

Guide API — Tesla Delivery Advisor

Version de la documentation : Serveur 26 — 8 septembre 2026

Serveur central : <https://b3-demo1.kevinpecro.info>

Base API : <https://b3-demo1.kevinpecro.info/api/v1>

Ce document décrit l'API centrale utilisée par les applications **Delivery Advisor (DA)**. Il correspond aux routes réellement exposées dans `api/index.php` de Serveur 26 et inclut les dernières fonctionnalités : aide à l'affectation, disponibilité client V2, planning, stock prévisionnel, transferts inter-agences, parking et mode automatique Bruxelles.

1. Principes généraux

Un Delivery Advisor correspond à **un seul Delivery Center**. Chaque centre possède sa propre clé API.

La clé Bearer identifie automatiquement le Delivery Center. Les applications DA ne doivent donc pas envoyer de `delivery_center_id` pour choisir l'agence sur les routes métier : le serveur central déduit le centre à partir de la clé.

Le serveur central reste la source de vérité pour :

- le catalogue Tesla : modèles, versions et options ;
- les usines et délais logistiques ;
- les véhicules, leurs statuts et leurs dates ;
- les commandes clients ;
- les affectations commande ↔ véhicule ;
- les réponses de disponibilité des clients ;
- les rendez-vous ;
- les incidents ;
- les livraisons ;
- le stock et sa projection ;
- les transferts entre Delivery Centers ;
- le plan de parking et l'occupation réelle des places ;
- le mode automatique du DA de Bruxelles.

Une application locale peut conserver une copie des données pour son affichage, mais elle ne doit jamais modifier directement la base centrale.

2. Authentification

Toutes les routes sauf `GET /ping` nécessitent :

```
Authorization: Bearer VOTRE_CLE_API
```

Exemple :

```
curl -H "Authorization: Bearer VOTRE_CLE_API" \  
https://b3-demo1.kevinpecro.info/api/v1/centre
```

Une clé n'est acceptée que si :

- elle correspond à un `delivery_center` ;
- `api_active = 1` ;
- le Delivery Center est actif.

La dernière activité API du centre est mise à jour à chaque authentification réussie.

Ne placez jamais la clé dans un dépôt Git public. Dans une application DA, stockez-la côté serveur ou dans une variable d'environnement.

3. Format des réponses et codes HTTP

Réponse correcte

```
{  
  "success": true,  
  "data": {},  
  "meta": {  
    "server_time": "2026-09-08 13:00:00",  
    "timezone": "Europe/Paris"  
  }  
}
```

Réponse d'erreur

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Ce véhicule est déjà affecté."
  }
}
```

Codes HTTP principaux

Code	Signification
200	Lecture ou action réussie
201	Ressource créée, par exemple incident ou demande de transfert
204	Réponse CORS OPTIONS
401	Clé absente, invalide ou désactivée
404	Route ou ressource introuvable
422	Règle métier non respectée ou données invalides
500	Erreur serveur ou module SQL requis non installé

Les dates métier sont interprétées dans le fuseau **Europe/Paris**.

Format conseillé :

YYYY-MM-DD HH:MM:SS

4. Vue d'ensemble des endpoints

Lecture

Méthode	Endpoint	Fonction
GET	/api/v1/ping	Tester le service sans authentification
GET	/api/v1/centre	Informations et capacité du centre
GET	/api/v1/horaires	Horaires du centre
GET	/api/v1/dashboard	KPI rapides
GET	/api/v1/stock-previsionnel	Stock, arrivages, sorties et projection de capacité
GET	/api/v1/catalogue	Catalogue modèles / versions / options
GET	/api/v1/usines	Usines et délais vers le centre

Méthode	Endpoint	Fonction
GET	/api/v1/vehicules	Véhicules du centre
GET	/api/v1/vehicules/{VIN}	Détail d'un véhicule
GET	/api/v1/vehicules-matching	Véhicules libres couvrant au moins une commande
GET	/api/v1/vehicules/{VIN}/commandes-compatibles	Clients/commandes compatibles avec un VIN
GET	/api/v1/vehicules/{VIN}/creneaux	Créneaux livrables encore disponibles
GET	/api/v1/commandes	Commandes du centre
GET	/api/v1/commandes/{id-ou-reference}	Détail d'une commande
GET	/api/v1/commandes/{id-ou-reference}/vehicules-compatibles	VIN compatibles avec une commande
GET	/api/v1/commandes/{id-ou-reference}/disponibilites-client	Historique des propositions client
GET	/api/v1/planning	Rendez-vous du Delivery Center
GET	/api/v1/parking	Plan et occupation du parking
GET	/api/v1/transferts	Espace de transferts inter-agences
GET	/api/v1/automatisation	État du mode automatique Bruxelles
GET	/api/v1/evenements	Événements métier du centre
GET	/api/v1/sync	Synchronisation incrémentale

Actions

Méthode	Endpoint	Fonction
POST	/api/v1/commandes/{id-ou-reference}/disponibilite-client	Demander au client s'il accepte un créneau
POST	/api/v1/commandes/{id-ou-reference}/affectation	Affecter un VIN
DELETE	/api/v1/commandes/{id-ou-reference}/affectation	Retirer l'affectation
POST	/api/v1/commandes/{id-ou-reference}/desaffectation	Variante sans méthode DELETE
POST	/api/v1/commandes/{id-ou-reference}/rendez-vous	Créer le rendez-vous accepté
POST	/api/v1/commandes/{id-ou-reference}/incident	Signaler un incident
POST	/api/v1/commandes/{id-ou-reference}/livraison	Clôturer une livraison
POST	/api/v1/transferts/demande	Demander un véhicule à un autre centre
POST	/api/v1/transferts/proposition	Proposer un véhicule à un autre centre
POST	/api/v1/transferts/{id}/accepter	Accepter un transfert reçu
POST	/api/v1/transferts/{id}/refuser	Refuser un transfert reçu
POST	/api/v1/automatisation/demarrer	Démarrer le DA automatique Bruxelles
POST	/api/v1/automatisation/pause	Mettre le DA automatique en pause
POST	/api/v1/automatisation/tick	Exécuter une décision automatique

5. Tester l'API

```
GET /api/v1/ping
```

Aucune clé n'est requise.

Exemple :

```
{
  "success": true,
  "data": {
    "service": "Tesla DA API",
    "version": "v1",
    "status": "ok"
  },
  "meta": {
    "server_time": "2026-09-08 13:00:00",
    "timezone": "Europe/Paris"
  }
}
```

GET /api/v1/ est également réécrit vers le ping par Apache.

6. Centre et horaires

6.1 Informations du centre

```
GET /api/v1/centre
```

Retourne notamment :

- id, code, nom, ville, pays ;
- capacite_stockage ;
- occupation_actuelle ;
- vehicules_disponibles ;
- places_restantes ;
- taux_occupation ;
- vehicules_en_transport ;
- vehicules_en_amont ;
- duree_creneau_minutes ;

- capacite_livraisons_simultanees ;
- horaires.

Règle actuelle commune aux Delivery Centers :

```
duree_creneau_minutes = 60
capacite_livraisons_simultanees = 5
```

Ces deux valeurs sont imposées par la logique centrale, même si une ancienne valeur différente existe encore en base.

6.2 Horaires

```
GET /api/v1/horaires
```

Chaque ligne contient :

```
{
  "jour_numero": 1,
  "jour": "lundi",
  "ouvert": true,
  "heure_debut": "09:00",
  "heure_fin": "18:00"
}
```

Si un centre ne possède **aucune** ligne d'horaires, l'API applique les horaires pédagogiques par défaut :

- lundi à vendredi : 09:00-18:00 ;
- samedi : 09:00-17:00 ;
- dimanche : fermé.

Dès qu'au moins un horaire est configuré en base, les jours absents sont considérés comme fermés.

7. Dashboard rapide

```
GET /api/v1/dashboard
```

La réponse contient deux groupes de compteurs.

commandes

- commandes_a_affecter ;
- vehicules_affectes ;

- commandes_pretes;
- livrees_aujourd'hui;
- rendez_vous_planifies.

vehicules

- disponibles;
- en_transport;
- en_amont;
- arrivees_aujourd'hui;
- arrivees_7_jours.

Cette route est adaptée à une page d'accueil ou à des tuiles KPI.

8. Stock et capacité prévisionnelle

```
GET /api/v1/stock-previsionnel?days=30
```

Paramètres

Paramètre Obligatoire

Valeur

days non horizon de 1 à 90 jours, défaut 30

La projection est calculée **par le serveur central**.

La réponse contient notamment :

- generated_at;
- horizon;
- centre;
- capacite_stockage;
- occupation_actuelle;
- places_restantes_actuelles;
- taux_occupation_actuel;
- arrivages_7_jours;
- pic_previsionnel;
- risque_depassement;
- premier_depassement;
- stock;
- arrivages;

- arrivages_sans_date;
- sorties_planifiees;
- projection.

Règles de calcul

Le stock physique correspond aux véhicules du centre en statut :

```
disponible, reserve, attribue
```

La projection :

- ajoute +1 lorsqu'un véhicule arrive ;
- retire -1 lorsqu'une livraison possède déjà un rendez-vous confirmé ;
- ne peut pas intégrer les arrivages dont la date est inconnue ;
- traite une arrivée avant une sortie si deux mouvements ont exactement le même horodatage.

Un véhicule affecté à un client reste donc compté dans le stock tant qu'il n'est pas réellement sorti.

9. Catalogue Tesla

```
GET /api/v1/catalogue
```

Retourne :

```
{
  "modeles": [],
  "versions": [],
  "types_options": []
}
```

Chaque version contient ses `options_compatibles`.

Utilisez cette route plutôt que de coder en dur les versions ou options dans l'application DA.

10. Usines et délais logistiques

```
GET /api/v1/usines
```

Pour chaque usine, l'API retourne notamment :

- id;
- code;
- nom;
- ville;
- pays;
- actif;
- delai_jours vers le Delivery Center authentifié ;
- modeles_produits.

11. Véhicules

11.1 Tous les véhicules du centre

GET /api/v1/vehicules

Un DA ne peut voir que les véhicules dont delivery_center_id correspond à sa clé API.

Paramètres

Paramètre	Exemple	Description
statut	transport,disponible	liste de statuts séparés par des virgules
updated_since	2026-09-08 10:00:00	véhicules modifiés depuis cette date
limit	100	défaut 500, maximum 1000
include_options	0	0 évite de charger les options

Statuts utilisés dans la simulation :

```
planifie
production
produit
sortie_usine
transport
disponible
reserve
attribue
livre
```

Chaque véhicule contient notamment :

- son vin ;
- modèle/version/transmission ;
- usine ;
- dates de production, expédition et arrivée ;

- statut ;
- options ;
- affecte_commande s'il est déjà utilisé.

11.2 Détail d'un VIN

```
GET /api/v1/vehicules/LRW26090200002007
```

Un VIN appartenant à un autre Delivery Center retourne 404.

12. Véhicules utiles au DA

12.1 Véhicules couvrant au moins une commande

```
GET /api/v1/vehicules-matching?limit=250
```

limit : défaut 250, maximum 1000.

Cette route retourne uniquement les véhicules :

- du centre authentifié ;
- non livrés ;
- non déjà affectés à une commande ;
- compatibles avec au moins une commande nouvelle ou a_affecter.

Les statuts considérés sont :

```
planifie, production, produit, sortie_usine, transport, disponible
```

Chaque ligne contient notamment :

- matching_orders_count ;
- premiere_commande_compatible ;
- date_disponibilite_centre ;
- les options du véhicule.

Le tri privilégie les véhicules déjà disponible, puis les dates d'arrivée/production les plus proches.

12.2 Commandes compatibles avec un VIN

```
GET /api/v1/vehicules/{VIN}/commandes-compatibles
```

Le central retourne uniquement les commandes :

- du même Delivery Center ;
- sans véhicule affecté ;
- en statut nouvelle ou a_affecter ;
- de la même version ;
- dont toutes les options demandées existent sur le VIN.

Les commandes sont classées de la plus ancienne à la plus récente.

13. Créneaux disponibles pour un véhicule

```
GET /api/v1/vehicules/{VIN}/creneaux?days=28
```

Paramètres :

Paramètre Obligatoire		Description
days	non	de 1 à 60, défaut 21
from	non	date minimale YYYY-MM-DD

L'API calcule directement les créneaux qui respectent :

1. le préavis minimum client ;
2. la disponibilité réelle/prévisionnelle du véhicule ;
3. le temps de réception et préparation ;
4. les horaires du Delivery Center ;
5. la grille fixe d'une heure ;
6. la capacité maximale de 5 clients par créneau ;
7. les rendez-vous déjà pris.

Règle de préparation du véhicule

Un véhicule déjà disponible peut être planifié normalement.

Pour un véhicule à venir, la livraison est possible à partir du **premier horaire d'ouverture du lendemain de son arrivée**.

Exemple :

Arrivée : mardi 8 septembre à 14:25
Ouverture suivante : mercredi 9 septembre à 09:00
Premier créneau livrable : mercredi 9 septembre à 09:00

Le préavis client est calculé séparément entre **maintenant** et le rendez-vous proposé. On ne fait jamais :

date_arrivee + 48 heures

Extrait de réponse :

```
{
  "vin": "XP7...",
  "date_arrivee_vehicule": "2026-09-08 14:25:00",
  "date_disponibilite_vehicule": "2026-09-09 09:00:00",
  "preavis_minimum_heures": 48,
  "duree_creneau_minutes": 60,
  "capacite_livraisons_simultanees": 5,
  "jours": [
    {
      "date": "2026-09-09",
      "jour_semaine": 3,
      "slots": [
        {
          "datetime": "2026-09-09 09:00:00",
          "heure": "09:00",
          "heure_fin": "10:00",
          "places_occupees": 1,
          "places_restantes": 4,
          "capacite": 5
        }
      ]
    }
  ]
}
```

Si la date d'arrivée est inconnue, `jours` est vide et un message explicatif est renvoyé.

14. Commandes clients

14.1 Lister les commandes

```
GET /api/v1/commandes
```

Paramètres :

Paramètre	Exemple	Description
statut	nouvelle,a_affecter	statuts séparés par des virgules
updated_since	2026-09-08 10:00:00	commandes modifiées depuis la date
limit	100	défaut 500, maximum 1000

Chaque commande contient notamment :

- id et reference ;
- statut ;
- date de commande ;
- client ;
- modèle/version ;
- options ;
- VIN éventuellement affecté ;
- état et arrivée prévue du véhicule ;
- rendez-vous ;
- livraison réelle le cas échéant.

14.2 Détail d'une commande

Par ID :

```
GET /api/v1/commandes/158
```

Par référence :

```
GET /api/v1/commandes/CMD-260902-120501-48672
```

La fiche détaillée contient aussi :

- evenements ;
- disponibilites_client.

14.3 VIN compatibles avec une commande

```
GET /api/v1/commandes/{id-ou-reference}/vehicules-compatibles
```

Le central filtre :

- le bon Delivery Center ;

- la bonne version ;
- toutes les options demandées ;
- les VIN non livrés ;
- les VIN non utilisés par une autre commande.

Le tri privilège :

1. disponible ;
2. transport ;
3. produit ;
4. planifie ;
5. puis les autres statuts non livrés.

15. Disponibilité client V2

Avant d'affecter définitivement un véhicule et de créer le rendez-vous, le DA doit demander au serveur central si le client accepte le couple **VIN + créneau**.

```
POST /api/v1/commandes/{id-ou-reference}/disponibilite-client
Content-Type: application/json
```

Corps :

```
{
  "vin": "XP726090200001017",
  "date_livraison_proposee": "2026-09-10 14:00:00"
}
```

`date_rendez_vous` est également accepté comme alias du champ `date_livraison_proposee`, mais le nom recommandé reste `date_livraison_proposee`.

Contrôles effectués

Le central vérifie :

- la commande appartient au centre ;
- la commande n'est ni livrée ni annulée ;
- le VIN appartient au centre ;
- le véhicule n'est pas livré ;
- le véhicule n'est pas affecté à une autre commande ;
- version et options compatibles ;
- créneau valide selon horaires, préparation et capacité ;
- respect du préavis minimum.

Probabilités par défaut

Configuration par défaut :

- préavis minimum : **48 h** ;
- seuil favorable : **96 h** ;
- entre 48 h et moins de 96 h : **60 %** de réponse OUI ;
- à partir de 96 h : **80 %** de réponse OUI ;
- après un refus : **80 %** de probabilité que le client soit indisponible pour toute la journée.

Ces valeurs peuvent être configurées côté serveur central.

Mémorisation des réponses

La disponibilité n'est pas retirée au hasard à chaque appel.

Même client + même créneau

Le serveur réutilise la première réponse obtenue.

```
"reponse_source": "reponse_existante"
```

La cohérence est conservée même si le même client/créneau est interrogé depuis une autre commande ou avec un autre VIN : une ligne de demande est alors créée pour conserver la validation du nouveau couple commande/VIN sans refaire de tirage.

Journée entière indisponible

Après un **NON**, le serveur peut déclarer le client indisponible pour toute la journée.

Dans ce cas, toute nouvelle proposition sur cette date retourne automatiquement **NON** avec :

```
{
  "reponse_source": "jour_indisponible",
  "jour_indisponible": true
}
```

Une journée ne peut pas devenir totalement indisponible si le client a déjà accepté un créneau le même jour.

Nouveau tirage

Pour un créneau jamais demandé :

```
"reponse_source": "nouveau_tirage"
```

Exemple de réponse positive

```
{
  "request_id": 42,
  "commande_reference": "CMD-260904-00123",
  "vin": "XP726090200001017",
  "date_livraison_proposee": "2026-09-10 14:00:00",
  "delai_heures": 121.5,
  "probabilite_appliquee": 80,
  "client_disponible": true,
  "reponse": "OUI",
  "reponse_source": "nouveau_tirage",
  "jour_indisponible": false,
  "message": "Le client confirme être disponible pour le créneau proposé."
}
```

Une réponse NON reste une réponse HTTP 200 : l'appel a réussi, mais le client simulé refuse la proposition.

Historique des propositions

```
GET /api/v1/commandes/{id-ou-reference}/disponibilites-client
```

Retourne notamment :

- id de la demande ;
- vehicule_id ;
- vin ;
- date_proposee ;
- delai_heures ;
- probabilite_appliquee ;
- reponse ;
- date_demande ;
- jour_indisponible.

16. Affecter ou désaffecter un véhicule

16.1 Affectation

```
POST /api/v1/commandes/{id-ou-reference}/affectation
```

Corps :

```
{
  "vin": "XP726090200001017"
}
```

Le central vérifie :

- commande du bon centre ;
- commande non clôturée ;
- VIN du bon centre ;
- VIN non livré ;
- VIN non déjà utilisé ;
- version compatible ;
- toutes les options compatibles.

Le VIN peut être affecté même s'il est encore en amont ou en transport. Il ne pourra être livré que lorsqu'il sera physiquement disponible dans le centre.

Si le véhicule est déjà disponible, la commande passe à `prete`; sinon elle passe à `vehicule_affecte`.

16.2 Désaffectation REST

```
DELETE /api/v1/commandes/{id-ou-reference}/affectation
```

16.3 Désaffectation compatible clients HTTP simples

```
POST /api/v1/commandes/{id-ou-reference}/desaffectation
```

Corps facultatif :

```
{
  "message": "Réaffectation nécessaire après échange avec le client."
}
```

La commande revient à `a_affecter`.

Une commande déjà `livree` ne peut plus être désaffectée.

17. Rendez-vous

```
POST /api/v1/commandes/{id-ou-reference}/rendez-vous
```

Corps :

```
{
  "date_rendez_vous": "2026-09-10 14:00:00",
  "note": "Le client souhaite une présentation des fonctions Autopilot."
}
```

Le rendez-vous est accepté uniquement si :

- la commande appartient au centre ;
- la commande n'est pas clôturée ;
- un véhicule est déjà affecté ;
- le client a déjà répondu **OUI** pour ce véhicule et ce créneau exact ;
- le créneau respecte toujours horaires, disponibilité du VIN et capacité ;
- le créneau n'est pas devenu complet entre la proposition et l'enregistrement.

Le serveur verrouille le Delivery Center pendant la réservation afin d'empêcher deux requêtes concurrentes de consommer simultanément la dernière place d'un créneau.

18. Planning

```
GET /api/v1/planning
```

Exemple :

```
GET /api/v1/planning?from=2026-09-08&to=2026-09-14
```

Paramètres :

Paramètre Obligatoire	Description
from	non début, défaut = aujourd'hui
to	non fin, défaut = from + 13 jours

La période ne peut pas dépasser **92 jours**.

La réponse contient :

- from;

- to;
- `duree_creneau_minutes = 60;`
- `capacite_par_creneau = 5;`
- `rendez_vous.`

Chaque rendez-vous contient les informations commande, client et véhicule.

19. Signaler un incident

```
POST /api/v1/commandes/{id-ou-reference}/incident
```

Corps :

```
{
  "type": "client_absent",
  "message": "Client absent au rendez-vous."
}
```

`message` est obligatoire.

Si `type` est absent, la valeur `incident` est utilisée.

Le type d'événement enregistré devient :

```
incident_{type}
```

Codes conseillés :

```
client_absent
client_report
vehicule_endommagement
document_manquant
probleme_paiement
autre
```

La réponse est créée avec HTTP 201.

20. Confirmer une livraison

```
POST /api/v1/commandes/{id-ou-reference}/livraison
```

Corps minimal :

```
{ }
```

Ou :

```
{  
  "date_livraison": "2026-09-10 14:52:00",  
  "commentaire": "Livraison terminée sans réserve."  
}
```

Sans `date_livraison`, le serveur utilise l'heure courante.

Le central refuse la clôture si :

- aucun VIN n'est affecté ;
- la commande est déjà livrée ;
- aucun rendez-vous n'est enregistré ;
- le véhicule n'appartient plus au centre ;
- le véhicule n'est pas disponible ;
- aucun accord client OUI ne correspond au VIN et au rendez-vous exacts.

Après validation :

- commande → livrée ;
- `date_livraison_reelle` est enregistrée ;
- véhicule → livre ;
- la place de parking active du véhicule est fermée immédiatement ;
- un événement `livraison_effectuee` est ajouté.

21. Parking d'agence

```
GET /api/v1/parking
```

Pour afficher un autre créneau :

```
GET /api/v1/parking?at=2026-09-08%2014:00:00
```

Le paramètre `at` est interprété en Europe/Paris puis ramené au début du créneau concerné.

La route :

1. synchronise d'abord l'occupation du parking ;
2. crée automatiquement un plan générique si le centre n'en possède pas encore ;

3. retourne la géométrie logique, les places, les véhicules présents et les retraits prévus sur le créneau.

Exemple de structure

```
{
  "centre": {
    "id": 10,
    "code": "BE_BRUXELLES",
    "nom": "Tesla Bruxelles",
    "ville": "Bruxelles"
  },
  "creneau": {
    "debut": "2026-09-08 14:00:00",
    "fin": "2026-09-08 15:00:00",
    "duree_minutes": 60,
    "est_heure_courante": true
  },
  "stats": {
    "places": 70,
    "occupees": 54,
    "libres": 16,
    "retraits_heure": 3
  },
  "allees": [],
  "retraits": []
}
```

Contenu des allées et places

Chaque allée contient ses places.

Une place contient notamment :

- id;
- code;
- cote : gauche **ou** droite ;
- position;
- occupee ;
- vehicule **si présente** ;
- commande **si le VIN est associé** à un dossier client.

retraits

La liste `retraits` contient les clients dont le rendez-vous se situe dans le créneau affiché, avec :

- la commande ;
- le client ;
- le véhicule ;
- la place et l'allée lorsque le véhicule est encore garé.

Gestion automatique des places

Le serveur :

- affecte la première place libre aux véhicules physiquement présents ;
- libère une place lors d'une livraison ;
- libère une place si le véhicule quitte le centre ;
- libère une place à la fin du créneau de retrait ;
- conserve l'historique dans `parking_affectation`.

Le plan du parking est générique et fonctionne pour **tous les Delivery Centers**.

Les vues **3D** et **Dessus** de l'espace DA utilisent la même réponse `GET /parking`. Le changement de caméra est uniquement une fonctionnalité d'interface : il n'ajoute pas de nouvelle route API.

22. Transferts inter-agences

22.1 Espace de transferts

```
GET /api/v1/transferts
```

La réponse contient :

```
{
  "regles": {
    "delai_da_jours": 3,
    "delai_central_jours": 4,
    "vehicule_source_requis": "disponible"
  },
  "centres": [],
  "vehicules_externes": [],
  "mes_vehicules_transferables": [],
  "demandes_recues": [],
  "demandes_envoyees": [],
  "transferts_en_cours": [],
  "historique": []
}
```

Véhicule transférable entre DA

Pour être proposé ou demandé, un véhicule doit :

- être physiquement `disponible` dans le centre source ;
- ne pas être affecté à une commande active ;
- ne pas avoir de transfert `en_attente` ou `en_transit` ;
- ne correspondre à **aucune commande locale non affectée** du centre source.

La liste `vehicules_externes` expose en priorité les véhicules susceptibles de couvrir des besoins du centre courant via `besoins_chez_nous`.

22.2 Demander un véhicule à un autre centre

POST `/api/v1/transferts/demande`

Corps :

```
{
  "vin": "XP7...",
  "commentaire": "Besoin pour une commande urgente."
}
```

`commentaire` est facultatif.

Le centre source est déduit du propriétaire actuel du VIN et le centre destination est le centre authentifié.

Réponse HTTP : 201.

22.3 Proposer un véhicule à un autre centre

```
POST /api/v1/transferts/proposition
```

Corps :

```
{
  "vin": "XP7...",
  "destination_center_id": 12,
  "commentaire": "Véhicule libre dans notre stock."
}
```

vin et destination_center_id sont obligatoires.

Le centre authentifié doit être le propriétaire du VIN.

Réponse HTTP : 201.

22.4 Accepter

```
POST /api/v1/transferts/{id}/accepter
```

Corps facultatif :

```
{
  "commentaire": "Transfert accepté."
}
```

Seul le centre qui doit réellement prendre la décision peut répondre :

- pour une demande, le centre source ;
- pour une proposition, le centre destination.

Après acceptation DA → DA :

- transfert → en_transit ;
- véhicule → transport ;
- destination du véhicule mise à jour immédiatement ;
- départ = maintenant ;
- arrivée prévue = **maintenant + 3 jours**.

22.5 Refuser

```
POST /api/v1/transferts/{id}/refuser
```

Corps facultatif identique :

```
{  
  "commentaire": "Besoin finalement couvert autrement."  
}
```

Le transfert passe à `refuse`.

Différence avec le serveur central

Les décisions DA → DA utilisent un délai de **3 jours**.

Les transferts décidés depuis le back-office central utilisent **4 jours**. Le central peut également réacheminer un véhicule déjà en transit ; cette logique n'est pas exposée comme action à une clé DA.

23. Mode automatique Delivery Advisor — Bruxelles uniquement

Le mode automatique est volontairement limité au Delivery Center :

```
BE_BRUXELLES
```

Une clé d'un autre centre reçoit une erreur 422 :

```
Le mode automatique Delivery Advisor est réservé au centre de Bruxelles.
```

23.1 Lire l'état

```
GET /api/v1/automatisation
```

La réponse contient notamment :

- `actif;`
- `date_demarrage;`
- `date_pause;`
- `derniere_execution;`
- `derniere_action;`
- `dernier_message;`
- `nb_propositions;`
- `nb_refus;`
- `nb_rendez_vous;`
- `nb_erreurs;`
- `nb_cycles_sans_action;`
- `disponible;`
- `centre.`

23.2 Démarrer

```
POST /api/v1/automatisation/demarrer
```

Aucun corps n'est requis.

Le serveur :

- **passer** `actif` à `true`;
- **mémoriser** `date_demarrage`;
- **effacer** `date_pause`;
- **mettre à jour** le message d'état.

Une fois actif, le cron `cron/update_statuses.php` exécute un cycle automatique pour Bruxelles à chaque passage du cron.

23.3 Pause

```
POST /api/v1/automatisation/pause
```

Aucun corps n'est requis.

La pause est persistante en base. L'action déjà entrée dans son traitement peut se terminer, puis aucun nouveau cycle ne doit démarrer.

23.4 Exécuter un cycle

POST /api/v1/automatisation/tick

Aucun corps n'est requis.

Cette route exécute **une seule décision automatique**. Elle est principalement destinée au moteur du DA et au cron.

Statuts possibles :

status	Signification
paused	mode inactif
busy	un autre cycle détient déjà le verrou
idle	aucune action utile à faire
refused	le client a refusé la proposition
appointment	accord client, affectation si nécessaire et rendez-vous créés

Logique du pilote automatique

Le moteur :

1. termine d'abord les commandes déjà affectées mais sans rendez-vous ;
2. réutilise un accord client déjà obtenu si le créneau est encore valide ;
3. sinon prend un véhicule libre qui couvre une commande ;
4. parcourt les jours disponibles ;
5. choisit les clients compatibles dans l'ordre chronologique ;
6. ne repropose pas un créneau déjà refusé ;
7. après un refus sur une journée, passe au client suivant pour cette journée ;
8. si le client accepte, affecte le VIN si nécessaire ;
9. crée le rendez-vous sur exactement le créneau accepté ;
10. en cas d'échec de création du rendez-vous après une affectation automatique, désaffecte le VIN pour revenir à un état cohérent.

Un verrou SQL empêche deux cycles automatiques Bruxelles de s'exécuter en même temps.

24. Historique des événements

GET /api/v1/evenements

Paramètres :

Paramètre Obligatoire	Description
since	non événements depuis cette date
limit	non défaut 500, maximum 1000

Chaque événement contient :

- id;
- commande_id;
- type;
- source;
- message;
- donnees;
- date_evenement.

Exemples de types :

```
vehicule_affecte
vehicule_desaffecte
disponibilite_client_acceptee
disponibilite_client_refusee
rendez_vous_planifie
incident_client_absent
livraison_effectuee
```

25. Synchronisation incrémentale

```
GET /api/v1/sync?since=2026-09-08%2010:00:00
```

since est obligatoire.

La réponse contient :

```
{
  "serveur_time": "2026-09-08 13:00:00",
  "since": "2026-09-08 10:00:00",
  "centre": {},
  "vehicules": [],
  "commandes": [],
  "evenements": []
}
```

La route récupère jusqu'à :

- 1000 véhicules modifiés ;
- 1000 commandes modifiées ;
- 1000 événements.

Logique recommandée côté DA :

1. mémoriser l'heure du dernier sync
2. GET /sync?since=derniere_sync
3. mettre à jour la BDD/cache local
4. mémoriser data.serveur_time ou meta.server_time
5. réutiliser cette date au prochain appel

26. Workflow recommandé d'une affectation manuelle

Le flux normal d'un Delivery Advisor est :

1. GET /vehicules-matching
2. GET /vehicules/{VIN}/commandes-compatibles
3. GET /vehicules/{VIN}/creneaux
4. POST /commandes/{ref}/disponibilite-client
5. si NON : tester un autre client / jour / créneau
6. si OUI : POST /commandes/{ref}/affectation
7. POST /commandes/{ref}/rendez-vous avec le même créneau
8. le jour J : POST /commandes/{ref}/livraison

Le serveur central revalide toujours les règles au moment des actions. Une interface DA ne doit donc jamais considérer qu'une donnée ancienne garantit encore qu'une action sera acceptée.

27. Exemple PHP minimal

Le projet contient :

examples/api-da-client.php

Exemple :

```

const TESLA_API_URL = 'https://b3-demo1.kevinpecro.info/api/v1';
const TESLA_API_KEY = 'REPLACER_PAR_LA_CLE_DU_CENTRE';

function teslaApi(string $method, string $endpoint, ?array $body = null): array
{
    $url = rtrim(TESLA_API_URL, '/') . '/' . ltrim($endpoint, '/');
    $ch = curl_init($url);

    curl_setopt_array($ch, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_CUSTOMREQUEST => strtoupper($method),
        CURLOPT_HTTPHEADER => [
            'Authorization: Bearer ' . TESLA_API_KEY,
            'Accept: application/json',
            'Content-Type: application/json',
        ],
        CURLOPT_TIMEOUT => 15,
    ]);

    if ($body !== null) {
        curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($body, JSON_UNESCAPED_UNICODE));
    }

    $raw = curl_exec($ch);
    if ($raw === false) {
        throw new RuntimeException(curl_error($ch));
    }

    $http = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    $data = json_decode($raw, true);
    if (!is_array($data)) {
        throw new RuntimeException('Réponse JSON invalide.');
```

Exemple d'affectation :

```

$availability = teslaApi('POST', 'commandes/CMD-260902-120501-48672/disponibilite-client', [
    'vin' => 'XP7...',
    'date_livraison_proposee' => '2026-09-10 14:00:00',
]);

if ($availability['client_disponible']) {
    teslaApi('POST', 'commandes/CMD-260902-120501-48672/affectation', [
        'vin' => 'XP7...',
    ]);

    teslaApi('POST', 'commandes/CMD-260902-120501-48672/rendez-vous', [
        'date_rendez_vous' => '2026-09-10 14:00:00',
        'note' => 'Créneau confirmé par le client.',
    ]);
}

```

28. Modules SQL requis pour les dernières fonctionnalités

Sur une base historique qui n'a pas encore reçu les dernières migrations, les modules concernés sont notamment :

Fonctionnalité	Migration
Disponibilité client	sql/update_disponibilite_client.sql
Disponibilité client par journée	sql/update_disponibilite_client_jour.sql
Workspace DA / créneaux	sql/update_da_workspace.sql
Préparation après arrivée	sql/update_da_creneaux_preparation.sql
Règle 1 h / 5 clients	sql/update_da_creneaux_1h_5_clients.sql
Transferts inter-agences	sql/update_transferts_centres.sql
Parking	sql/update_parking_plan.sql
Mode automatique Bruxelles	sql/update_da_mode_automatique.sql

Le projet contient également le correctif d'alignement :

```
sql/update_bdd_2026-09-08.sql
```

Ce script corrige notamment les champs de créneaux, le snapshot du dashboard et l'état du mode automatique Bruxelles. **Il ne remplace pas** les migrations dédiées aux transferts, au parking ou à la disponibilité client si celles-ci n'ont jamais été importées.

Une installation neuve peut utiliser le schéma complet fourni par le projet.

29. Règles métier essentielles à retenir

- La clé API limite toutes les données au Delivery Center authentifié.
- Un DA ne choisit pas arbitrairement son `delivery_center_id`.
- Une version incompatible ou une option manquante bloque l'affectation.
- Un VIN déjà affecté à une autre commande ne peut pas être réutilisé.
- Un véhicule à venir peut être affecté, mais pas livré avant son arrivée réelle.
- Les créneaux sont de **60 minutes** avec **5 clients maximum**.
- Un véhicule à venir devient livrable au premier horaire d'ouverture du **lendemain** de son arrivée.
- Le préavis client est calculé depuis l'instant de la proposition, pas depuis l'arrivée du VIN.
- Une même réponse client pour un même créneau est mémorisée.
- Un refus peut rendre toute la journée indisponible pour ce client.
- Un rendez-vous exige un accord OUI préalable pour le même VIN et le même créneau.
- Une livraison exige un véhicule physiquement `disponible`.
- La livraison libère immédiatement la place de parking active.
- Les transferts DA → DA durent **3 jours** après acceptation.
- Le mode automatique est strictement réservé à `BE_BRUXELLES`.
- Toutes les actions importantes sont journalisées par le serveur central.
- Les données de ce projet sont fictives et destinées à une simulation pédagogique.

30. Architecture recommandée côté DA

API centrale Tesla

↓

GET /centre, /vehicules-matching, /commandes, /planning,
/stock-previsionnel, /parking, /transferts, /sync

↓

Cache / BDD locale éventuelle du DA

↓

Dashboard et aide à la décision

↓

POST disponibilité / affectation / rendez-vous / incident / livraison
POST transferts

↓

Serveur central Tesla

L'espace de démonstration /da/ fourni dans Serveur 26 respecte ce principe : le navigateur passe par `da/proxy.php`, qui ajoute la clé Bearer puis appelle réellement `/api/v1/` . . . Le dossier DA ne se connecte pas directement à MariaDB.

