, akkor az útvonalhoz tartozó kapcsolódó útvonalak is szerepelnek a válaszban.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja a megadott ID-val rendelkez ® atot.

Content: application/json | RouteDetailsMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesTransitRouteDetails;
}
```

Response 400:

Rossz formátumú a `date` paraméter.

Response 404:

Az ID-hoz nem található járat.

GET /{dialect}/api/where/schedule-for-stop

Request Parameters:

stopId: Array<string>; // A releváns megállók azonosítói.
date: string; // A kért dátum.
onlyDepartures: boolean; // Igaz esetén az érkezési és el &V¦VÇ ett érkezési id ² ¶' ÆW7¦æV² † w—`a a válaszból.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja a menetrendet az adott megállóhoz.

Content: application/json | ScheduleForStopOTPMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesTransitSchedule;
}
```

Response 400:

Nem sikerült beolvasni a dátumot.

Response 404:

Ismeretlen stopId.

GET /{dialect}/api/where/search

Request Parameters:

query: string; // Sz !Qfeltétel, amire illesztve vannak a zavarok, járatok és megállók.
lat: number; // A látható térkép középpontjának szélessgéi koordinátája.
lon: number; // A látható térkép középpontjának hosszúsági koordinátája.
minResult: integer; // A visszaadott elemek minimális száma.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja az olyan zavarokat, megállókat és járatokat, amelyek tulajdonságai illeszkednek a megadott sz !Qrfeltételre.

Content: application/json | SearchMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitEntryWithReferencesTransitSearch;
}
```

GET /{dialect}/api/where/statistics

Request Parameters:

dialect: Dialect; // Az API referenciáinak dialektusa.

Response 200:

Egyszer `égpont hibakereséshez, ami egy "OK" választ ad vissza.

Content: application/json | StatisticsResponse

```
{
  result: string;  // Egyszer  `álasz.
}
```

GET /{dialect}/api/where/stops-for-location

Request Parameters:

lat: number; // A helyszín középpontjának szélessgéi koordinátája.
lon: number; // A helyszín középpontjának hosszúsági koordinátája.
latSpan: number; // A lekérési terület szélességi íve. (Terület szélessége: `lat +/- latspan`).
lonSpan: number; // A lekérési terület hosszúsági íve. (Terület hosszúsága: `lon +/- lonSpan`).
radius: integer; // Ha a `latSpan` vagy a `longSpan` nincs kitöltve, a keresési terület a középpontól számított `radius` méter távolság mind a négy irányban.
query: string; // A válasz sz .—>—&R † 7¦î ÇB °eresési kifejezés.
minResult: integer; // A visszaadott elemek minimális száma.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Egy megállólistát ad vissza az adott helyhez. Ha a lat vagy a lon null, akkor az összes megálló bekerül a válaszba.

Content: application/json | StopsForLocationResponse

```
{
```

```
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitStop;
}
```

GET /{dialect}/api/where/ticketing-locations

Request Parameters:

ifModifiedSince: integer; // Csak azokat az elemeket adjuk vissza, amik módosultak az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ha nincs változás üres listákkal tér vissza. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
full: boolean; // Ha igaz, és az adat változott az `ifModifiedSince` query paraméterben megadott id öçB 7F Â ¶°or az összes adat visszaadásra kerül. Ha nincs megadva csak a módosult elemek szerepelnek a válaszban. Csak az `ifModifiedSince` query paraméterrel együtt szerepelhet, mivel az `If-Modified-Since` fejléc minden esetben a teljes választ tartalmazza.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja a jegyértékesít †VÇ•eket és jegytípusokat.

Content: application/json | TicketingMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitTicketing;
}
```

Response 304:

Nem változott az adat az `ifModifiedSince` query vagy header paraméterben megadott id öçB 7F à

GET /{dialect}/api/where/trip-details

Request Parameters:

vehicleId: string; // Amennyiben meg van adva, azon menet érkezik a válaszban, amit az adott járm teljesít.
tripId: string; // Ha nincs kitöltve a járm onosító, az itt megadott azonosítóval rendelkez ÖVæWB ÆW7¢ a válaszban.
date: string; // Ha nincs kitöltve a járm onosító, ezen a dátumon lesz keresve az adott azonosítójú

menet.
ifModifiedSince: integer; // Akkor adunk vissza adatot, ha az módosult az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ellenkez W6WF&Vâ …EE 3 B `álasz, ha nincs változás. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja a részletes információit egy menetnek, ami az azonosítója és a dátum alapján, vagy az azt teljesít ® m onosítója alapján lett lekérve.

Content: application/json | TripDetailsOTPMethodResponse

```
{
  currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
  version: integer; // A válasz API verziója.
  status: Status;
  code: integer; // A válasz státusz kódja.
  text: string; // A válasz szövege.
  data: TransitEntryWithReferencesTransitTripDetailsOTP;
}
```

Response 304:

Nem változott az adat az `ifModifiedSince` paraméterben megadott id öçB 7F à

Response 400:

Nem sikerült beolvasni a dátumot.

Response 404:

A következ W6WFV² Vw—°e: -Ismeretlen járm onosító. -Ismereten menetazonosító. -A kért menet nem közlekedik az adott napon.

GET /{dialect}/api/where/vehicle-for-trip

Request Parameters:

tripId: Array<string>; // A lekért menetek azonosítói.
date: Array<string>; // A lekért menetrendi napok. Alapértelmezetten az aktuális nap.
ifModifiedSince: integer; // Akkor adunk vissza adatot, ha az módosult az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ellenkez W6WF&Vâ …EE 3 B `álasz, ha nincs változás. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route

referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszatér a megadott menetekhez és menetrendi napokhoz tartozó járm `ekkel.

Content: application/json | VehicleForTripMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitListEntryWithReferencesTransitVehicle;
}
```

Response 304:

Nem változott az adat az `ifModifiedSince` paraméterben megadott id   öçB 7F à

Response 400:

A menet azonosítókat és menetrendi napokat tartalmazó tömbök mérete nem egyezik.

Response 404:

Az egyik menet azonosítóhoz nem található menet.

GET /{dialect}/api/where/vehicles-for-location

Request Parameters:

query: string; // A válasz listát sz !Q feltétel, amit tartalmaznia kell a járm onosítójának, rendszámának vagy a típusának.
lat: number; // A terület középpontjának szélességi koordinátája.
lon: number; // A terület középpontjának hosszúsági koordinátája.
latSpan: number; // A lekérési terület szélességi íve. (Terület szélessége: `lat +/- latspan`).
lonSpan: number; // A lekérési terület hosszúsági íve. (Terület hosszúsága: `lon +/- lonSpan`).
radius: integer; // Ha a `latSpan` vagy a `longSpan` nincs kitöltve, a keresési terület a középpontól számított `radius` méter távolság mind a négy irányban.
ifModifiedSince: integer; // Akkor adunk vissza adatot, ha az módosult az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ellenkez W6WF&Vâ …EE 3 B `álasz, ha nincs változás. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszatér az adott területen található összes járm `el. Ha a `lat` vagy a `lon` null, az összes járm benne lesz a válaszban.

Content: application/json | VehiclesForLocationMethodResponse

{
currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
version: integer; // A válasz API verziója.
status: Status;
code: integer; // A válasz státusz kódja.
text: string; // A válasz szövege.
data: TransitListEntryWithReferencesTransitVehicle;
}

Response 304:

Nem változott az adat az `ifModifiedSince` paraméterben megadott id öçB 7F à

GET /{dialect}/api/where/vehicles-for-route

Request Parameters:

routeId: Array<string>; // A lekért járatok azonosítói.
related: boolean; // Tartalmazza-e a válasz a kapcsolódó járatokat is a referenciákban.
ifModifiedSince: integer; // Akkor adunk vissza adatot, ha az módosult az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ellenkez W6WF&Vâ …EE 3 B `álasz, ha nincs változás. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

A megadott járat azonosítókhoz tartozó járm `ekkel tér vissza.

Content: application/json | VehiclesForRouteMethodResponse

{
currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
version: integer; // A válasz API verziója.
status: Status;
code: integer; // A válasz státusz kódja.
text: string; // A válasz szövege.
data: TransitListEntryWithReferencesTransitVehicle;
}

Response 304:

Nem változott az adat az `ifModifiedSince` paraméterben megadott id öçB 7F à

Response 404:

A járat azonosítóhoz nem található járat.

GET /{dialect}/api/where/vehicles-for-stop

Request Parameters:

stopId: string; // A lekért megálló azonosítója.
ifModifiedSince: integer; // Akkor adunk vissza adatot, ha az módosult az itt megadott id .–Ç•eg (UNIX id .–Ç•eg másodpercben) után. Ellenkez W6WF&Vâ …EE 3 B `álasz, ha nincs változás. Els bsége van a header paraméterrel szemben.
If-Modified-Since: string; // Ha meg van adva, akkor csak abban az esetben érkezik vissza adat, ha az változott a megadott id öçB 7F à Különben HTTP 304 a válasz. A query paraméternek els bsége van ezzel szemben.
appVersion: string; // A kliensalkalmazás verziója.
version: ApiVersion; // Az API verziója.
dialect: Dialect; // Az API referenciáinak dialektusa.
includeReferences: Array<ReferencesSchema>; // A referenciák típusát határozza meg. `true` vagy `COMPACT` esetén minden referencia szerepel, `false` esetén üres. `COMPACT` módban a route referenciák `description` mez ¦R ¶–† w•ásra kerül.

Response 200:

Visszaadja az összes járm `et, ami olyan menetet teljesít, ami tartalmazza a megadott megállót.

Content: application/json | VehiclesForStopMethodResponse

```
{
    currentTime: integer; // Az aktuális szerverid W§&VFÞ 6öG W&6&Vâà
    version: integer; // A válasz API verziója.
    status: Status;
    code: integer; // A válasz státusz kódja.
    text: string; // A válasz szövege.
    data: TransitListEntryWithReferencesTransitVehicle;
}
```

Response 304:

Nem változott az adat az `ifModifiedSince` paraméterben megadott id öçB 7F à

Response 404:

A megálló azonosítóhoz nem található megálló.

Schemas

ApiVersion

string
Values: 2, 3, 4

Dialect

string
Values: otp, mobile

ReferencesSchema

string
Values: true, false, compact, agencies, routes, trips, stops, alerts, stations

AlertSearchMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitSearch;
}
```

EffectType

string
Values: NO_SERVICE, WARNING

MobileTransitReferences

```
{
    agencies: Array<TransitAgency>;  // Szolgáltatók referenciáinak listája.
    routes: Array<TransitRoute>;  // Járatok referenciáinak listája.
    stops: Array<TransitStop>;  // Megállók referenciáinak listája.
    trips: Array<TransitTrip>;  // Menetek referenciáinak listája.
    alerts: Array<TransitAlert>;  // Zavarok referenciáinak listája.
}
```

OTPTransitReferences

```
{
    agencies: object;  // Szolgáltatók referenciáinak listája.
    routes: object;  // Járatok referenciáinak listája.
    stops: object;  // Megállók referenciáinak listája.
```

```
    trips: object;  // Menetek referenciáinak listája.
    alerts: object;  // Zavarok referenciáinak listája.
}
```

Status

string
Values: NOT_MODIFIED, OK, UNKNOWN_ERROR, NOT_FOUND, INVALID_VALUE, NOT_OPERATING, OUTSIDE_BOUNDS, PATH_NOT_FOUND, NO_TRANSIT_TIMES, REQUEST_TIMEOUT, BOGUS_PARAMETER, TOO_CLOSE, LOCATION_NOT_ACCESSIBLE, MISSING_MODE, ERROR_NO_GRAPH, PLANNER_SERVICE_UNAVAILABLE, ERROR_VEHICLE_LOCATION_SERVICE, ERROR_BIKE_RENTAL_SERVICE, ERROR_TICKETING_SERVICE, ERROR_TRANSIT_INDEX_SERVICE, MOVED_TEMPORARILY

TransitAgency

```
{
    id: string;  // A szolgáltató azonosítója.
    name: string;  // A szolgáltató neve.
    url: string;  // A szolgáltató linkje.
    timezone: string;  // A szolgáltató id   ónája.
    lang: string;  // A szolgáltató nyelve.
    phone: string;  // A szolgáltató telefonszáma.
}
```

TransitAlert

```
{
    id: string;  // A zavar azonosítója.
    start: integer;  // A zavar kezd   –AQpontja másodpercben.
    end: integer;  // A zavar végének id   öçF¦  Þ 6öG W&6&Vâà
    timestamp: integer;  // A zavar id  .–Ç•ege másodpercben.
    modifiedTime: integer;  // A zavar utolsó módosításának id   öçF¦  Þ 6öG W&6&Vâà
    stopIds: Array<string>;  // A zavar által érintett megállók azonosítóinak listája.
    routeIds: Array<string>;  // A zavar által érintett járatok azonosítóinak listája.
    url: TranslatedString;
    header: TranslatedString;
    description: TranslatedString;
    disableApp: boolean;  // A zavar hatására használhatatlan lesz-e az alkalmazás.
    startText: TranslatedString;
    endText: TranslatedString;
    routes: Array<TransitAlertRoute>;  // A zavar által érintett járatok listája.
}
```

TransitAlertRoute

```
{
    routeId: string;  // A zavar által érintett járat azonosítója.
    stopIds: Array<string>;  // A zavar által érintett megállók azonosítóinak listája a járaton.
    header: TranslatedString;
    effectType: EffectType;
}
```

TransitEntryWithReferencesTransitSearch

{
  limitExceeded: boolean; // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és érkezési id N× W?¢ ÆV°éréseknél használjuk.
  entry: TransitSearch;
  references: TransitReferences;
  class: string; // Az adat típusa. Egy entitás esetén "entryWithReferences".
}

TransitReferences

One of:

OTPTransitReferences
{
  agencies: object; // Szolgáltatók referenciáinak listája.
  routes: object; // Járatok referenciáinak listája.
  stops: object; // Megállók referenciáinak listája.
  trips: object; // Menetek referenciáinak listája.
  alerts: object; // Zavarok referenciáinak listája.
}

or

MobileTransitReferences
{
  agencies: Array<TransitAgency>; // Szolgáltatók referenciáinak listája.
  routes: Array<TransitRoute>; // Járatok referenciáinak listája.
  stops: Array<TransitStop>; // Megállók referenciáinak listája.
  trips: Array<TransitTrip>; // Menetek referenciáinak listája.
  alerts: Array<TransitAlert>; // Zavarok referenciáinak listája.
}

TransitRoute

{
  id: string; // A járat azonosítója.
  shortName: string; // A járat rövid neve.
  longName: string; // A járat hosszú neve.
  description: string; // A járat neve. Ha egy | szerepel a névben, akkor a végállomásokat választja el amelyek külön sorokban megjeleníthet V²à COMPACT referenciák kérése esetében nincs kitöltve.
  type: string; // A járat típusa.
  url: string; // A járat linkje.
  color: string; // A járat színe. Deprecated: használjuk a `style` attribútumot helyette.
  textColor: string; // A járat szövegének színe. Deprecated: használjuk a `style` attribútumot helyette.
  agencyId: string; // A járatot üzemeltet 7 olgáltató azonosítója.
  iconDisplayType: string; // A járat ikonjának megjelenítési típusa. Deprecated: használjuk a `style` attribútumot helyette.
  iconDisplayText: string; // A járat ikonjának megjelenítési szövege. Deprecated: használjuk a `style` attribútumot helyette.
  bikesAllowed: boolean; // Kerékpár szállítása engedélyezett-e a járaton.
  style: TransitRouteStyle;
  sortOrder: integer; // A járat rendezési sorszáma a többi járathoz viszonyítva.
}

TransitRouteStyle

```
{
    color: string;  // A járat színe.
    stop: TransitStopStyle;
    icon: TransitRouteStyleIcon;
    vehicleIcon: TransitVehicleStyleIcon;
}
```

TransitRouteStyleIcon

```
{
    type: string;  // Az ikon típusa.
    text: string;  // Az ikon felirata.
    textColor: string;  // Az ikon feliratának színe.
}
```

TransitSearch

```
{
    query: string;  // A keresett kifejezés.
    stopIds: Array<string>;  // Az illeszked   ÖV~ ÆÏ6²   onosítói.
    routeIds: Array<string>;  // Az illeszked   ®  atok azonosítói.
    alertIds: Array<string>;  // Az illeszked   ¦ varok azonosítói.
}
```

TransitStop

```
{
    id: string;  // A megálló azonosítója.
    vertex: string;  // A megálló utazástervez     onosítója, amelyet a `fromPlace` és a `toPlace`
        megadásához lehet használni.
    lat: number;  // A megálló szélességi koordinátája.
    lon: number;  // A megálló hosszúsági koordinátája.
    name: string;  // A megálló neve.
    code: string;  // A megálló kódja.
    direction: string;  // A megálló iránya.
    platformCode: string;  // A megálló peron-kódja.
    description: string;  // A megálló leírása.
    locationType: integer;  // A megálló hely-típusa.
    locationSubType: string;  // A megálló hely-altípusa.
    parentStationId: string;  // A szül  ÖÖV~ ÆÏ2   onosítója.
    type: string;  // A megálló típusa. Deprecated: használjuk a `style` attribútumot.
    wheelchairBoarding: boolean;  // Akadálymentesített-e a megálló.
    routeIds: Array<string>;  // A megállót érint   ®  atok azonosítói.
    stopColorType: string;  // A megálló színezésének típusa. Deprecated: használjuk a `style` attribútumot.
    alertIds: Array<string>;  // A megállóra vonatkozó aktív zavarok azonosítói.
    style: TransitStopStyle;
}
```

TransitStopStyle

```
{
```

```
    colors: Array<string>;  // A megálló stílushoz tartozó színek tömbje.
    type: string;  // A megálló stílus típusa.
    image: string;  // A megálló stílus képe.
}
```

TransitTrip

```
{
    id: string;  // A menet azonosítója.
    routeId: string;  // A menethez tartozó járat azonosítója.
    shapeId: string;  // A menet útvonalvezetésének azonosítója.
    blockId: string;  // A menethez tartozó GTFS block_id, amennyiben létezik.
    tripHeadsign: string;  // A menet célállomása.
    tripShortName: string;  // A menet rövid neve.
    serviceId: string;  // A menet közlekedési naptárának azonosítója.
    directionId: string;  // A menet irányának azonosítója.
    bikesAllowed: boolean;  // Szállítható-e kerékpár a meneten.
    wheelchairAccessible: boolean;  // Alacsonypadlós-e a menet.
}
```

TransitVehicleStyleIcon

```
{
    name: string;  // Az ikon neve.
}
```

TranslatedString

```
{
    translations: object;  // Lokalizáció és a hozzá tartozó fordítások összerendelése.
    someTranslation: string;  // A fordítás-összerendelések els   VÆVÖP.
}
```

ArrivalsAndDeparturesForLocationOTPMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status:
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitDepartureGroup:
}
```

TransitDepartureGroup

```
{
    routeId: string;  // A csoporthoz tartozó járat azonosító.
    headsign: string;  // A csoporthoz tartozó célállomás.
    stopTimes: Array<TransitScheduleStopTime>:  // A csoport menetrendi bejegyzései.
}
```

TransitListEntryWithReferencesTransitDepartureGroup

```
{
  list: Array<TransitDepartureGroup>;  // A lekért adatok listája.
  outOfRange: boolean;  // Az értéke mindig `false`.
  limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
    érkezési id  N× W?¢ ÆV°éréseknél használjuk.
  references: TransitReferences;
  class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

TransitScheduleStopTime

```
{
  stopId: string;  // A megálló azonosítója.
  stopHeadsign: string;  // A megállóban kijelzett célállomás.
  arrivalTime: integer;  // A megállóba érkezés tervezett ideje epoch másodbercben. Az els  ÖV~ ÆÏ6î Â
    hiányzik.
  departureTime: integer;  // A megállóból való indulás tervezett ideje epoch másodbercben. Az utolsó
    megállónál hiányzik.
  predictedArrivalTime: integer;  // A megállóba érkezés becsült ideje epoch másodbercben, ha a
    járathoz van valós idej  F Bà Az els  ÖV~ ÆÏ6î Â †ž àyzik.
  predictedDepartureTime: integer;  // A megállóból való becsült indulás ideje epoch másodbercben, ha a
    járathoz van valós idej  F Bà Az utolsó megállónál hiányzik.
  uncertain: boolean;  // Igaz, ha a menethez tartozó valós idej  F Fö² &— onytalanok.
  tripId: string;  // A bejegyzést tartalmazó menet azonosítója.
  serviceDate: string;  // A bejegyzéshez tartozó menet menetrendi napja.
  wheelchairAccessible: boolean;  // A bejegyzéshez tartozó menet alacsonypadlóssága.
  mayRequireBooking: boolean;  // Igaz, ha a menet legalább egy rákövetkez  ÖV~ ÆÏ6¦ `oglalást igényel.
  groupIds: Array<string>;  // A megálló csoportjának azonosítója.
  alertIds: Array<string>;  // A megállóhoz tartozó aktív zavarok azonosítóinak listája.
}
```

ArrivalsAndDeparturesForStopOTPMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitEntryWithReferencesTransitArrivalsAndDepartures;
}
```

TransitArrivalsAndDepartures

```
{
  stopId: string;  // A megálló azonosítója.
  routeIds: Array<string>;  // A megállóhoz tartozó járatok azonosítói.
  alertIds: Array<string>;  // A megállóhoz tartozó aktív zavarok.
  nearbyStopIds: Array<string>;  // A közeli megállók azonosítói.
  stopTimes: Array<TransitScheduleStopTime>;  // A megállóhoz tartozó menetrendi bejegyzések.
}
```

TransitEntryWithReferencesTransitArrivalsAndDepartures

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: TransitArrivalsAndDepartures;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

BicycleRentalMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitBikeRentalStation;
}
```

TransitBikeRentalStation

```
{
    id: string;  // Az állomás azonosítója.
    lat: number;  // Az állomás szélességi koordinátája.
    lon: number;  // Az állomás hosszúsági koordinátája.
    name: string;  // Az állomás neve.
    code: string;  // Az állomás kódja.
    type: string;  // Az állomás típusa.
    bikes: integer;  // Az állomáson elérhet   °erékpárok száma.
    spaces: integer;  // Az állomás összes helyének száma. Ez mindig fix 999.
}
```

TransitListEntryWithReferencesTransitBikeRentalStation

```
{
    list: Array<TransitBikeRentalStation>;  // A lekért adatok listája.
    outOfRange: boolean;  // Az értéke mindig `false`.
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    references: TransitReferences;
    class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

MetadataResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
```

```
    data: TransitEntryWithReferencesTransitMetadata;
}
```

TransitEntryWithReferencesTransitMetadata

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: TransitMetadata;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

TransitMetadata

```
{
    time: integer;  // Aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    timeZone: string;  // A szerver tranzit adatainak id   ónája.
    readableTime: string;  // A szerver ideje ISO8601 formátumban.
    validityStart: string;  // A szerveren található adatok érvényességének kezdete.
    validityEnd: string;  // A szerveren található adatok érvényességének vége.
    completeValidityStart: string;  // Azon id  7¦ ² °ezdete, amelyre az összes szolgáltató biztosít adatot.
        Több szolgáltató esetén kés   bi lehet, mint a `validityStart`.
    completeValidityEnd: string;  // Azon id  7¦ ² `ége, amelyre az összes szolgáltató biztosít adatot. Több
        szolgáltató esetén korábbi lehet, mint a `validityEnd`.
    lowerLeftLatitude: number;  // A betöltött adatok területének befoglaló téglalapjának két sarokpontja.
    lowerLeftLongitude: number;  // A betöltött adatok területének befoglaló téglalapjának két sarokpontja.
    upperRightLatitude: number;  // A betöltött adatok területének befoglaló téglalapjának két sarokpontja.
    upperRightLongitude: number;  // A betöltött adatok területének befoglaló téglalapjának két sarokpontja.
    boundingPolyLine: string;  // A betöltött adatokat határoló sokszög vonala (polyline).
    alertIds: Array<string>;  // Az egész alkalmazásra vonatkozó aktív zavarok azonosítói.
    feedIds: Array<string>;  // Az aktív adatforrások azonosítói.
    dayTypes: object;  // A menetrendi napok, és a hozzá tartozó napok típusát tartalmazó összerendelés
        egy hétre (az aktuális nap, és a következ   b'à
}
```

MultiRouteDetailsMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitRouteDetails;
}
```

TransitListEntryWithReferencesTransitRouteDetails

```
{
    list: Array<TransitRouteDetails>;  // A lekért adatok listája.
    outOfRange: boolean;  // Az értéke mindig `false`.
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
```

```
    érkezési id   N× W?¢ ÆV°éréseknél használjuk.
  references: TransitReferences;
  class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

TransitPolyline

```
{
  levels: string;  // A minta magassági adatai (mindig üres).
  points: string;  // A minta kódolt pontjai.
  length: integer;  // A minta pontjainak száma.
}
```

TransitRouteDetails

```
{
  id: string;  // A járat azonosítója.
  shortName: string;  // A járat rövid neve.
  longName: string;  // A járat hosszú neve.
  description: string;  // A járat neve. Ha egy | szerepel a névben, akkor a végállomásokat választja el
    amelyek külön sorokban megjeleníthet  V²à COMPACT referenciák kérése esetében nincs kitöltve.
  type: string;  // A járat típusa.
  url: string;  // A járat linkje.
  color: string;  // A járat színe. Deprecated: használjuk a `style` attribútumot helyette.
  textColor: string;  // A járat szövegének színe. Deprecated: használjuk a `style` attribútumot helyette.
  agencyId: string;  // A járatot üzemeltet   7 olgáltató azonosítója.
  iconDisplayType: string;  // A járat ikonjának megjelenítési típusa. Deprecated: használjuk a `style`
    attribútumot helyette.
  iconDisplayText: string;  // A járat ikonjának megjelenítési szövege. Deprecated: használjuk a `style`
    attribútumot helyette.
  bikesAllowed: boolean;  // Kerékpár szállítása engedélyezett-e a járaton.
  style: TransitRouteStyle;
  sortOrder: integer;  // A járat rendezési sorszáma a többi járathoz viszonyítva.
  variants: Array<TransitRouteVariant>;  // A járathoz tartozó járat variánsok listája.
  alertIds: Array<string>;  // A járaton található aktív zavarok azonosítóinak listája.
}
```

TransitRouteVariant

```
{
  name: string;  // A járat variáns neve.
  stopIds: Array<string>;  // A járat variáns megállóinak azonosítói.
  mayRequireBooking: boolean;  // Igaz, ha a járaton vannak olyan menetek, amelyeken szükséges lehet
    az el  &R `oglalás.
  bookableStopIds: Array<string>;  // A járat variáns megálló azonosítói, amelyekre lehetséges lehet a
    foglalás.
  direction: string;  // A járat variáns iránya.
  headsign: string;  // A járat variáns célállomása.
  polyline: TransitPolyline;
  routeId: string;  // A járat variánshoz tartozó járat azonosítója. Akkor van kitöltve, ha különbözik az
    eredetit  Âà
  type: string;  // A járat variáns típusa.
}
```

OnboardDepartSearchMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status;
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitListEntryWithReferencesTransitVehicle;
}
```

TransitCoordinatePoint

```
{
  lat: number;  // Szélességi koordináta.
  lon: number;  // Hosszúsági koordináta.
}
```

TransitListEntryWithReferencesTransitVehicle

```
{
  list: Array<TransitVehicle>;  // A lekért adatok listája.
  outOfRange: boolean;  // Az értéke mindig `false`.
  limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
    érkezési id   N× W?¢ ÆV°éréseknél használjuk.
  references: TransitReferences;
  class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

TransitVehicle

```
{
  vehicleId: string;  // A járm     onosítója.
  stopId: string;  // A járm  •g¢ F  tozó megálló azonosítója.
  stopSequence: integer;  // A járm    ÇF Ã — intett aktuális megálló sorszáma a meneten.
  routeId: string;  // A menethez tartozó járat azonosítója, amit a járm   FVÆ¦W>×Bà
  bearing: number;  // A járm   — ányszöge.
  location: TransitCoordinatePoint;
  serviceDate: string;  // A járm    ÇF Â FVÆ¦W>×FWGB ÖVæWB ÖVæWG&VæF’ æ  ¦ à
  licensePlate: string;  // A járm   &VæG7® Ö à
  label: string;  // A járm   ¶–¦VÇ¡Qjén megjelen   >–Î ÆÆöÞ 6°ép.
  model: string;  // A járm   N× W6 à
  deviated: boolean;  // Igaz, ha a járm   ÆWN— t az útvonaláról.
  stale: boolean;  // Igaz, ha a járm    ÷®Ö6Ÿ6¦  ¶Ÿg&VvVFWGBà
  lastUpdateTime: integer;  // A járm  •g¢ F  tozó utolsó valós idej    F B –AQbélyege másodpercben.
  status: string;  // A járm    ÆÆ  ÷F à
  congestionLevel: string;  // A járm  •g¢ F  tozó torlódási állapot.
  vehicleRouteType: string;  // A járm   `ajtája. Deprecated: használjuk a `style` attribútumot.
  stopDistancePercent: integer;  // Hol tart a járm     °ét megálló közti szakaszon, százalékban.
  wheelchairAccessible: boolean;  // Igaz, ha a járm    Æ 76öàypadlós.
  occupancy: TransitVehicleOccupancy;
  capacity: TransitVehicleOccupancy;
  tripId: string;  // A járm  •g¢ F  tozó menet azonosítója.
  vertex: string;  // A menet utazástervez     onosítója, amelyet a `fromPlace` megadásához lehet
```

használni.
  style: TransitVehicleStyle;
}

TransitVehicleOccupancy

{
  adults: integer; // Hány feln GB `an a járm  `ön.
  children: integer; // Hány gyermek van a járm `ön.
  strollers: integer; // Hány babakocsi van a járm `ön.
  wheelchairs: integer; // Hány kerekesszék van a járm `ön.
  other: integer; // Hány be nem kategorizált entitás van a járm `ön.
}

TransitVehicleStyle

{
  icon: TransitVehicleStyleIcon;
}

OnboardDepartPosition

{
  lat: number; // A szélességi koordinátája egy pozició pontnak.
  lon: number; // A hosszúsági koordinátája egy pozició pontnak.
  timestamp: integer; // Az id öçF¦ Vw' ÷¦–6Ÿ2 öçFæ ² W ö6, Þ 6öG W&6&Vâà
  accuracy?: number; // A pontossága egy pozició pontnak méterben.
  speed?: number; // A sebesség egy pozició pontban m/s-ban.
}

TraverseMode

string
Values: WALK, BICYCLE, CAR, TRAM, SUBWAY, RAIL, BUS, FERRY, CABLE_CAR, GONDOLA, FUNICULAR, TRANSIT, AIRPLANE, TROLLEYBUS, MONORAIL, SUBURBAN_RAILWAY, COACH

WalkProfile

string
Values: SLOW, MID, FAST

ApiTripSearchMetadata

{
  searchWindowUsed: integer;
  nextDateTime: integer;
  prevDateTime: integer;
}

BikeStreetCategory

string
Values: CYCLEWAY, CYCLELANE, LOW_TRAFFIC, OTHER, PEDESTRIAN

DisplayedLeg

```
{
    first: boolean;  // Jelzi, ha a láb az els    ¢ §F—FW vben.
    last: boolean;  // Jelzi, ha a láb az utolsó az útitervben.
    time: integer;  // Az utolsó lábon az érkezési id  Â  ¢ g77 es többin az indulási.
    walkTo: boolean;  // Jelzi, hogy a láb nem tranzit típusú.
    name: string;  // Az indulási hely neve, kivéve az utolsón, ahol a érkezési helyé.
}
```

ElevationPoint

```
{
    distance: number;  // A magassági pont távolsága a láb kezdetéhez képest méterben.
    elevation: number;  // A pont magassága.
}
```

EncodedPolylineBean

```
{
    points: string;  // A geometria (polyline) pontjai kódolva.
    length: integer;  // A geometria (polyline) pontjainak száma.
}
```

Itinerary

```
{
    duration: integer;  // Az útvonal hossza másodpercben.
    startTime: string;  // Az indulási id   öçB W§&VFÞ 6öG W&6&Vâà
    endTime: string;  // Az érkezési id   öçB W§&VFÞ 6öG W&6&Vâà
    walkTime: integer;  // Sétálási id    ¢ §G`onalon másodpercben.
    bikeTime: integer;  // Kerékpározási id    ¢ §G`onalon másodpercben.
    transitTime: integer;  // Járm  `ön töltött id    ¢ §G`onalon másodpercben.
    waitingTime: integer;  // Várakozási id   ®  m  &R  ¢ §G`onalon másodpercben.
    bikeDistance: number;  // Kerékpározási távolság az útvonalon méterben.
    walkDistance: number;  // Sétálási távolság az útvonalon méterben.
    walkLimitExceeded: boolean;  // Jelzi, hogy a sétálási limit túl lett lépve az útitervben.
    notAllTicketsAvailable: boolean;  // Jelzi, hogy a visszaadott útvonalak tartalmazhatnak olyan járatokat
        (pl.: vonatok), amire a terv során érintett jegyértékesítési helyen nem lehet jegyet váltani.
    bikeCategoryDistances: object;  // Az útiterv során hány méter kerékpározást tartalmaztak az egyes
        kategóriák. Az objektum kulcsai a kategóriák (`BikeStreetCategory`).
    elevationLost: number;  // Az útiterv során hány métert süllyed az útvonal.
    elevationGained: number;  // Az útiterv során hány métert emelkedik az útvonal.
    transfers: integer;  // Átszállások száma az útitervben.
    generalizedCost: integer;  // Az utazástervez    Æv÷ itmus által használt súly.
    waitTimeAdjustedGeneralizedCost: integer;  // A várakozási id  `el igazított súly.
    legs: Array<Leg>;  // Az útiterv lábai (részei).
    displayedLegs: Array<DisplayedLeg>;  // Az útiterv kivonatos megjelenítéséhez használható  lábak és
```

```
    adatok.
  tooSloped: boolean;  // Jelzi, ha az útvonal meredekebb, mint a paraméterekben kért.
  patternItineraries: Array<Itinerary>;  // Útitervminták engedélyezése esetén a mintába tartozó útitervek.
  patternFrequency: PatternStatistics;
  patternDuration: PatternStatistics;
  displayProductRecommendation: boolean;  // Jelzi, ha az útvonalhoz kell termékajánlást mutatni.
}
```

Leg

```
{
  startTime: string;  // A láb kezdete ezredmásodpercben.
  endTime: string;  // A láb vége ezredmásodpercben.
  departureDelay: integer;  // Tranzit láb esetén a láb indulása és a tényleges felszállás közötti késleltetés.
  arrivalDelay: integer;  // Tranzit láb esetén a láb vége és a tényleges leszállás közötti késleltetés.
  realTime: boolean;  // Jelzi, hogy a láb adatai valós idej  `orrásból származnak.
  distance: number;  // A láb bejárása alatt megtett táv méterben.
  pathway: boolean;  // Jelzi, hogy a láb egy pathway (pl. metró lejáró).
  mode: string;  // A lábon használt közlekedési mód.
  agencyName: string;  // Tranzit láb esetén a járatot biztosító szolgáltató neve.
  agencyUrl: string;  // Tranzit láb esetén a járatot biztosító szolgáltató linkje.
  agencyTimeZoneOffset: integer;  // Tranzit láb esetén a járatot biztosító szolgáltató id  ónájának
    eltolódása.
  routeColor: string;  // Tranzit láb esetén a járat hátterének színe a megjelenítéséhez.
  routeId: string;  // Tranzit láb esetén a járat azonosítója.
  routeTextColor: string;  // Tranzit láb esetén a járat szövegének színe a megjelenítéséhez.
  interlineWithPreviousLeg: boolean;  // Jelzi, hogy az utas maradjon a járm  `ön a végállomásnál, mert
    másik menetként folytatódik (hurokjárat).
  tripBlockId: string;  // Tranzit láb esetén a menethez tartozó GTFS block_id, amennyiben létezik.
  headsign: string;  // Tranzit láb esetén a menet célállomása.
  agencyId: string;  // Tranzit láb esetén a járatot biztosító szolgáltató azonosítója.
  tripId: string;  // Tranzit láb esetén a menet azonosítója.
  serviceDate: string;  // Tranzit láb esetén a menet menetrendi napja.
  from: Place;
  to: Place;
  legGeometry: EncodedPolylineBean;
  legElevation: Array<ElevationPoint>;  // A lábhoz tartozó magassági adatok.
  alertIds: Array<string>;  // A lábhoz tartozó aktív zavarok azonosítói.
  routeShortName: string;  // Tranzit láb esetén a járat rövid neve.
  routeLongName: string;  // Tranzit láb esetén a járat hosszú neve.
  boardRule: string;  // Igényvezérelt menet esetén a felszállási szabály.
  alightRule: string;  // Igényvezérelt menet esetén a leszállási szabály.
  rentedBike: boolean;  // Jelzi, hogy a lábat bérelt kerékpárral kell bejárni.
  wait: string;  // A várakozási id  ÖVæàyiségi típusa a tranzit típusú lábon.
  routeIds: Array<string>;  // Útiterv minták esetén, a lábnak a mintában megfelel  Â Þ 2 Î & °on használt
    járatok azonosítóinak listája.
  tripIds: Array<string>;  // Útiterv minták esetén, a lábnak a mintában megfelel  Â Þ 2 Î & °on használt
    menetek azonosítóinak listája.
  hasAlertInPattern: boolean;  // Útiterv minták esetén jelzi, hogy a mintában a megfelel  Î & ²
    valamelyikén szerepel aktív riasztás. Nincs kitöltve, ha az útiterv minták ki vannak kapcsolva.
  generalizedCost: integer;  // Az utazástervez  Æv÷ itmus által használt súly.
  requiresBooking: boolean;  // Igaz, ha szükséges foglalás.
  onBoardAccess: boolean;  // Igaz, ha járatról történt a tervezés.
  vertex: string;  // A menet utazástervez  onosítója, amelyet a `fromPlace` megadásához lehet
    használni.
  duration: integer;
  timeZone: object;
  transitLeg: boolean;
```

```
    intermediateStops: Array<Place>;  // Tranzit láb esetén a köztes megállók a láb kezdete és végz  N—6R
        között.
    steps: Array<WalkStep>;  // Az útvonal lépései gyalogos, kerékpáros vagy autós láb lesetén.
}
```

PatternStatistics

```
{
    min: integer;  // Minimum érték. Akkor van kitöltve, ha az adatok különböz  V²à
    max: integer;  // Maximum érték. Akkor van kitöltve, ha az adatok különböz  V²à
    avg: integer;  // Átlag érték. Akkor van kitöltve, ha az adatok megegyeznek.
    text: string;  // A klienseken megjelenített szöveg.
}
```

Place

```
{
    name: string;  // Megálló esetén a megálló neve, POI-nál a POI neve.
    ticketingLocation: TicketingLocation;
    stopId: string;  // A megálló azonosítója, ha a hely egy megálló.
    stopCode: string;  // A megálló kódja, ha a hely egy megálló.
    platformCode: string;  // A megálló peron-kódja, ha a hely egy megálló.
    lon: number;  // A hely hosszúsági koordinátája.
    lat: number;  // A hely szélességi koordinátája.
    arrival: string;  // A helyre érkezés id   öçF¦  W§&VFÞ 6öG W&6&Vâà
    departure: string;  // A helyr  Â `aló indulás id   öçF¦  W§&VFÞ 6öG W&6&Vâà
    orig: string;  // Az indulási és az érkezési hely címkéje. Az els   —2  ¢ WFöÇ?2 §G`onal lábon van kitöltve.
    stopIndex: integer;  // A megálló indexe, ha a hely egy megálló.
    stopSequence: integer;  // A megálló sorszáma a meneten, ha a hely egy megálló.
}
```

PlanTripResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesResponse;
}
```

PlannerError

```
{
    id: integer;  // A hibaüzenet azonosítója.
    message: string;  // A hibaüzenet kódja.
    missing: Array<string>;  // A rossz paraméterek listája.
    noPath: boolean;  // Igaz, ha nem sikerült útvonalat találni.
    msgFromMessage: string;
}
```

Response

```
{
    requestParameters: object;  // A tervezési paraméterek.
    plan: TripPlan;
    metadata: ApiTripSearchMetadata;
    error: PlannerError;
}
```

TicketingLocation

```
{
    id: string;  // A jegyértékesítési hely azonosítója.
    type: string;  // A jegyértékesítési hely típusa.
    state: string;  // A jegyértékesítési hely állapota.
    visible: boolean;  // Ha `false`, akkor hibás az adat, így a felületen elrejtésre kerül.
    place: string;  // A jegyértékesítési hely helye.
    address: string;  // A jegyértékesítési hely címe.
    description: string;  // A jegyértékesítési hely leírása.
    operator: string;  // A jegyértékesítési hely üzemeltet ¦P.
    lat: number;  // A jegyértékesítési hely szélességi koordinátája.
    lon: number;  // A jegyértékesítési hely hosszúsági koordinátája.
    cashAccepted: boolean;  // Jelzi, hogy a jegyértékesítési helyen elfogadnak készpénzt.
    creditCardsAccepted: boolean;  // Jelzi, hogy a jegyértékesítési helyen elfogadnak bankkártyát.
    passIdCreation: boolean;  // Lehet-e a helyszínen igazolványt csináltatni.
    ticketPassExchange: boolean;  // Be lehet-e váltani a helyszínen jegyet.
    openingPeriods: Array<TicketingPeriod>;  // A jegyértékesítési hely nyitvatartása.
    products: Array<string>;  // Az árusított termékek azonosítói.
    lastModified: string;  // A jegyértékesítési hely utolsó frissítésének id   öçF¦  W§&VFÞ 6öG W&6&Vâà
}
```

TicketingPeriod

```
{
    dayOfWeek: string;  // A nyitvatartási id   æ  ¦ à Lehet HOL (ünnepnap) és 0247 (éjjel-nappali).
    opens: string;  // Nyitási id   „ƒ¦ÖÒ `ormátumban.
    closes: string;  // Zárási id   „ƒ¦ÖÒ `ormátumban.
    opensSeconds: integer;  // Nyitási id   Þ 6öG W&6&Vâà
    closesSeconds: integer;  // Zárási id   Þ 6öG W&6&Vâà
}
```

TransitEntryWithReferencesResponse

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: Response;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

TripPlan

```
{
    date: string;  // Az útvonal kezdete ezredmásodpercben.
    from: Place;
    to: Place;
    itineraries: Array<Itinerary>;  // A talált útvonalak listája.
}
```

WalkStep

```
{
    distance: number;  // A lépés hossza méterben.
    relativeDirection: string;  // A lépés iránya.
    streetName: string;  // Az utca neve, amin a lépés halad.
    absoluteDirection: string;  // A lépés abszolút iránya.
    stayOn: boolean;  // Jelzi, hogy az utca irányt vált egy keresztez  N—6&Vâà
    area: boolean;  // Jelzi, hogy a lépés egy nyílt terület, pl.: pláza, vagy vonat peron.
    bogusName: boolean;  // Jelzi, hogy az utca nevét a rendszer generálta.
    lon: number;  // A lépés kezdetének hosszúsági koordinátája.
    lat: number;  // A lépés kezdetének szélességi koordinátája.
    bicycleStreetDirection: string;  // A szakasz egyirányúsága.
    bicycleCategory: BikeStreetCategory;
    walkingBike: boolean;  // Jelzi, hogy a szakszon sétálni kell kerékpáros tervezés esetén.
    geometry: EncodedPolylineBean;
}
```

ReferencesMethodErrors

```
{
    agencyIds: Array<string>;  // A szolgáltató ID-k, amelyek feloldása sikertelen volt.
    alertIds: Array<string>;  // A zavar ID-k, amelyek feloldása sikertelen volt.
    routeIds: Array<string>;  // A járat ID-k, amelyek feloldása sikertelen volt.
    stopIds: Array<string>;  // A megálló ID-k, amelyek feloldása sikertelen volt.
}
```

ReferencesMethodResponse

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id  N× W?¢ ÆV°éréseknél használjuk.
    entry: ReferencesMethodResult;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

ReferencesMethodResult

```
{
    errors: ReferencesMethodErrors;
}
```

RouteDetailsForStopMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitRoute;
}
```

TransitListEntryWithReferencesTransitRoute

```
{
    list: Array<TransitRoute>;  // A lekért adatok listája.
    outOfRange: boolean;  // Az értéke mindig `false`.
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    references: TransitReferences;
    class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

RouteDetailsMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitRouteDetails;
}
```

TransitEntryWithReferencesTransitRouteDetails

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: TransitRouteDetails;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

ScheduleForStopOTPMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitSchedule;
}
```

TransitEntryWithReferencesTransitSchedule

```
{
  limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
     érkezési id   N× W?¢ ÆV°éréseknél használjuk.
  entry: TransitSchedule;
  references: TransitReferences;
  class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

TransitRouteSchedule

```
{
  routeId: string;  // A menetrendhez tarozó járat azonosítója.
  alertIds: Array<string>;  // A járathoz tartozó aktív zavarok.
  directions: Array<TransitRouteScheduleForDirection>;  // A menetrendi adatok irány szerint
     csoportosítva.
}
```

TransitRouteScheduleForDirection

```
{
  directionId: string;  // Az irány azonosítója.
  groups: object;  // Célmegálló és a hozzá tartozó menetrendi adatok összerendelése.
  stopTimes: Array<TransitScheduleStopTime>;  // Az irányhoz tartozó menetrendi bejegyzések.
}
```

TransitSchedule

```
{
  stopId: string;  // A menetrend megállójának azonosítója.
  serviceDate: string;  // A menetrendhez tartozó dátum.
  date: string;  // A menetrendhez tartozó dátum.
  routeIds: Array<string>;  // A menetrendhez tartozó járatok azonosítói.
  nearbyStopIds: Array<string>;  // A közeli megállók azonosítói.
  alertIds: Array<string>;  // A megállóhoz tartozó aktív zavarok azonosítói.
  schedules: Array<TransitRouteSchedule>;  // A megállóhoz tartozó menetrend.
}
```

TransitScheduleGroup

```
{
  groupId: string;  // A menetrendi csoport azonosítója.
  headsign: string;  // A menetrendi csoport célállomása.
  description: string;  // A menetrendi csoport leírása.
}
```

SearchMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
```

```
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitSearch;
}
```

StatisticsResponse

```
{
    result: string;  // Egyszer   `álasz.
}
```

StopsForLocationResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitStop;
}
```

TransitListEntryWithReferencesTransitStop

```
{
    list: Array<TransitStop>;  // A lekért adatok listája.
    outOfRange: boolean;  // Az értéke mindig `false`.
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    references: TransitReferences;
    class: string;  // Az adat típusa. Lista esetén "listWithReferences".
}
```

TicketingMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitEntryWithReferencesTransitTicketing;
}
```

TicketingProduct

```
{
    id: string;  // A termék azonosítója.
    groupId: string;  // A termék csoportjának azonosítója.
```

```
    groupName: string;  // A termék csoportjának neve.
    name: string;  // A termék neve.
    url: string;  // A termék linkje.
    price: string;  // A termék ára.
    visible: boolean;  // Ha `false`, akkor hibás az adat, így a felületen elrejtésre kerül.
    lastModified: string;  // A termék utolsó frissítésének id   öçF¦  W§&VB Þ 6öG W&6&Vâà
}
```

TransitEntryWithReferencesTransitTicketing

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: TransitTicketing;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

TransitTicketing

```
{
    lastModifiedTime: integer;  // A jegyértékesítési adatok legutóbbi módosulásának id   öçF¦ à
    oldestModifiedTime: integer;  // A legrégebb óta módosított értékesítési hely vagy termék
        módosításának id   öçF¦ à
    locations: Array<TicketingLocation>;  // A jegyértékesítési pontok listája.
    products: Array<TicketingProduct>;  // A termékek listája.
}
```

TransitEntryWithReferencesTransitTripDetailsOTP

```
{
    limitExceeded: boolean;  // Igaz, ha a lista több elemet tartalmaz, mint a limit paraméter. Indulási és
        érkezési id   N× W?¢ ÆV°éréseknél használjuk.
    entry: TransitTripDetailsOTP;
    references: TransitReferences;
    class: string;  // Az adat típusa. Egy entitás esetén "entryWithReferences".
}
```

TransitTripDetailsOTP

```
{
    tripId: string;  // A menet azonosítója.
    serviceDate: string;  // A menet menetrendi napja.
    vertex: string;  // A menet utazástervez     onosítója, amelyet a `fromPlace` megadásához lehet
        használni.
    vehicle: TransitVehicle;
    polyline: TransitPolyline;
    alertIds: Array<string>;  // Aktív zavarok a meneten.
    stopTimes: Array<TransitTripStopTime>;  // Menet megállóinak listája.
    nextBlockTripId: string;  // A következ   ÖVæWB   onosítója a csoportban, ha a menet hurokjárat.
    mayRequireBooking: boolean;  // Igaz, ha a menet (legalább egy szakasza) foglalást igényel.
}
```

TransitTripStopTime

```
{
  stopId: string;  // A megálló azonosítója.
  stopHeadsign: string;  // A megállóban kijelzett célállomás.
  arrivalTime: integer;  // A megállóba érkezés tervezett ideje epoch másodbercben. Az els  ÖV~ ÆÏ6î Â
    hiányzik.
  departureTime: integer;  // A megállóból való indulás tervezett ideje epoch másodbercben. Az utolsó
    megállónál hiányzik.
  predictedArrivalTime: integer;  // A megállóba érkezés becsült ideje epoch másodbercben, ha a
    járathoz van valós idej  F Bà Az els  ÖV~ ÆÏ6î Â †ž àyzik.
  predictedDepartureTime: integer;  // A megállóból való becsült indulás ideje epoch másodbercben, ha a
    járathoz van valós idej  F Bà Az utolsó megállónál hiányzik.
  uncertain: boolean;  // Igaz, ha a menethez tartozó valós idej  F Fö² &— onytalanok.
  requiresBooking: boolean;  // Igaz, ha ez a megálló foglalást igényel.
  stopSequence: integer;  // A megálló sorrendje a járaton.
  shapeDistTraveled: number;  // Milyen messze található a megálló az els  AQl a minta mentén méterben.
}
```

TripDetailsOTPMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status:
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitEntryWithReferencesTransitTripDetailsOTP:
}
```

VehicleForTripMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status:
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitListEntryWithReferencesTransitVehicle:
}
```

VehiclesForLocationMethodResponse

```
{
  currentTime: integer;  // Az aktuális szerverid  W§&VFÞ 6öG W&6&Vâà
  version: integer;  // A válasz API verziója.
  status: Status:
  code: integer;  // A válasz státusz kódja.
  text: string;  // A válasz szövege.
  data: TransitListEntryWithReferencesTransitVehicle:
}
```

VehiclesForRouteMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitVehicle;
}
```

VehiclesForStopMethodResponse

```
{
    currentTime: integer;  // Az aktuális szerverid   W§&VFÞ 6öG W&6&Vâà
    version: integer;  // A válasz API verziója.
    status: Status;
    code: integer;  // A válasz státusz kódja.
    text: string;  // A válasz szövege.
    data: TransitListEntryWithReferencesTransitVehicle;
}
```