Webhook Collection

This collection contains all necessary/preferred Webhooks that can be used by our clients.

Insert: Inserted in the database

Update: Updates in the database

Deleted: Deleted from the database

Action Items

	Webhooks	What it does	Example
	Backend		
1.	**Customers**		
1.1	backend Customer CustomerDeleted	webhook triggered, if: **a customer got deleted**	Operations Manager deletes Customer from the Wunder System
1.2	backend Customer afterInsert	webhook triggered, if: **a customer got created**	Operations Manager creates Customer from the Wunder System
2.	**Invoicing**		
2.1	backend Invoice afterInsert	webhook triggered, if: **an invoice got created**	A trip ended and an invoice was created after the trip
3.	**Location**		
3.1	backend Location afterUpdate	webhook triggered, if: **a location got changed**	Operations Manager changed the e.g suffix of the Location
4.	**Notification**		
4.1	backend Notification afterInsert	webhook triggered, if: **a notification got created**	Operations Manager sent out a Notification to the customers
5.	**Dynamic Parameters**		

5.1	backend Parameter afterInsert	webhook triggered, if: **a dynamic parameter got created**	Operations Manager added a new Dynamic Parameter
5.2	backend Parameter afterUpdate	webhook triggered, if: **a dynamic parameter got updated**	Operations Manager changed an existing Dynamic Parameter
5.3	backend Parameter afterDelete	webhook triggered, if: **a dynamic parameter got deleted**	Operations Manager deleted an Dynamic Parameter
6.	**Parking Tickets**		
6.1	backend ParkingTicket afterInsert	webhook triggered, if: **a parking ticket got created**	Customer parks the car and pulls a parking ticket. Operations Manager has configured a Parking Provider in the Wunder System.
6.2	backend ParkingTicket afterUpdate	webhook triggered, if: **a parking ticket got updated**	The state of the Parking Ticket changed, e.g is expired.
7.	**Payment Provider**		
7.1	backend PaymentNotification afterInsert	webhook triggered, if **the payment provider received a payment**	Payment Provider notifies the Wunder System that it received a payment

7.2	backend PaymentNotification afterUpdate	webhook triggered, if **the payment provider updated a payment**	Payment Provider notifies the Wunder System that it updated a payment
8.	**Refund**		
8.1	backend PaymentRefund afterInsert	webhook triggered, if **a refund got created**	Operations Manager is using the refund feature in the Backend to balance an invoice (just possible with Stripe and PayOne)
9.	**Payment Type**		
9.1	backend PaymentType afterInsert	webhook triggered, if **a payment type got activated**	Operations Manager has added e.g. Credit Card to his service in the Wunder System
9.2	backend PaymentType afterUpdate	webhook triggered, if **a payment type got updated**	Operations Manager has changed the state of a payment type e.g. from active inactive in the Wunder System
10.	**Plans**		
10.1	backend Plan afterInsert	webhook triggered, if **a plan got created**	Operations Manager created a new Plan
11.	**Promotion Code**		
11.1	backend PromotionCode afterInsert	webhook triggered, if **a promotion code got created**	Operations Manager created a new Promotion Code

11.2	backend PromotionCode afterUpdate	webhook triggered, if **a promotion code got updated**	Operations Manager updated an existing Promotion Code
11.3	backend PromotionCode afterDelete	webhook triggered, if **a promotion code got deleted**	Operations Manager deleted an existing Promotion Code
12.	**Reservations**		
12.1	backend Reservation reservationStart	webhook triggered, if **a reservation got created**	Operations Manager creates a new reservation through the Wunder System
13.	**Terms of Service**		
13.1	backend TermsRelease afterInsert	webhook triggered, if **a terms of service release got created**	Operations Manager creates a new Terms of Service Release
13.2	backend TermsRelease afterDelete	webhook triggered, if **a terms of service release got deleted**	Operations Manager deletes an existing Terms of Service Release
14.	**Territories**		
14.1	backend Territory afterInsert	webhook triggered, if **a territory got created**	Operations Manager creates a new Territory
14.2	backend Territory afterUpdate	webhook triggered, if **a territory got updated**	Operations Manager changed an already existing Territory

14.3	backend Territory afterDelete	webhook triggered, if **a territory got deleted**	Operations Manager deleted an already existing Territory
15.	**Files**		
15.1	backend UploadedFile afterInsert	webhook triggered, if **a file got uploaded**	A new file (verification, ID, After Rental Picture) got uploaded to the Wunder System
16.	**Vehicle States**		
16.1	backend VehicleStateChange afterInsert	webhook triggered, if **a vehicle state got created**	Operations Manager changed the state of a vehicle in the Wunder System
17.	**Vehicle Type**		
17.1	backend VehicleType afterInsert	webhook triggered, if **a vehicle type got created**	Operations Manager creates a new Vehicle Type
17.2	backend VehicleType afterUpdate	webhook triggered, if **a vehicle type got updated**	Operations Manager changes an existing Vehicle Type
18.	**Vehicle**		
18.1	backend Vehicle afterInsert	webhook triggered, if **a vehicle got created**	Operations Manager creates a new Vehicle
18.2	backend Vehicle afterUpdate	webhook triggered, if **a vehicle got updated**	Operations Manager changed the config of an already existing Vehicle

18.3	backend Vehicle afterDelete	webhook triggered, if **a vehicle got deleted**	Operations Manager deletes a Vehicle
	Frontend		
1.	**Data Changes**		
1.1	frontend DataChangeRequest afterInsert	webhook triggered, if **a change request got created**	Customer wants to change their data, e.g password, address, phone number, etc.
2.	**Vehicle State**		
2.1	frontend Car vehicleStateChanged	webhook triggered, if **a vehicle state change got created**	Vehicle State changed due to not having enough fuel automatically. This needs to be configured by the Operations Manager in the Wunder System
3.	**Reservation**		
3.1	frontend Reservation reservationEnd	webhook triggered, if **a reservation ended**	A reservation was started by a customer in the App
3.2	frontend Reservation reservationStart	webhook triggered, if **a reservation started**	A reservation was finished by a customer in the App