Admin API - Dokumentation

- Basket API – Dokumentation
- Transaktions API - Dokumentation
- Statistik API - Dokumentation
- Touchpoints / Customerjourney API - Dokumentation
- Order Webservice
  - Create / easy.affiliate REST-API (Orders)
  - Update / easy.affiliate REST-API (Orders)
  - Create / easy.affiliate REST-API (Baskets)
  - Update / easy.affiliate REST-API (Baskets)
- Publisher Webservice
  - Verwaltung von Projekten über die V6 REST-API
  - Publisher-Verwaltung über die V6 REST-API
- Externe API anbinden
- Werbemittel API
  - Admedia API (V6) - POST (Create)
  - Admedia API (V6) - GET (Read)
- Basket API
  - Basket Update API
- Conversion Trace API
- Touchpoint Webservice

    - EMID Webservice

- Webservice Nachbuchungen
    - Nachbuchungsanfragen per API abrufen und bearbeiten (V6-REST-API)

Basket API – Dokumentation

Die Basket API dient dem Export und der Analyse von Warenkorb-Daten, die über das Partnerprogramm-System erfasst wurden. Sie ermöglicht es, detaillierte Informationen zu einzelnen Basket-Items strukturiert als CSV-Datei abzurufen und weiterzuverarbeiten.

Die API kann im User-Interface unter dem Menüpunkt **Daten-API** verwendet werden.

API-Aufruf

https://**www.domain.de**/api//**ACCESS-TOKEN**/admin/**ID**/get-orders_baskets**.csv**

Wert	Beschreibung
www.domain.de	System-Partnerprogramm
ACCESS-TOKEN	Access Token des Users
ID	User-ID
.csv	Art des Downloads

Basket-API Filterung

Für die Basket API können verschiedene Parameter als Filter verwendet werden.
Nachfolgend eine Übersicht aller verfügbaren Parameter:

Parameter	Wert / Beispiel	Beschreibung
`condition[period][from]`	`01.01.2025`	Definiert das **Startdatum** des Zeitraums
`condition[period][to]`	`31.01.2025`	Definiert das **Enddatum** des Zeitraums
`condition[l:campaigns]`	`1`	Filtert alle Baskets, die einer bestimmten **Kampagne** zugeordnet sind. Mehrfachnennung möglich.

Parameter	Wert / Beispiel	Beschreibung
`condition[datemode]`	`processing` `payout` `incoming`	Bestimmt die Art des Zeitraums: • `processing`: Baskets, die im Zeitraum **validiert** wurden • `payout`: Baskets, die im Zeitraum **ausgezahlt** wurden • `incoming`: Baskets, die im Zeitraum **erfasst** wurden (Transaktionsdatum)

Beispiel:

```
condition[period][from]=01.01.2025&condition[period][to]=31.01.2025&condition[datemode]=processing
```

Definition der Felder in der Basket-API

Die heruntergeladenen Transaktionen (z.B. als `.csv`) enthalten folgende Spalten:

Feldname	Beschreibung
`id`	Eindeutige ID des Basket-Items
`orderid`	Interne ID der zugehörigen Transaktion
`timestamp`	Zeitpunkt der Transaktion
`ordertoken`	Bestellnummer der Transaktion
`trigger_id`	ID des Triggers, der der Transaktion zugeordnet ist
`is_basket`	Gibt an, ob es sich um ein Basket-Item handelt (`true` / `false`)
`basket_item_id`	Eindeutige ID des Basket-Items
`category`	Übergebene Kategorie des Shops
`productid`	ID des Produkts im Shop
`productname`	Name des Produkts im Shop
`amount`	Anzahl der gekauften Produkte
`price`	Stückpreis des Produkts
`original_price`	Ursprünglicher Stückpreis vor Änderungen
`status`	Status des Basket-Items
`total`	Gesamtpreis (Anzahl × Preis)
`commission`	Provision, die der Publisher für das Basket-Item erhält

Feldname	Beschreibung
`currency_rate`	Umrechnungsrate bei Fremdwährung
`currency`	Währung der Transaktion
`is_attributed`	Gibt an, ob eine Attribution vorliegt (`true` / `false`)
`attribution`	Art der Attribution pro Basket-Item
`attributed_price`	Attribuierter Preis pro Basket-Item
`attributed_total`	Attribuierter Preis × Anzahl
`trigger_value`	Provisionshöhe pro Basket-Item
`original_turnover`	Getrackter Umsatz
`price_brutto`	Brutto-Preis des Basket-Items

Transaktions API - Dokumentation

Über die Transaktions-API holen Sie alle Transaktionen Ihres Partnerprogramms als Datei oder als JSON ab: mit Status, Umsatz, Provision, Publisher und Projekt. Damit lassen sich Abrechnung, Kostenauswertung und die Anbindung externer Tools (BI, Attribution, Data Warehouse) automatisieren. Diese Seite beschreibt, wie Sie einen Zugang einrichten, wie der Aufruf aussieht, welche Filter es gibt und was in jeder Spalte steht.

Überblick

Schritt	Was passiert	Wo
1. Zugang einrichten	eigener Benutzer mit eigenem API-Token, nur mit Leserechten	Systemsteuerung → Berechtigungsgruppen und Benutzerverwaltung
2. Link holen	fertiger Abruflink mit Token, Benutzer-ID und Format	API → Daten-API, Zeile **Transaktionen**
3. Filtern und abrufen	Zeitraum, Status, Kampagne an den Link hängen und regelmäßig abrufen	Ihr System oder das externe Tool

---

Schritt 1: Zugang einrichten

Jeder Benutzer im System hat seinen eigenen API-Token. Der Token gibt genau die Daten frei, die dieser Benutzer auch in der Oberfläche sehen darf.

Für externe Tools und Dienstleister empfehlen wir immer einen eigenen Benutzer, nicht Ihren persönlichen Zugang. So bestimmen Sie selbst, was das Tool sieht, und können den Zugang jederzeit sperren oder den Token erneuern, ohne dass Ihr eigener Login betroffen ist.

Berechtigungsgruppe anlegen

1. **Systemsteuerung → Berechtigungsgruppen** öffnen und **Neue Gruppe anlegen** klicken.
2. Einen sprechenden Namen vergeben, zum Beispiel `API lesend`.
3. Nur das freigeben, was das Tool braucht:

Berechtigung	Einstellung	Wofür
Transaktionsliste	`ACCESS_READ`	Transaktions-API
Export-Links	`yes`	Abruflinks auf der Seite Daten-API
Publisher Daten anzeigen	`yes`, wenn Publisher-Namen gebraucht werden	Spalten `publisher_prename`, `publisher_surname`, `publisher_searchtitle`
Statistiken, die abgerufen werden sollen	`ACCESS_READ`	nur bei Nutzung der Statistik-API
alles andere	`ACCESS_DENY` bzw. `no`	

Was die einzelnen Rechte bedeuten, steht unter Berechtigungen für Nutzer des Systems einstellen.

Benutzer anlegen

1. **Systemsteuerung → Benutzerverwaltung** öffnen und einen neuen Benutzer anlegen.
2. Name und E-Mail so wählen, dass klar ist, wofür der Zugang da ist, zum Beispiel `API Kostenimport`.
3. Die eben angelegte Berechtigungsgruppe zuweisen und speichern.

Token ablesen

- **Als dieser Benutzer einloggen** und **API → Daten-API** öffnen. Dort stehen der Token und die fertigen Abruflinks.
- Superuser sehen den Token auch direkt in der **Benutzerverwaltung** beim Benutzer, Feld **API-Token** (Auge-Symbol zum Einblenden).

Token erneuern: In der Benutzerverwaltung über das Pfeil-Symbol neben dem Token oder auf der Seite Daten-API über **Neu generieren**. Der alte Token ist danach sofort ungültig. Das externe Tool braucht dann den neuen Link.

Zugang beenden: Benutzer in der Benutzerverwaltung deaktivieren oder löschen. Der Token funktioniert danach nicht mehr.

Schritt 2: Der Aufruf

```
https://<Systemdomain>/api/<ACCESS-TOKEN>/admin/<BENUTZER-ID>/get-orders.<FORMAT>
```

Teil	Inhalt
`<Systemdomain>`	Domain Ihres Systems, die Sie auch für den Login nutzen
`<ACCESS-TOKEN>`	API-Token des Benutzers aus Schritt 1
`<BENUTZER-ID>`	ID dieses Benutzers
`<FORMAT>`	`csv`, `json` oder `xml`

Am einfachsten kopieren Sie den Link fertig von **API → Daten-API**, Zeile **Transaktionen**. Token und Benutzer-ID sind dort schon eingesetzt.

Beispiel: alle bestätigten und ausgezahlten Transaktionen aus Januar 2026 als JSON

```
https://<Systemdomain>/api/<ACCESS-TOKEN>/admin/<BENUTZER-ID>/get-orders.json?condition[l:status]=confirmed,paid&condition[period][from]=01.01.2026&condition[period][to]=31.01.2026&condition[datemode]=incoming
```

Mit `curl`:

```
curl -g "https://<Systemdomain>/api/<ACCESS-TOKEN>/admin/<BENUTZER-ID>/get-orders.csv?condition[period][from]=01.01.2026&condition[period][to]=31.01.2026" -o transaktionen.csv
```

`-g` ist nötig, damit `curl` die eckigen Klammern nicht selbst auswertet.

Antworten

Code	Bedeutung
`200`	Daten kommen im gewählten Format.
`401`	Token oder Benutzer-ID passen nicht zusammen, oder der Token wurde erneuert.
`404`	Die Adresse stimmt nicht, meist ein Tippfehler im Pfad oder Format.

Schritt 3: Filtern

Alle Filter werden als Parameter an den Link gehängt und mit `&` verbunden.

Zeitraum

Parameter	Beispiel	Bedeutung
`condition[period][from]`	`01.01.2026`	Startdatum
`condition[period][to]`	`31.01.2026`	Enddatum
`condition[datemode]`	`incoming`	Worauf sich der Zeitraum bezieht, siehe unten

`datemode`	Der Zeitraum gilt für
`incoming`	das Transaktionsdatum, also wann die Bestellung erfasst wurde
`processing`	das Datum der Bestätigung oder Stornierung
`payout`	das Auszahlungsdatum

Für Kostenauswertungen nach Bestelldatum nehmen Sie `incoming`, für die tatsächlich ausgezahlten Provisionen eines Monats `payout`.

Status

Parameter	Wert	Liefert
`condition[l:status]`	`open`	offene Transaktionen
	`confirmed`	bestätigte Transaktionen
	`paid`	ausgezahlte Transaktionen
	`canceled`	stornierte Transaktionen

Mehrere Status mit Komma trennen: `condition[l:status]=confirmed,paid`

Kampagne

Parameter	Beispiel	Bedeutung
`condition[l:campaigns]`	`1` oder `1,3`	nur Transaktionen dieser Kampagnen

Die Felder

Die Datei enthält eine Zeile je Transaktion. Beträge sind netto, sofern nicht anders angegeben.

Transaktion

Feld	Inhalt
`id`	Eindeutige ID der Transaktion in unserem System
`ordertoken`	Ihre Bestellnummer, wie sie beim Tracking übergeben wurde
`ebestid`	Verschlüsselte Bestellnummer, wie der Publisher sie sieht
`type`	Art der Transaktion, zum Beispiel Sale, Lead oder View
`status`	0 offen, 1 bestätigt, 2 storniert, 3 ausgezahlt
`sub_status`	Ihr eigener Bearbeitungsstatus
`cancel_reason`	Stornogrund
`visibility`	Sichtbarkeit für den Publisher
`timestamp`	Zeitpunkt der Transaktion
`order_timestamp`	Zeitpunkt des Bestellaufrufs
`action_timestamp`	Zeitpunkt des Klicks oder Views, der zur Bestellung führte
`status_change_date`	Datum der letzten Statusänderung
`last_change`	Zeitpunkt der letzten Änderung
`payoutdate`	Auszahlungsdatum
`description`	Beschreibung, die mit der Transaktion übergeben wurde
`basket_items`	Anzahl der Artikel im Warenkorb

Umsatz und Provision

Feld	Inhalt
`turnover`	Provisionsrelevanter Bestellwert
`turnover_brutto`	Bestellwert inklusive Steuern
`source_turnover`	Bestellwert, wie er ursprünglich übergeben wurde
`original_turnover`	Bestellwert in der Währung der Bestellung
`attributed_turnover`	Anteiliger Bestellwert, wenn ein Attributionsmodell aktiv ist
`order_currency`	Währung der Bestellung

Feld	Inhalt
`commission`	**Provision des Publishers**, aktueller Stand nach dem Sales-Abgleich
`source_commission`	Provision zum Zeitpunkt der Transaktion, ändert sich nicht
`attribution`	Betrag der gezahlten Attribution
`trigger_id`	Trigger der Transaktion
`trigger_title`	Name des Triggers
`trigger_type`	Fixvergütung oder prozentuale Vergütung
`trigger_value`	Vergütung, die zum Zeitpunkt der Transaktion im Trigger hinterlegt war
`commission_group_id`	Provisionsgruppe des Publishers zum Zeitpunkt der Transaktion
`campaign_group_title`	Name dieser Provisionsgruppe
`salary_id`	Auszahlung, mit der die Transaktion abgerechnet wurde
`bonus_id`	Bonuslauf, falls die Transaktion aus einem Bonus stammt

Publisher, Projekt und Kampagne

Feld	Inhalt
`publisher_id`	ID des Publishers
`publisher_prename`, `publisher_surname`	Vor- und Nachname des Publishers
`publisher_searchtitle`	Suchtitel des Publishers
`project_id`	ID des Projekts (Website oder Kanal des Publishers)
`project_title`	Name des Projekts
`campaign_id`	ID der Kampagne
`campaign_title`	Name der Kampagne
`advertiser_id`	ID des Advertisers
`advertiser_title`	Name des Advertisers
`admedia_id`	Werbemittel, über das die Transaktion entstand
`subid`	Sub-ID, die der Publisher mitgegeben hat

Herkunft und Tracking

Feld	Inhalt
`source`	Quelle der Transaktion
`referrer`	Seite, auf der geklickt wurde
`country`	Land der Transaktion
`user_agent`	Browser und Gerät des Kunden
`session_id`	Session-ID, die mit der Transaktion übergeben wurde
`action_id`	Action-ID, die mit der Transaktion übergeben wurde
`order_actions_id`	Verweis auf den provisionsrelevanten Klick oder View
`customer_journey_status`	Kennziffer zur Customer Journey
`delivered_tagcode_count`	Anzahl der ausgespielten Tags
`delivered_tagcode_serversided_url`	Übermittelter serverseitiger Trackingaufruf

Externe Quellen

Nur gefüllt, wenn Transaktionen aus einem angebundenen Netzwerk oder einer externen Quelle stammen.

Feld	Inhalt
`external_sources_entity_id`	Konto-ID in der externen Quelle
`external_sources_turnover`	Bestellwert laut externer Quelle
`external_sources_currency_code`	Währung in der externen Quelle
`external_sources_currency_rate`	Umrechnungskurs in Ihre Systemwährung
`source_project_id`	Publisher- oder Projekt-ID in der externen Quelle
`source_project_title`	Publisher- oder Projektname in der externen Quelle

Häufige Fragen

Wie komme ich an die Affiliate-Kosten je Publisher? Bestätigte und ausgezahlte Transaktionen abrufen (`condition[l:status]=confirmed,paid`) und die Spalte `commission` je `publisher_id` oder `project_title` summieren. Offene Transaktionen können sich im Sales-Abgleich noch ändern.

Brauche ich nur Summen statt einzelner Transaktionen? Dafür ist die Statistik-API da, zum Beispiel `get-statistic_publisher` oder `get-statistic_daily`.

Wie oft darf ich abrufen? Einmal pro Stunde oder einmal pro Nacht reicht für fast alle Fälle. Rufen Sie lieber kurze Zeiträume häufiger ab als ein ganzes Jahr auf einmal.

Das Tool bekommt plötzlich `401`. Der Token wurde erneuert oder der Benutzer deaktiviert. Neuen Link von der Seite Daten-API holen und im Tool hinterlegen.

Statistik API - Dokumentation

Die Statistik API ermöglicht den Export und die Auswertung aggregierter Leistungsdaten im Rahmen des Partnerprogramm-Systems. Sie stellt unterschiedliche Statistik-Arten bereit, wie z. B. Auswertungen nach Tag, Kampagne, Projekt oder Werbemittel, und unterstützt so bei der Performance-Analyse und Optimierung.

Alle APIs können im User-Interface unter dem Menüpunkt **Daten-API** verwendet werden.

API-Aufruf

https://**www.domain.de**/api//**ACCESS-TOKEN**/admin/**ID**/get-statistic_**daily.csv**

Wert	Beschreibung
www.domain.de	System-Partnerprogramm
ACCESS-TOKEN	Access Token des Users
ID	User-ID
get-statistic_{Statistik-Art}	Art der Statistik (z.B. Tagesbasiert, etc.)
.csv	Art des Downloads

Statistik-API Filterung

Für die Statistik API können verschiedene Parameter als Filter verwendet werden.
Nachfolgend eine Übersicht aller verfügbaren Parameter:

Statistik-Arten

Statistik-Art	Parameter
Statistik nach Tag	`get-statistic_daily`
Statistik nach Advertiser	`get-statistic_advertiser`

Statistik-Art	Parameter
Statistik nach Kampagnen	`get-statistic_campaign`
Statistik nach Werbemittel	`get-statistic_media`
Statistik nach Publisher	`get-statistic_publisher`
Statistik nach Projekt	`get-statistic_project`
Statistik nach SubID	`get-statistic_subid`
Statistik nach Referrer	`get-statistic_referrer`
Statistik nach Channel	`get-statistic_channel`
Statistik nach Provisionsgruppe	`get-statistic_provisionsgroup`
Statistik nach Trigger	`get-statistic_trigger`
Statistik nach Journey	`get-statistic_journey`
Statistik nach A/B-Test	`get-statistic_abtest`
Statistik nach Gerätetyp	`get-statistic_devicetype`
Statistik nach Gerät	`get-statistic_device`
Statistik nach Browser	`get-statistic_browser`
Statistik nach Betriebssystem	`get-statistic_os`
Statistik nach Land	`get-statistic_geocountry`
Manuelle Statistik	`get-statistic_manualstatistics`

Parameter	Wert / Beispiel	Beschreibung
`condition[period][from]`	`01.01.2025`	Startdatum der Statistik
`condition[period][to]`	`31.01.2025`	Enddatum der Statistik
`condition[paymentstatus]`	`all` = alle `0` = nicht ausgezahlt `1` = ausgezahlt	Filtert nach Auszahlungsstatus
`condition[dynamicdate]`	`currentmonth` = aktueller Monat `lastmonth` = vorheriger Monat `currentweek` = aktuelle Woche `last10d` = letzte 10 Tage	Dynamische Zeiträume
`condition[l:publisher]`	`1234,5678`	Filtert nach Publisher-IDs

Parameter	Wert / Beispiel	Beschreibung
`condition[project_id]`	`12345`	Filtert nach Projekt-IDs
`condition[bruttodata]`	`true`	Gibt Bruttozahlen zurück (z. B. Klicks/Views)
`condition[l:mandant]`	`1`	Filtert nach Mandanten-IDs
`condition[l:advertiser]`	`2,3`	Filtert nach Advertiser-IDs
`condition[l:campaigns]`	`1,3`	Filtert nach Kampagnen-IDs

Aufschlüsselung nach weiteren Kriterien (Dynamische-Statistik)

Die Statistik kann zusätzlich nach weiteren Kriterien **aufgeschlüsselt (segmentiert)** werden. Hierzu wird der Parameter `condition[l:criterions]` verwendet.

Parameter	Wert	Beschreibung	Beispiel
`condition[l:criterions]`	`day`	Aufschlüsselung nach Tag	`condition[l:criterions]=day`
	`publisher`	Aufschlüsselung nach Publisher	`condition[l:criterions]=publisher`
	`media`	Aufschlüsselung nach Werbemittel	`condition[l:criterions]=media`
	`advertiser`	Aufschlüsselung nach Advertiser	`condition[l:criterions]=advertiser`
	`campaign`	Aufschlüsselung nach Kampagnenname	`condition[l:criterions]=campaign`
	`campaign_id`	Aufschlüsselung nach Kampagnen-ID	`condition[l:criterions]=campaign_id`
Kombination	z. B. `day,publisher,media`	Mehrdimensionale Aufschlüsselung	`condition[l:criterions]=day,publisher,media`

Hinzufügen von KPIs

Über den Parameter `condition[l:additional_parameters]` können zusätzliche **Key Performance Indicators (KPIs)** zur Statistik hinzugefügt werden.

Parameter	Wert	Beschreibung	Beispiel
`condition[l:additional_parameters]`	`all`	Fügt alle verfügbaren KPIs hinzu	`condition[l:additional_parameters]=all`

Definition der Felder in der Statistik API

Die Spaltennamen der heruntergeladenen Datei beinhalten folgende Parameter:

Feldname	Beschreibung
`criterion`	Das Kriterium, welches für die Statistik ausgewählt wurde. Wurde beispielsweise die Statistik nach Tag ausgewählt, werden in diesem Feld die betroffenen Tage ausgegeben. Wird Statistik nach Campagne ausgewählt, werden hier die entsprechenden Kampagnen-ID ausgewählt. In der Regel wird hier die ID ausgegeben.
`title`	Der Name der entsprechend ausgewählten Criterion. Hier steht z. B. der Name der Kampagne oder des Projekts.
`leads_open_count`	Anzahl der offenen Leads.
`leads_confirmed_count`	Anzahl der bestätigten Leads.
`leads_canceled_count`	Anzahl der stornierten Leads.
`sales_open_count`	Anzahl der offenen Sales.
`sales_confirmed_count`	Anzahl der bestätigten Sales.
`sales_canceled_count`	Anzahl der stornierten Sales.
`all_open_count`	Anzahl der offenen Leads und Sales addiert.
`all_confirmed_count`	Anzahl der bestätigten Leads und Sales addiert.
`all_canceled_count`	Anzahl der stornierten Leads und Sales addiert.
`leads_open_provision`	Provisionssumme der offenen Leads.
`leads_confirmed_provision`	Provisionssumme der bestätigten Leads.
`leads_canceled_provision`	Provisionssumme der stornierten Leads.
`sales_open_provision`	Provisionssumme der offenen Sales.

`sales_confirmed_provision`	Provisionssumme der bestätigten Sales.
`sales_canceled_provision`	Provisionssumme der stornierten Sales.
`all_open_provision`	Provisionssumme der offenen Leads und Sales addiert.
`all_confirmed_provision`	Provisionssumme der bestätigten Leads und Sales addiert.
`all_canceled_provision`	Provisionssumme der stornierten Leads und Sales addiert.
`views`	Anzahl der Views.
`clicks`	Anzahl der Klicks.
`epc`	Die berechnete, durchschnittliche Einnahme pro Klick.
`ctr`	Der Wert gibt an, wie viele User tatsächlich auf das Werbemittel klickten, nachdem sie es gesehen haben.
`leads_canceled_turnover`	Gesamter Warenkorbwert der stornierten Leads.
`leads_open_turnover`	Gesamter Warenkorbwert der offenen Leads.
`leads_confirmed_turnover`	Gesamter Warenkorbwert der bestätigten Leads.
`sales_canceled_turnover`	Gesamter Warenkorbwert stornierten Sales.
`sales_open_turnover`	Gesamter Warenkorbwert der offenen Sales.
`sales_confirmed_turnover`	Gesamter Warenkorbwert der bestätigten Sales.

Touchpoints / Customerjourney API - Dokumentation

Die Customerjourney Orders API dient dem Export und der Analyse von Customer-Journey-Daten, die über das Partnerprogramm-System erfasst wurden. Sie ermöglicht es, detaillierte Informationen zu den einzelnen Kontaktpunkten (Actions) einer Bestellung strukturiert als CSV-Datei abzurufen und die Pfade der Nutzer bis zur Conversion präzise nachzuvollziehen.

Die API kann im User-Interface unter dem Menüpunkt **Daten-API** verwendet werden.

API-Aufruf

`https://DOMAIN/api/ACCESS-TOKEN/admin/ID/get-customerjourney_orders.csv`

Wert	Beschreibung
DOMAIN	System-Partnerprogramm Domain
ACCESS-TOKEN	Access Token des Users
ID	User-ID
.csv	Art des Downloads (Dateiformat)

Customerjourney-API Filterung

Für die Customerjourney API können verschiedene Parameter als Filter verwendet werden. Nachfolgend eine Übersicht der verfügbaren Parameter:

Parameter	Wert / Beispiel	Beschreibung
condition[period][from]	2026-02-04	Definiert das Startdatum des Zeitraums (Bestelldatum).

Parameter	Wert / Beispiel	Beschreibung
condition[period][to]	2026-02-04	Definiert das Enddatum des Zeitraums (Bestelldatum).
condition[dynamicdate]	`currentmonth`, `lastmonth`, `last10d`, `currentweek`	Definition eines dynamischen Zeitraums. **Hinweis:** Kann nicht zusammen mit `period[from]` oder `period[to]` verwendet werden.
condition[l:campaigns]	1	Filtert alle Daten, die einer bestimmten Kampagne zugeordnet sind. Mehrfachnennung möglich.
condition[l:projects]	1234	Alle Transaktionen, welche unter der entsprechenden Projekt-ID des Programms hinterlegt sind. Mehrfachnennung möglich.

Beispiele für gefilterte Aufrufe:

- **Fixer Zeitraum:** `condition[period][from]=2026-02-04&condition[period][to]=2026-02-04&condition[l:campaigns]=1`
- **Dynamisch:** `condition[dynamicdate]=currentweek&condition[l:projects]=1234`

Definition der Felder (Spalten)

Die heruntergeladenen Transaktionen enthalten folgende Spalten:

Feldname	Beschreibung
order_id	Interne ID der zugehörigen Transaktion / Bestellung
ordertoken	Bestellnummer der Transaktion aus dem Shopsystem
campaign_id	ID der Kampagne, welcher die Bestellung zugeordnet ist
order_timestamp	Zeitpunkt des Bestelleingangs (Format: YYYY-MM-DD HH:MM:SS+TZ)
trigger_id	ID des Triggers, der der Bestellung zugeordnet ist
project_id	ID des Projekts (Attribution Target)
admedia_id	ID des Werbemittels der Bestellung
action_timestamp	Zeitpunkt des Klicks oder der Einblendung (Touchpoint)
type	Art des Kontaktpunkts (z.B. click)
action_campaign_id	Kampagnen-ID des spezifischen Kontaktpunkts innerhalb der Journey
action_project_id	Projekt-ID des spezifischen Kontaktpunkts

Feldname	Beschreibung
action_admedia_id	Werbemittel-ID des spezifischen Kontaktpunkts
referrer	Die URL, von der der Nutzer zum Shop geleitet wurde (inkl. Tracking-Parametern)

Order Webservice

Create / easy.affiliate REST-API (Orders)

Einleitung

Um neue Orders automatisiert im easy.affiliate-System anzulegen, steht ein REST-konformer Webservice zur Verfügung. Über die HTTP-Methode **POST** können neue Transaktionen übermittelt werden. Jeder User erhält hierfür einen Authentifizierungs-Token und eine Login-ID, die im Frontend einsehbar sind.

Authentifizierung

Für den Zugriff sind folgende Header erforderlich:

Header	Beschreibung	Typ
Content-Type	Muss auf `application/json` gesetzt sein	String
X-Network-ID	Netzwerk-ID: meist `-1` für mandantenübergreifende API	Integer
X-Auth-Token	Dein API-Token (im Frontend sichtbar)	String
X-Auth-ID	Deine Login-ID (im Frontend sichtbar)	Integer

Endpunkte

- **Admin:** `https://SUBDOMAIN.de/ws/V6/admin/JSON/Orders`
- **Advertiser:** `https://SUBDOMAIN.de/ws/V6/advertiser/JSON/Orders`

□□ Beispiel (cURL)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "X-Network-ID: -1" \
  -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" \
  -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" \
  -d '[{
    "ordertoken": "testorder001",
    "campaign_id": 1,
    "trigger_id": 2,
    "status": 1,
    "turnover": 199.99,
    "trigger_tstamp": "1785315641"
  }]' \
  https://SUBDOMAIN.de/ws/V6/admin/JSON/Orders
```

□□ Body-Parameter

Parameter	Beschreibung	Datentyp	Pflichtfeld
ordertoken	Eindeutiger Identifier für die Order. Muss systemweit eindeutig sein. Wird kein Wert übergeben, generiert das System automatisch einen (`AUTOID_…`).	String	□ Ja
campaign_id	ID der Kampagne, der die Order zugeordnet wird.	Integer	□ Ja
trigger_id	ID des Triggers, der bei Nutzung von `ordertoken` zwingend angegeben werden muss. Alternativ per `trigger_title`.	Integer	□ Ja
status	Status der Order: `0` = offen, `1` = bestätigt, `2` = storniert. *(Bei der Anlage immer `0`.)*	Integer	□ Ja
turnover	Bestellwert in Kampagnenwährung. Komma und Punkt als Dezimaltrenner erlaubt.	Float	□ Ja

Parameter	Beschreibung	Datentyp	Pflichtfeld
commission	Provisionswert in Kampagnenwährung. Kann übergeben werden, falls selbst berechnet.	Float	Nein
cancel_reason	Angabe eines Stornogrundes bei Status = 2 (storniert).	String	Nein
currency	Bestellwährung als 3-stelliger ISO-Code (z. B. `EUR`, `CHF`). Default = Kampagnen-Einstellung, sonst `EUR`. Umrechnungskurs wird automatisch gesetzt.	String	Nein
descr	Freitext-Beschreibung der Order (max. 255 Zeichen).	String	Nein
trigger_tstamp	Bestellzeitpunkt als Unix-Timestamp. Fehlt der Wert, wird die aktuelle Zeit genutzt.	Integer	Nein
emid	Klick-/Action-ID aus dem Tracking (24-stellig). Dient der Klick-Zuordnung und kann die `campaign_id` auflösen.	String	Nein*
vc	Gutscheincode-Tracking: löst Kampagne/Werbemittel über die Admedia-Einstellungen auf.	String	Nein*
project_id	Direkte Zuordnung per Projekt (Publisher). Erzeugt eine Track-Action, sofern eine Connection zur Kampagne besteht.	Integer	Nein*
subid	Attributionswert (SubID), wird am Auftrag gespeichert (max. 500 Zeichen).	String	Nein
sub_status	Substatus der Order (frei belegbar).	String	Nein
get_parameters	Objekt `{ "key": "value", … }` mit beliebigen Zusatzfeldern (z. B. `voucher`, `coupon`).	Object	Nein

Parameter	Beschreibung	Datentyp	Pflichtfeld
(beliebiger Key)	Jeder weitere, nicht reservierte Parameter wird als Key/Value am Auftrag gespeichert (Key max. 25 Zeichen).	String	Nein

> * **Kampagnen-/Klick-Zuordnung:** `campaign_id` ist Pflicht, kann aber alternativ über `emid` oder `vc` aufgelöst werden. Für die Zuordnung zur ursprünglichen Klick-/View-Aktion sollte **eine** der Attributions-Methoden (`emid` oder `project_id`) mitgegeben werden.

> **Dublettenschutz:** Dieselbe Kombination aus `campaign_id` + `trigger_id` + `ordertoken` wird innerhalb abgelehnt (`ORDER_EXISTS` / `ORDER_REQUEST_FREEZED`).

□□ Hinweise

Wichtige Implementierungsdetails:

- **ordertoken:** Muss **eindeutig** sein – idealerweise eine Kombination aus Zeitstempel, Shop-ID oder externer Ordernummer.
- **trigger_id:** Wenn du diese nicht kennst, frage im Frontend deine Kampagnenkonfiguration ab.
- **Bulk-Aktionen:** Du kannst auch mehrere Orders in einem Array auf einmal anlegen.
- **Updates:** Wird eine Order mit gleichem `ordertoken` bereits im System gefunden, kann ein `PUT-Request` genutzt werden, um sie zu aktualisieren.

Update / easy.affiliate REST-API (Orders)

Mit diesem Aufruf ändern Sie bestehende Transaktionen (Orders): Status setzen, Umsatz korrigieren, Stornogrund hinterlegen. Einzeln oder viele auf einmal. Neue Transaktionen legt dieser Aufruf nie an, dafür gibt es Create Orders.

Den kompletten Ablauf eines Sales-Abgleichs, auch per Datei-Upload und SFTP, beschreibt Automatisierter Sales-Abgleich.

Endpunkte

Login	URL
Admin	`https://<Ihre-Domain>/ws/V6/admin/REST/Orders`
Advertiser	`https://<Ihre-Domain>/ws/V6/advertiser/REST/Orders`

Methode: `PUT`. Statt `REST` funktioniert auch `JSON` im Pfad, beide verhalten sich gleich.

Welche Transaktionen ein Login ändern darf:

- **Advertiser-Login:** nur Transaktionen der Kampagnen seines Advertisers.
- **Admin-Login mit Mandant:** nur Transaktionen der Kampagnen dieses Mandanten.
- **Admin-Login ohne Mandant:** alle Transaktionen der Installation.

Transaktionen außerhalb dieses Bereichs behandelt die API wie nicht vorhanden.

Header

Header	Wert
`Content-Type`	`application/json`

Header	Wert
`X-Auth-ID`	Login-ID des Benutzers
`X-Auth-Token`	Access-Token des Benutzers, zu finden unter **API → Daten-API**
`X-Network-ID`	ID des Mandanten. Bei nur einem Mandanten oder mandantenübergreifend `-1`.

Wo die Login-ID steht: Wo ist die User ID hinterlegt?

Beispiel

```
curl -X PUT "https://<Ihre-Domain>/ws/V6/admin/REST/Orders" \
  -H "Content-Type: application/json" \
  -H "X-Auth-ID: <LOGIN-ID>" \
  -H "X-Auth-Token: <ACCESS-TOKEN>" \
  -H "X-Network-ID: -1" \
  -d '[
    { "id": 12345, "status": 1 },
    { "id": 12346, "status": 2, "cancel_reason": "Retoure" },
    { "id": 12347, "status": 1, "turnover": 89.90 },
    { "campaign_id": 1, "ordertoken": "123a456b", "status": 2, "cancel_reason": "Storno" }
  ]'
```

Der Body ist eine Liste von Objekten, je Objekt eine Transaktion. Ein einzelnes Objekt ohne Liste geht auch.

Transaktion finden

Weg	Felder	Hinweis
Transaktions-ID (empfohlen)	`id`	eindeutig, mehr ist nicht nötig. Geben Sie zusätzlich `campaign_id` mit, muss sie zur Transaktion passen.

Weg	Felder	Hinweis
Bestellnummer	`campaign_id` + `ordertoken`	`campaign_id` ist hier Pflicht. Gibt es zur Bestellnummer mehrere Transaktionen in der Kampagne (z. B. Lead und Sale), ist die Zuordnung nicht eindeutig. Dann immer `id` verwenden.
Klick-ID	`emid` + `ordertoken`	Eine 24-stellige `emid` ersetzt die `campaign_id`, die Kampagne wird aus dem Klick ermittelt.

Felder

Feld	Bedeutung	Typ
`status`	`0` offen, `1` bestätigt, `2` storniert. Auch als Wort: `offen`, `bestätigt`, `storniert`, `open`, `confirmed`, `cancelled`. Andere Werte werden abgewiesen.	Integer
`turnover`	Netto-Umsatz in Kampagnenwährung, Punkt als Dezimaltrenner	Float
`original_turnover`	Umsatz in Fremdwährung. **Überschreibt** `turnover`, wenn gesetzt.	Float
`commission`	fester Provisionsbetrag. Nur mitgeben, wenn er bewusst von der Provisionsgruppe abweichen soll.	Float
`cancel_reason`	Stornogrund, Freitext	String
`status_change_date`	Datum der Statusänderung (ISO 8601). Ohne Angabe setzt das System es bei einem Statuswechsel selbst. Bestimmt auch das Abrechnungsdatum.	Datetime
`sub_status`	Substatus, frei belegbar	String
`description`	Beschreibung, höchstens 255 Zeichen	String
`order_currency`	Bestellwährung als ISO-Code, z. B. `EUR`	String
`subid_value`	SubID der Transaktion. Heißt hier `subid_value`, nicht `subid` wie beim Anlegen.	String

Feld	Bedeutung	Typ
`get_parameters`	Objekt `{ "key": "value" }`, wird je Transaktion ergänzt oder überschrieben	Object

Felder, die Sie nicht mitschicken, bleiben unverändert.

Provision

Ohne `commission` gilt: Bei prozentualer Provision rechnet das System die Provision aus dem neuen `turnover` neu. Feste Provisionen bleiben, wie sie sind. Der Wert `"commission": "autocalculate"` bewirkt dasselbe und ist nicht nötig. Ein Zahlenwert in `commission` wird genau so übernommen.

Nicht änderbar

Ausgezahlte Transaktionen (Status 3) sind abgerechnet und lassen sich nicht mehr ändern. Status 3 kann auch nicht per API gesetzt werden, er entsteht bei der Auszahlung.

Antwort

```
{
  "data": [
    { "id": 12345, "status": 1, "turnover": 49.9, "commission": 4.99, "cancel_reason": null,
"status_change_date": "2026-10-28T14:50:44+00:00", ... },
    { "id": 12346, "status": 2, "turnover": 19.99, "commission": 2, "cancel_reason":
"Retoure", ... },
    { "id": 12347, "status": 1, "turnover": 89.9, "commission": 8.99, ... }
  ]
}
```

- `data` enthält jede geänderte Transaktion mit ihrem neuen Stand, so wie sie jetzt im System steht.
- **Gleichen Sie die IDs in `data` mit den gesendeten ab.** Fehlt eine, wurde sie nicht geändert: nicht gefunden, bereits ausgezahlt, außerhalb Ihres Bereichs oder ungültiger Status. In einer Liste gibt es dafür keine eigene Fehlermeldung.
- Schicken Sie ein einzelnes Objekt ohne Liste, steht die Transaktion direkt in `data`. Kann sie nicht geändert werden, kommt `"error": true`, der Grund steht in `msg`, zum Beispiel `status 7 is not allowed, use 0 (open), 1 (confirmed) or 2 (cancelled)`.

Im Beispiel oben fehlt die vierte Transaktion in der Antwort, wenn es die Bestellnummer `123a456b` in Kampagne 1 nicht gibt.

Fehler

Antwort	Bedeutung
`"message": "Authentification failed."`	Login-ID oder Access-Token falsch, oder der Login ist gesperrt
`"error": true` bei einem einzelnen Objekt	Transaktion nicht gefunden, nicht änderbar oder ungültiger Status. Der Grund steht in `msg`.
Transaktion fehlt in `data`	siehe Antwort

Jede Änderung steht in der Historie der Transaktion mit Zeitpunkt und Benutzer. Ein eigener Benutzer für die Anbindung, etwa „Salesabgleich API“, macht dort sofort sichtbar, welche Änderung aus der Schnittstelle kam.

Create / easy.affiliate REST-API (Baskets)

□□ Einleitung

Um neue Orders **inklusive Warenkorb** automatisiert im easy.affiliate-System anzulegen, steht ein REST-konformer Webservice zur Verfügung. Über die HTTP-Methode `POST` wird pro Aufruf eine Bestellung mit ihren Warenkorb-Positionen übermittelt.

Die Bestellung durchläuft dabei dieselbe Logik wie ein getrackter Verkauf mit Warenkorb (`/trck/ebasket/`): Publisher-Zuordnung, Provision je Position (Regeln nach Artikel, Marke, Kategorie oder Default des Triggers), Bestellwert sowie serverseitige Tags und Postbacks der Kampagne. **Bestellwert, Provision und Status werden immer vom System berechnet** und können nicht übergeben werden.

Jeder User erhält hierfür einen Authentifizierungs-Token und eine Login-ID, die im Frontend einsehbar sind.

□□ Authentifizierung

Für den Zugriff sind folgende Header erforderlich:

Header	Beschreibung	Typ
`Content-Type`	Muss auf `application/json` gesetzt sein	String
`X-Network-ID`	Netzwerk-ID des Mandanten (muss > 0 sein, **nicht** `-1`). Die Kampagne muss zu diesem Mandanten gehören.	Integer
`X-Auth-Token`	Dein API-Token (im Frontend sichtbar)	String
`X-Auth-ID`	Deine Login-ID (im Frontend sichtbar)	Integer

Berechtigung: Nur Admin-Logins mit Lese- und Schreibrecht auf die Bestellungen (`administration_orders`). Der Login muss zum Mandanten aus `X-Network-ID` gehören.

□□ Endpunkte

- **Admin:** `https://SUBDOMAIN.de/ws/V6/admin/REST/Baskets`

□□ Beispiel (cURL)

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "X-Network-ID: 1" \
  -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" \
  -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" \
  -d '{
    "campaign_id": 123,
    "trigger_id": 5,
    "ordertoken": "E-10045",
    "emid": "5f0123456789abcdef012345",
    "basket": [
      {"article_number": "A-1", "productname": "Hemd", "category": "Herren", "amount": 2,
"price": 19.90, "additional": {"brand": "X"}},
      {"article_number": "B-7", "amount": 1, "price": 5}
    ]
  }' \
  https://SUBDOMAIN.de/ws/V6/admin/REST/Baskets
```

□□ Body-Parameter (Order)

Parameter	Beschreibung	Datentyp	Pflichtfeld
`campaign_id`	ID der Kampagne, der die Order zugeordnet wird.	Integer	□ Ja
`trigger_id`	ID eines Triggers dieser Kampagne (z. B. Sale-Trigger).	Integer	□ Ja

Parameter	Beschreibung	Datentyp	Pflichtfeld
`ordertoken`	Eindeutiger Identifier für die Order (eigene Bestellnummer). 1–255 druckbare ASCII-Zeichen, ohne Leerzeichen und ohne `(` `)` `;`.	String	☐ Ja
`basket`	Liste der Warenkorb-Positionen (min. 1, max. 200), siehe unten.	Array	☐ Ja
`emid`	Klick-/Action-ID aus dem Tracking (24-stellig). Muss zu einem Klick dieser Kampagne gehören.	String	Nein*
`project_id`	Direkte Zuordnung per Projekt (Publisher). Das Projekt muss mit der Kampagne verbunden sein.	Integer	Nein*
`currency`	Bestellwährung als 3-stelliger ISO-Code (z. B. EUR, CHF). Default = Kampagnen-Einstellung, sonst EUR. Unbekannte Währungen werden abgelehnt.	String	Nein
`trigger_tstamp`	Bestellzeitpunkt als Unix-Timestamp (Sekunden). Fehlt der Wert, wird die aktuelle Zeit genutzt.	Integer	Nein
`descr`	Freitext-Beschreibung der Order (max. 255 Zeichen).	String	Nein

* **Publisher-Zuordnung:** Es darf **entweder** `emid` **oder** `project_id` übergeben werden, nicht beides. Beide werden vorab geprüft: Eine unbekannte/abgelaufene `emid`, eine `emid` einer anderen Kampagne oder ein nicht verbundenes Projekt wird abgelehnt, statt die Order still dem Systempartner zuzuordnen. Ohne beide Angaben entscheidet die Kampagne, ob die Order an den Systempartner geht (Antwort enthält dann `notice: NO_SESSION_FOUND`).

Nicht erlaubt: `turnover`, `commission`, `status` und alle weiteren, nicht aufgeführten Felder werden mit einem Fehler abgelehnt.

☐☐ Body-Parameter (Warenkorb-Position in `basket`)

Parameter	Beschreibung	Datentyp	Pflichtfeld
`article_number`	Artikelnummer / SKU (1–255 Zeichen). Provisionsregeln nach Artikel greifen darauf.	String	□ Ja
`amount`	Menge, ganze Zahl ≥ 1.	Integer	□ Ja
`price`	Preis **einer** Einheit, ≥ 0, **nur Punkt** als Dezimaltrenner (z. B. `19.90`), max. 4 Nachkommastellen. Bei Brutto-Tracking der Kampagne wird die MwSt. wie beim Tracking abgezogen.	Float	□ Ja
`productname`	Produktname (max. 255 Zeichen).	String	Nein
`category`	Kategorie für Provisionsregeln nach Kategorie. Leer = Artikelnummer.	String	Nein
`additional`	Objekt `{ "key": "value", … }` mit max. 50 flachen Zusatzfeldern, wird an der Position gespeichert. `additional.brand` wird für Provisionsregeln nach Marke genutzt. Keys: 1–50 Zeichen aus `A-Z a-z 0-9 _ -`; reservierte Keys (z. B. `currency`, `emid`, `campaign_id`, `commission`, `turnover`) sind nicht erlaubt.	Object	Nein

Der Bestellwert der Order ergibt sich aus der Summe `price × amount` aller Positionen, die Provision aus der Summe der Positions-Provisionen.

Dublettenschutz: Dieselbe Kombination aus `campaign_id` + `trigger_id` + `ordertoken` wird abgelehnt (`ORDER_EXISTS`), die Antwort enthält die `order_id` der vorhandenen Order. Ein erneuter Aufruf, z. B. nach einem Timeout, ist dadurch gefahrlos möglich.

□□ Antwort

Erfolg:

```
{
  "order_inserted": true,
  "order_id": 4711815,
  "emid": "5f0123456789abcdef012345",
  "sessionid": "5f0123456789abcdef000001",
  "basket": [
    {"id": 981234, "productid": "A-1", "productname": "Hemd"},
    {"id": 981235, "productid": "B-7", "productname": ""}
  ]
}
```

Fehler: `order_inserted: false`, `order_id: -1`, dazu `error` und `error_description`. Alle Eingabefehler werden gesammelt in einer Antwort geliefert.

Error	Bedeutung
`INVALID_REQUEST`	Eingabe ungültig (Pflichtfeld fehlt, Format falsch, unbekanntes Feld, Trigger gehört nicht zur Kampagne, `emid` / `project_id` passt nicht, Währung unbekannt). Details in `error_description`.
`ORDER_EXISTS`	Order mit diesem `ordertoken` existiert bereits für Kampagne und Trigger; `order_id` enthält die vorhandene Order.
`TRACKING_UNAVAILABLE`	Das Tracking hat nicht geantwortet, die Order wurde vermutlich nicht angelegt. Erneut senden (Dublettenschutz greift).
sonstige	Ablehnung durch die Tracking-Logik bzw. Kampagnen-Einstellungen.

□□ Hinweise

Wichtige Implementierungsdetails:

- **Ein Aufruf = eine Order:** Der Body ist ein einzelnes JSON-Objekt, kein Array. Mehrere Orders werden mit mehreren Aufrufen angelegt.
- **ordertoken:** Muss je Kampagne und Trigger eindeutig sein – idealerweise die externe Bestellnummer des Shops.
- **trigger_id:** Wenn du diese nicht kennst, frage im Frontend deine Kampagnenkonfiguration ab.
- **Preise:** Immer mit Punkt als Dezimaltrenner übergeben (`12.50`, nicht `12,50`).

Update / easy.affiliate REST-API (Baskets)

Einleitung

Um Warenkorb-Positionen von Orders automatisiert im easy.affiliate-System bearbeiten zu können, stellt die easy Marketing GmbH eine Webservice-API zur Verfügung. Diese ermöglicht es, einzelne Positionen bestehender Orders im Nachgang zu bearbeiten – z. B. bei Teilretouren, Teilstornos oder Preiskorrekturen.

Nach jeder Änderung werden **Bestellwert, Provision und Status der Order automatisch aus ihren Positionen neu berechnet**.

Jedem User stehen hierfür ein Authentifizierungs-Token und eine Login_id zur Verfügung, die über das Frontend abgerufen werden können.

- **User-ID:** Wo ist die User-ID hinterlegt? (Siehe Frontend-Profil)
- **Access-Token:** Wo ist der Access-Token hinterlegt? (Siehe API-Einstellungen)

Endpunkte

Zielgruppe	URL
Admin	`https://SUBDOMAIN.de/ws/V6/admin/REST/Baskets`

Beispiele

Beispiel mit cURL

```
curl -X PUT \
  -H "Content-Type: application/json" \
  -H "X-Network-ID: -1" \
  -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" \
  -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" \
  -d '[{ "id": 981234, "amount": 1 },{ "id": 981235, "status": 2 }]' \
  https://SUBDOMAIN.de/ws/V6/admin/REST/Baskets
```

Codebeispiel

Mit der folgenden Methode können eine oder mehrere Warenkorb-Positionen geupdatet werden. Ein Beispielaufruf sieht folgendermaßen aus:

Headers

Variable	Wert / Bedeutung
Content-Type	application/json
X-Network-ID	-1
X-Auth-Token	ADMIN_APIUSER_TOKEN
X-Auth-ID	ADMIN_APIUSER_LOGIN_ID

Body

```
[
    {
        "id": 981234,
        "amount": 1
    },
    {
        "basket_item_id": 981235,
        "status": 2
    },
    {
        "id": 981236,
        "price": 17.50,
        "additional": { "retoure_grund": "zu klein" }
    }
]
```

Für eine einzelne Position kann statt der Liste auch direkt ein Objekt gesendet werden, z. B. `{ "id": 981234, "amount": 1 }`.

Variablenerläuterung

Headers

Variable	Bedeutung	Datentyp
Content-Type	Der Content-Type des Requests	String
X-NETWORKID	Hier wird die ID des Mandanten eingetragen. Wenn nur ein Mandant vorhanden ist oder mandantenübergreifend gearbeitet wird, muss der Wert “-1” eingetragen werden.	Integer
X-AUTH-TOKEN	Hier wird der API-Authentifizierungs-Token des Admin Nutzers hinterlegt.	String
X-AUTH-ID	Hier wird die ID des Admin Nutzers hinterlegt.	Integer

Body

Parameter	Beschreibung	Datentyp	Pflichtfeld
`id`	ID der Warenkorb-Position. Steht in der Antwort der Anlage (`basket[].id`) bzw. im Order-Export.	Integer	✅ Ja (oder `basket_item_id`)
`basket_item_id`	Alternative Schreibweise für `id`.	Integer	✅ Ja (oder `id`)
`amount`	Neue Menge der Position.	Integer	Nein
`price`	Neuer Preis **einer** Einheit, Punkt als Dezimaltrenner.	Float	Nein
`status`	Status der Position: 0 = offen, 1 = bestätigt, 2 = storniert, 3 = ausgezahlt.	Integer	Nein

Parameter	Beschreibung	Datentyp	Pflichtfeld
`commission`	Provisionswert der Position in Kampagnenwährung. Wird er nicht übergeben, wird die Provision neu berechnet (Preis × Menge × Provisionssatz der Position).	Float	Nein
`additional`	Objekt `{ "key": "value", … }`. Die übergebenen Keys werden in die bestehenden Zusatzfelder der Position eingefügt bzw. überschrieben; nicht übergebene Keys bleiben erhalten.	Object	Nein

*) Hinweis: Alle Werte außer `additional` müssen numerisch sein. Weitere Felder (z. B. `productname`, `article_number`, `campaign_id`) werden ignoriert und können über diesen Endpunkt nicht geändert werden.

□□ Auswirkung auf die Order

Nach jeder Änderung wird die zugehörige Order aus **allen** ihren Positionen neu berechnet:

Wert der Order	Berechnung
`turnover`	Summe aus `price × amount` aller Positionen mit Status 0, 1 oder 3 (stornierte Positionen zählen nicht)
`commission`	Summe der Provisionen aller Positionen mit Status 0, 1 oder 3
`status`	Niedrigster Status aller Positionen (eine offene Position hält die Order offen; sind alle Positionen storniert, ist die Order storniert)

Die Neuberechnung erfolgt nur, solange die Order den Status offen, bestätigt oder storniert hat. **Bereits ausgezahlte Orders (Status 3) werden nicht mehr verändert.**

□□ Antwort

Die Antwort enthält die gespeicherten Positionen mit ihren aktuellen Werten (bei einer Liste als Array, bei einem einzelnen Objekt als Objekt).

Bei einer Liste werden Positionen, die nicht gefunden wurden oder ungültige Werte enthalten, übersprungen und fehlen in der Antwort – die übrigen Positionen werden trotzdem gespeichert. Bei einem einzelnen Objekt antwortet der Webservice in diesem Fall mit einem Fehler.

□□ Hinweise

- **Teilretoure:** Menge reduzieren (`amount`) oder die betroffene Position stornieren (`status: 2`) – Bestellwert und Provision der Order passen sich automatisch an.
- **Komplettstorno:** Alle Positionen auf `status: 2` setzen, die Order wird dann ebenfalls storniert.
- **Positions-IDs:** Kommen aus der Antwort von *Create / easy.affiliate REST-API (Baskets)* oder per `GET` auf denselben Endpunkt.

Publisher Webservice

Verwaltung von Projekten über die V6 REST-API

Einleitung

Die V6 REST-API bietet Admins die Möglichkeit, Projekte effizient zu verwalten. Mit dem `/Projects`-Endpoint können Projekte erstellt, ausgelesen, bearbeitet und gelöscht werden. Zusätzlich können Projekte anhand verschiedener Parameter gefiltert werden.

Endpoint

Für Admins: `https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects`

Parameter & Filter

Erforderliche Parameter (bei POST und PUT)

Parameter	Typ	Beschreibung
`title`	String	Titel des Projekts (Pflichtfeld)
`status`	Integer	Status des Projekts (Pflichtfeld)
`url`	String	URL des Projekts (Pflichtfeld)

Filteroptionen (bei GET)

Es kann nach allen Spalten der Tabelle `publisher.projects` gefiltert werden, z. B.:

- `id`
- `status`
- `publisher_id`
- `projecttype`
- `title`
- `url`
- `channel_id`
- `hidden`

Beispiel für GET mit Filtern:

```
curl -X GET -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token:
ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID"
"https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects?status=1&publisher_id=10"
```

CRUD-Operationen

GET: Abrufen von Projekten

Abrufen einer Liste von Projekten oder eines spezifischen Projekts anhand der `id`.

```
curl -X GET -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token:
ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID"
"https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects?id=5"
```

POST: Erstellen eines neuen Projekts

```
curl -X POST -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token:
ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" -d '{
  "title": "Neues Projekt",
  "status": 1,
  "url": "https://example.com"
}' "https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects"
```

PUT: Aktualisieren eines bestehenden Projekts

```
curl -X PUT -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" -d '{
  "id": 5,
  "title": "Geändertes Projekt",
  "status": 2,
  "url": "https://updated-example.com"
}' "https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects"
```

DELETE: Löschen eines Projekts

Löschen eines Projekts anhand der `id`.

```
curl -X DELETE -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" "https://SUBDOMAIN.de/ws/V6/admin/JSON/Projects?id=5"
```

Antworten (Responses)

Erfolgreiche Antwort

```
{
  "id": 5,
  "status": 1,
  "publisher_id": 10,
  "projecttype": "page",
  "title": "Projektbeispiel",
  "description": "Beschreibung des Projekts",
  "url": "https://example.com",
  "channel_id": 2,
  "project_identifier_string": "abc123",
```

```
  "reach": "1000",
  "statistic_type": "internal",
  "external_sources_entity_id": "-1",
  "trackingtemplate": "template",
  "kpi_whitelist": "kpi1,kpi2",
  "hidden": false,
  "login_type": "pub",
  "exclude_from_salary": 0,
  "default_project": false,
  "insert_timestamp": "2025-01-27T12:00:00"
}
```


Fehlerhafte Anfrage


```
{
  "error": "Missing required field: title"
}
```


Hinweise zur Fehlerbehandlung

- Bei fehlenden Pflichtfeldern (`title`, `status`, `url`) wird ein entsprechender Fehler zurückgegeben.
- Ungültige oder nicht vorhandene `id` bei GET, PUT oder DELETE resultieren in einer leeren Antwort oder einer Fehlermeldung.

Publisher-Verwaltung über die V6 REST-API

Authentifizierung

Erforderliche Header für alle Anfragen:

Header	Beschreibung
`X-Network-ID`	Netzwerk-ID, z.B. `-1` für global
`X-Auth-Token`	API-User-Token
`X-Auth-ID`	API-User-Login-ID
`Content-Type`	`application/json`

Endpunkt

Base URL: `https://SUBDOMAIN.de/ws/V6/admin/REST/Publisher`

Es werden die üblichen HTTP-Methoden verwendet: `POST` zum Anlegen, `PUT` zum Aktualisieren, `GET` zum Abrufen.

CREATE Operation (POST)

Methode & Pfad: `POST /ws/V6/admin/REST/Publisher`

Erforderliche Parameter

Parameter	Typ	Beschreibung

`email`	String	Max. 255 Zeichen
`salutation`	String	z.B. `mr` / `mrs`
`prename`	String	Max. 255 Zeichen
`surname`	String	Max. 255 Zeichen

Optionale Parameter

`company`, `street`, `zip`, `city`, `telephone`, `country` (ISO 3166 ALPHA-3), `billing_sepa_owner`, `billing_sepa_iban`, `billing_sepa_bic`, `companytype` (z.B. `un` / `priv`), `language_interface` (Default: `DEU`), `billing_mode` (Default: `1`), `billing_media` (Default: `1`), `country_billing` (Default: `DEU`), `country_publisher` (Default: `DEU`), `billing_limit` (Default: `25`), `dialog_email` (Default: Wert von `email`)

> Für `salutation`, `companytype` und `country` gibt es keine feste Werteliste, die serverseitig erzwungen wird — bitte trotzdem an die üblichen Werte halten, damit nachgelagerte Auswertungen (Rechnungsstellung, Sprachwahl etc.) korrekt funktionieren.

```
curl -X POST -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token:
ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" -d '{
  "email": "max.mustermann@example.com",
  "salutation": "mr",
  "prename": "Max",
  "surname": "Mustermann",
  "company": "Musterfirma",
  "street": "Musterstraße 1",
  "zip": "12345",
  "city": "Musterstadt",
  "country": "DEU",
  "billing_sepa_owner": "Max Mustermann",
  "billing_sepa_iban": "DE12345678901234567890",
  "billing_sepa_bic": "GENODEF1M01",
  "language_interface": "DEU",
  "billing_mode": 1,
  "billing_media": 1,
  "country_billing": "DEU",
```

```
  "country_publisher": "DEU",
  "billing_limit": 50
}' "https://SUBDOMAIN.de/ws/V6/admin/REST/Publisher"
```

Response

```
{
  "data": {
    "id": 123,
    "name": "Max Mustermann",
    "email": "max.mustermann@example.com"
  },
  "log": "no logmessage provided",
  "sessionId": "",
  "session": null
}
```

Die eigentlichen Daten stehen im `data`-Feld.

UPDATE Operation (PUT)

Methode & Pfad: `PUT /ws/V6/admin/REST/Publisher/{id}`

Ein bestehender Publisher wird über seine ID im URL-Pfad aktualisiert.

Aktualisierbare Felder Es können dieselben Felder übergeben werden, die auch beim Anlegen verwendet werden, z. B. `status`, `company`, `street`, `zip`, `city`, `telephone`, `country`, Billing-Felder oder Sprach-/Abrechnungswerte.

Welche Felder konkret geändert werden, hängt davon ab, welche Werte im Request-Body mitgegeben werden.

```
curl -X PUT -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token:
ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" -d '{
  "status": 1
}' "https://SUBDOMAIN.de/ws/V6/admin/REST/Publisher/123"
```

Response

```
{
  "data": {
    "id": 123
  },
  "log": "no logmessage provided",
  "sessionId": "",
  "session": null
}
```

Es werden nur die Felder geändert, die im Request übergeben werden.

READ Operation (GET)

Methode & Pfad: `GET /ws/V6/admin/REST/Publisher` bzw. `GET /ws/V6/admin/REST/Publisher/{id}`

Einzelnen Publisher abrufen

`GET /ws/V6/admin/REST/Publisher/{id}` — `id` ist Teil des URL-Pfads, keine Query-Parameter.

Liste abrufen (mit Filtern)

Filter werden als Query-Parameter im Format `params[feld]=wert` übergeben, z.B.:

```
GET /ws/V6/admin/REST/Publisher?params[status]=1
```

Filterbar sind sowohl die Basis-Felder des Publisher-Accounts (`id`, `status`, `type`) als auch alle Felder aus dem Registrierungsprozess (`email`, `prename`, `surname`, `company`, `street`, `zip`, `city`, `telephone`, `country`, Billing-Felder etc.). Mehrere Filter zusammen werden UND-verknüpft, z.B.:

```
GET /ws/V6/admin/REST/Publisher?params[status]=5&params[prename]=Max
```

Filter auf Registrierungsfelder sind exakte Treffer (kein Teilstring-/Wildcard-Match).

Status-Werte

Bekannte Werte für `status`:

Status	Bedeutung
`1`	Bestätigter, aktiver Publisher
`5`	Neu registriert, noch nicht freigegeben

(Für eine vollständige Übersicht aller möglichen Status-Werte bitte im Admin-Interface unter der Publisher-Verwaltung nachsehen.)

Response-Format der Liste

```
{
  "data": {
    "123": {
      "type": "pub",
      "id": 123,
      "status": 5,
      "settings": {
        "email": "max.mustermann@example.com",
        "prename": "Max",
        "surname": "Mustermann",
        "company": "",
        "street": "Musterstraße 1",
        "zip": "12345",
        "city": "Musterstadt",
        "country": "DEU",
        "...": "weitere im Registrierungsprozess gespeicherte Felder"
      }
    }
  },
  "log": "no logmessage provided",
  "sessionId": "",
  "session": null
}
```

Das `settings`-Objekt enthält alle für den Publisher gespeicherten Registrierungsdaten — welche Felder konkret enthalten sind, hängt davon ab, was beim jeweiligen Publisher tatsächlich hinterlegt wurde (kein festes Schema).

> **Bekannte Einschränkung:** Informationen zur Werbefläche (Werbefläche/Werbeflächen-URL), die im Registrierungsprozess ebenfalls abgefragt werden, sind über diese API aktuell **nicht** abrufbar. Das wird als Erweiterung nachgezogen.

Beispiele

- Einzelner Publisher: `GET /ws/V6/admin/REST/Publisher/123`
- Neu registrierte, noch nicht freigegebene Publisher: `GET /ws/V6/admin/REST/Publisher?params[status]=5`
- Publisher per E-Mail suchen: `GET /ws/V6/admin/REST/Publisher?params[email]=max.mustermann@example.com`

Fehlerbehandlung

Fehler werden in folgendem Format zurückgegeben:

```
{
  "error": true,
  "msg": "<Fehlerbeschreibung>",
  "type": "Ws\\Error",
  "data": []
}
```

Der HTTP-Statuscode ist bei Fehlern durchgängig `500`.

Fehler	Beispiel-`msg`
Fehlendes Pflichtfeld bei CREATE	`internal server error. Details: Missing required field: email`
Publisher nicht gefunden	`internal server error. Details: Model not found.`
Filter ohne Treffer	kein Fehler — liefert `"data": []`

Externe API anbinden

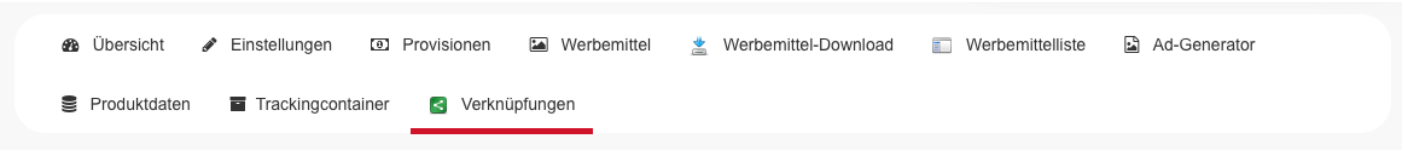
Es ist möglich für jedes externe Programm eine API für das System einzubinden. Die APIs werden durch den Support freigeschaltet. Hierfür muss bitte eine E-Mail an die Support@easy-m.de geschrieben werden. Anschließend wird überprüft, ob die API zur Verfügung steht oder angebunden werden kann.

Vorbereitung

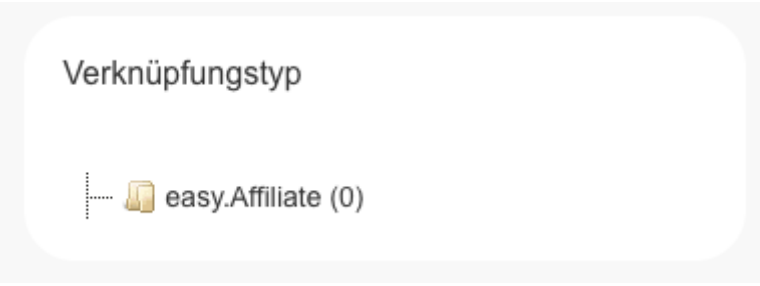
Um eine API für die Kampagne zu hinterlegen, muss zuerst die Kampagne angelegt werden.

Durchführung

Um eine API für eine Kampagne zu hinterlegen, muss auf das Kampagnen-Kontrollzentrum gewechselt werden. Anschließend muss zu dem Punkt “Verknüpfungen” navigiert werden:



Dort werden alle für die Kampagne verfügbaren APIs angezeigt:



Abschließend muss auf den Button “Neue Verknüpfung” geklickt werden, um die Daten, welche für die API benötigt werden, einzugeben.

Externe API Zugänge

Um die API des externen Dienstleisters anzusprechen werden entsprechende Zugangsdaten benötigt. Wo diese Daten in den einzelnen Systemen des Dienstleisters zu finden sind, wird in den einzelnen Anleitungen erläutert.

Werbemittel API

Admedia API (V6) - POST (Create)

Einleitung

Um Werbemittel nicht manuell, sondern durch einen automatisierten Prozess im easy.affiliate-System zu erstellen, stellt die easy Marketing GmbH eine Webservice-API bereit. Hierfür steht jedem User ein Authentifizierungs-Token und eine Login_id zur Verfügung. Der Authentifizierungs-Token sowie die Login_id werden dem User über das Frontend bereitgestellt.

User-ID: Wo ist die User-ID hinterlegt?

Access-Token: Wo ist der Access-Token hinterlegt?

Endpoint für Admin:

https://SUBDOMAIN.de/ws/V6/admin/JSON/Admedia

Endpoint für Advertiser:

https://SUBDOMAIN.de/ws/V6/advertiser/JSON/Admedia

Beispiel mit cURL

```
curl -X POST -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" -d "$EXAMPLE_JSON_DATA" <https://DOMAIN/ws/V6/admin/JSON/Admedia>
```

Codebeispiel

Mit der folgenden Methode können Werbemittel erstellt werden. Ein Beispielaufruf sieht folgendermaßen aus:

Headers:

```
Content-Type: application/json
X-Network-ID: -1
X-Auth-Token: ADMIN_APIUSER_TOKEN
X-Auth-ID: ADMIN_APIUSER_LOGIN_ID
```

Body:

```
{
    "campaign_id": 1,
    "project_ids": [50012],
    "title": "Werbemittel Titel",
    "deeplink": "https://example.de/link",
    "category_id": 5,
    "mediatype": "text",
    "linktext": "Klicken Sie hier",
    "status": 1
}
```

Variablenerläuterung

Headers

Variable	Bedeutung	Datentyp
Content-Type	Der Content-Type des Requests	String
X-NETWORKID	ID des Mandanten oder "-1"	Integer
X-AUTH-TOKEN	API-Authentifizierungs-Token des Admin Nutzers	String
X-AUTH-ID	ID des Admin Nutzers	Integer

Body

Parameter	Erklärung	Datentyp	Pflichtfeld

campaign_id	ID der Kampagne	Integer	Ja
project_ids	Whitelist von Projekt-IDs	JSON-ARRAY	Nein
title	Titel des Werbemittels	String	Ja
deeplink	Deeplink des Werbemittels	String	Ja
category_id	ID der Kategorie	Integer	Ja
click_tracking_getparameter	Click Tracking Parameter	String	Nein
valid_from	Gültig ab Datum	String	Nein
valid_until	Gültig bis Datum	String	Nein
mediatype	Typ des Werbemittels (z.B. text, banner)	String	Ja
linktext	Linktext (nur bei mediatype = text)	String	Ja, wenn Typ “text”
imageurl	Bild-URL (nur bei mediatype = banner)	String	Ja, wenn Typ “banner”
size_width	Breite des Banners	String	Ja, wenn Typ “banner”
size_height	Höhe des Banners	String	Ja, wenn Typ “banner”
status	Status des Werbemittels	Integer	Ja

Response

Nach erfolgreicher Erstellung gibt die API folgende Informationen als JSON zurück:

- Werbemittel-ID: Integer
- Werbemittel-Titel: String
- Werbemittel-Kategorie: String
- View-Link: String
- Click-Link: String
- View-Link (Shortener): String
- Click-Link (Shortener): String
- Adcode: String

Admedia API (V6) - GET (Read)

Abfrage generierter Werbemittel über die V6 REST-API

Um eine einfache und effiziente Verwaltung der generierten Werbemittel zu gewährleisten, stellt die V6 REST-API eine Abfragefunktion zur Verfügung. Mit dieser Funktion können Nutzer relevante Daten zu den Werbemitteln und den zugehörigen Trackinglinks abrufen.

Endpoint

Für Admins: `<https://SUBDOMAIN.de/ws/V6/admin/JSON/Admedia>`

Erforderliche Parameter:

- `campaign_id`: ID der betreffenden Kampagne (Pflichtfeld)
- `project_id`: ID des betreffenden Projekts (Optional)

Beispielaufruf mit cURL

```
curl -X GET -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" <https://DOMAIN/ws/V6/admin/JSON/Admedia?params[campaign_id]=1&params[project_id]=50001>
```

Erwartete Antwort

Nach erfolgreichem Aufruf gibt die API die folgenden Informationen zu den Werbemitteln im JSON-Format zurück:

- **id**: ID des Werbemittels (Integer)
- **title**: Titel des Werbemittels (String)
- **campaign_id**: ID der zugeordneten Kampagne (Integer)
- **project_id**: ID des zugeordneten Projekts (Integer, falls angegeben)
- **clicklink**: Clicklink des Werbemittels (String)
- **viewlink**: Viewlink des Werbemittels (String)
- **adcode**: Javascriptcode/HTML-Code des Trackinglinks (String)

Fehlerbehandlung

Bei fehlendem oder ungültigem `campaign_id` wird ein entsprechender Fehler zurückgegeben.

Bei nicht vorhandenen Daten für die angegebenen Parameter wird ein leerer Datensatz zurückgegeben.

Basket API

Basket Update API

□□ Einleitung

Zur automatischen Bearbeitung von Warenkörben, bieten wir eine Webservice-API an. Diese ermöglicht die nachträgliche Anpassung bereits existierender Warenkörbe. Jeder Nutzer erhält für den Zugriff einen Access-Token und eine User-ID, die über das Frontend bereitgestellt werden.

□□ Authentifizierung

Für den Zugriff sind folgende Header erforderlich:

Header	Beschreibung	Typ
Content-Type	Der Content-Type des Requests	String
X-Network-ID	Netzwerk-ID: meist `-1` für mandantenübergreifende API	Integer
X-Auth-Token	Dein Access-Token (im Frontend sichtbar)	String
X-Auth-ID	Deine User-ID (im Frontend sichtbar)	Integer

□□ Endpunkte

- **Admin:** `https://SUBDOMAIN.de/ws/V6/admin/JSON/Baskets`
- **Advertiser:** `https://SUBDOMAIN.de/ws/V6/advertiser/JSON/Baskets`

□□ Beispiel (cURL)

```
curl -X PUT -H "Content-Type: application/json" -H "X-Network-ID: -1" -H "X-Auth-Token: AUTH-TOKEN" -H "X-Auth-ID: USER-ID" -d '[{"basket_item_id": 76041, "amount": 2, "status": "0", "additional": {"variable1": "text", "variable2": 1}}]' https://SUB.DOMAIN.DE/ws/V6/admin/JSON/Baskets
```

□□ Body-Parameter

Parameter	Beschreibung	Datentyp	Pflichtfeld
basket_item_id	Eindeutige ID des Warenkorbelements	Integer	□ Ja
amount	Anzahl der Warenkorbelemente	Integer	Nein
price	Nettopreis des Warenkorbelements	Float	□ Nein
status	Status der Order: `0` = offen, `1` = bestätigt, `2` = storniert	Integer	□ Ja
turnover	Bestellwert in Kampagnenwährung	Float	□ Ja
additional	Zusätzliche Informationen zum Basket-Item	JSON	Nein

Beispiel Request

Headers

```
Content-Type: application/json
X-Network-ID: -1
X-Auth-Token: ACCESS-TOKEN
X-Auth-ID: USER-ID
```

Body

```
[
    {
            "basket_item_id": 76041,
            "amount": 1,
```

```
"status": "1",
"additional": {
    "variable1": "text",
    "variable2": 1
}
```

Conversion Trace API

Diese API liefert Einblick in einzelne Conversion-Tracking-Vorgänge ("Traces") – jeder eingehende Tracking-Aufruf (Bestellabschluss) wird mit allen Details protokolliert, egal ob er erfolgreich einer Kampagne zugeordnet werden konnte oder nicht. Diese Doku richtet sich an alle, die selbst gegen die API sprechen wollen (Scripting, eigene Auswertungen, Support-Tools).

Endpunkt & Request-Format

URL: `POST https://SUBDOMAIN/ws/admin/JSON/`

Content-Type: `application/x-www-form-urlencoded`

Der eigentliche Request steckt als JSON-String im Formularfeld `REQUEST`, dazu kommt `NETWORKID` als eigenes Feld:

Feld	Beschreibung
`NETWORKID`	Ihre Netzwerk-ID
`REQUEST`	URL-encodeter JSON-String, siehe unten

Aufbau von `REQUEST`

```
{
  "function": { "class": "orders", "method": "getOrderTraces" },
  "params": [
    { "limit": 50, "offset": 0, "sort": "timestamp", "sort_dir": "DESC" },
    { "id": "LOGIN_ID", "type": "adm", "network_id": "NETWORK_ID" }
  ],
  "token": "IHR_API_TOKEN",
  "login_id": "LOGIN_ID"
}
```

Key	Beschreibung
`function.class`	Immer `orders` für alle hier beschriebenen Methoden.

Key	Beschreibung
`function.method`	Name der aufzurufenden Methode, siehe Methoden.
`params[0]`	Die eigentlichen Filter/Argumente der jeweiligen Methode.
`params[1]`	Ihr Login-Kontext: `id` (Ihre Login-ID), `type` (i.d.R. `adm`), `network_id` (Ihre Netzwerk-ID).
`token` / `login_id`	Ihre Zugangsdaten, siehe Authentifizierung. Alternativ als Header `X-Auth-Token` / `X-Auth-ID` statt im Body.

Response-Format

Jede Antwort – Erfolg wie Fehler – kommt in derselben Hülle:

```
{
  "data": { "...": "Rückgabewert der Methode" },
  "log": "no logmessage provided",
  "sessionId": "",
  "session": null
}
```

Die eigentlichen Daten stehen immer im `data`-Feld.

Bei Fehlern (HTTP 500):

```
{ "error": true, "msg": "...", "type": "Ws\\Error", "data": [] }
```

Authentifizierung

Jeder Aufruf braucht ein gültiges Zugangsdaten-Paar: `login_id` + `token` (im `REQUEST`-JSON, siehe oben) oder alternativ als Header `X-Auth-ID` / `X-Auth-Token`.

Ihren API-Token erhalten Sie über Ihre Ansprechperson, oder – falls Sie bereits über einen klassischen Login verfügen – über den Login-Call: `function.class: "user"`, `method: "login"` mit `identifier`/`password`/`usertype`. Die Antwort enthält Ihre `login_id` sowie einen `token`, den Sie danach für alle weiteren Aufrufe wiederverwenden.

Methoden

`orders.getOrderTraces`

Gefilterte, sortierte, paginierte Liste der Traces.

Filter (`params[0]`):

Parameter	Typ	Beschreibung
`limit`	int	Default `50`, maximal `200`
`offset`	int	Default `0`
`order_id`	int	Exakter Treffer auf eine Bestell-ID
`campaign_id`	int	Exakter Treffer auf eine Kampagnen-ID
`trigger_id`	int	Exakter Treffer auf eine Trigger-ID
`ordertoken`	string	Teilstring-Suche im Ordertoken
`result`	string	Exakter Treffer auf `result` (siehe Result-Werte)
`order_created`	`"yes"` / `"1"` oder `"no"` / `"0"`	Filtert danach, ob aus dem Trace eine Bestellung entstanden ist
`date_from`	Datum/Zeitstempel	Traces ab diesem Zeitpunkt
`date_to`	Datum/Zeitstempel	Traces bis zu diesem Zeitpunkt (Tagesgrenze inklusive)
`matching_method`	string	Exakter Treffer auf die Zuordnungsmethode (z.B. `VoucherCode`)
`ip_hash`	string	Exakter Treffer auf den (gehashten) IP-Wert
`freetext`	string, mehrzeilig	Durchsucht Ordertoken, IP-Hash, Ergebnis-Detail und Fehlergrund gleichzeitig, sowie die Bestell-ID (exakt). Jede Zeile ist eine eigene Suche – mehrere Zeilen = mehrere IDs/Tokens auf einmal suchen.
`sort`	string	Eine von `timestamp`, `order_id`, `campaign_id`, `result`, `matching_method`, `ordertoken` (Default `timestamp`)
`sort_dir`	`ASC` / `DESC`	Default `DESC`

Response (`data`):


```
{
  "traces": [
    {
      "id": 123,
      "order_id": 456,
      "campaign_id": 78,
      "trigger_id": 9,
      "ordertoken": "abc123",
      "timestamp": "2026-07-06T10:00:00+02:00",
      "user_agent": "...",
      "ip_hash": "...",
      "matching_method": "VoucherCode",
      "matching_cascade": [ { "method": "VoucherCode", "matched": true, "reason": "...",
"details": {}, "timestamp": "..." } ],
      "patches_applied": [ { "type": "pre", "name": "...", "match_string": "...", "applied":
true, "changes": {} } ],
      "request_params": { "...": "alle GET/POST-Parameter des Trackingaufrufs" },
      "duplicate_check": { "constraint_key": "...", "is_duplicate": false, "is_freezed":
false, "detail": "..." },
      "device_info": { "device": "...", "device_type": "...", "os": "...", "browser": "...",
"browser_version": "..." },
      "connection_info": { "ip": "...", "referer": "...", "cookies": "..." },
      "result": "order_created",
      "result_detail": "...",
      "error_reason": null,
      "tracking_type": "client",
      "campaign_title": "Kampagnenname",
      "trigger_title": "Triggername"
    }
  ],
  "total": 1337,
  "limit": 50,
  "offset": 0
}
```


Die verschachtelten Felder (`matching_cascade`, `patches_applied`, `request_params`, `duplicate_check`, `device_info`, `connection_info`) kommen als echtes JSON-Objekt/Array zurück, nicht als String.

`orders.getOrderTrace`

Alle Traces zu **einer** Bestell-ID (chronologisch, neueste zuerst).

Params: `{ "id": 456 }`

Response: Array von Trace-Objekten (gleiches Format wie oben, ohne `campaign_title` / `trigger_title`).

`orders.getOrderTraceResults`

Liefert verfügbare Filterwerte (Result, Match Method, Campaign) inkl. Trefferzahl – praktisch, um eigene Filter-Dropdowns zu befüllen.

Params: keine.

Response:

```
{
  "results": [ { "result": "order_created", "count": 900 }, ... ],
  "matching_methods": [ { "matching_method": "VoucherCode", "count": 300 }, ... ],
  "campaigns": [ { "campaign_id": 78, "campaign_title": "...", "count": 120 }, ... ]
}
```

`orders.getOrderTraceStats`

Liefert aggregierte Kennzahlen (stündlicher Verlauf + Zähler). Akzeptiert dieselben Filter wie `getOrderTraces`, mit Ausnahme von `order_id` / `ordertoken` / `trigger_id` / `ip_hash` / `sort` / `sort_dir`.

Filter (`params[0]`): `campaign_id`, `result`, `matching_method`, `order_created`, `freetext`, `date_from`, `date_to` (gleiche Bedeutung wie bei `getOrderTraces`).

Response:

```
{
  "chart": [ { "hour": "2026-07-06T09:00:00+00:00", "total": 42, "success": 30, "errors": 2,
"no_match": 10 } ],
  "errors_1h": 2,
  "success_1h": 30,
  "no_match": 10
}
```

`chart` ist stündlich gruppiert. Ohne `date_from`/`date_to` beziehen sich `chart` auf die letzten 24h und die Zähler `errors_1h`/`success_1h` auf die letzte Stunde, `no_match` auf die letzten 24h. Sobald ein Datumsfilter gesetzt ist, gilt dieser Zeitraum für alle drei.

`orders.retraceOrder`

Führt einen gespeicherten Trace **erneut aus** – ein echter Tracking-Aufruf mit den ursprünglichen Parametern (User-Agent, Cookies, IP, Referer werden nachgebildet).

Params: `{ "trace_id": 123 }`

Response (Auszug):

```
{
  "trace_id": 123,
  "type": "etrack",
  "http_code": 200,
  "total_time": "184ms",
  "request": { "method": "GET", "user_agent": "...", "headers": [...], "cookies": "...",
"params": {...} },
  "response": { "...": "Antwort des Tracking-Aufrufs" },
  "success": true,
  "order_id": 999,
  "original_trace": { "id": 123, "result": "order_created", "order_id": 456, "timestamp":
"...", "matching_method": "VoucherCode" }
}
```

> **Achtung:** Das ist kein Dry-Run. Bei erfolgreichem Retrace entsteht eine echte neue Bestellung im System.

`orders.retrackTags`

Spielt für angegebene Bestellungen serverseitige Tags gegen die aktuell hinterlegten Bedingungen neu durch. Zwei Aufrufvarianten:

Parameter	Beschreibung
`traces`	Array bereits geladener Trace-Objekte (z.B. aus einem vorherigen `getOrderTraces`-Aufruf)

Parameter	Beschreibung
`order_ids`	Alternativ: String mit Bestell-IDs, komma-/zeilen-/leerzeichen-getrennt

Result-Werte

Wert	Bedeutung
`order_created`	Bestellung wurde erfolgreich erstellt
`duplicate`	Doppelte Bestellung erkannt
`freezed`	Bestellung eingefroren (Schutz vor Mehrfachbuchung)
`no_match`	Keine passende Kampagne/Zuordnung gefunden (kein Fehler)
`error`	Technischer Fehler
`campaign_blocked`	Kampagne ist blockiert/pausiert

Fehlerbehandlung

Fehler kommen im Format `{ "error": true, "msg": "...", "type": "Ws\\Error", "data": [] }` mit HTTP-Status 500. Eine leere Ergebnismenge (z.B. Filter ohne Treffer) ist **kein** Fehler – hier kommt ganz normal `"data"` mit leerem `traces`-Array bzw. `"total": 0` zurück.

Touchpoint Webservice

EMID Webservice

Einleitung

Um zu einer **EMID** automatisiert die zugehörigen Touchpoint-Daten aus dem **easy.affiliate-System** auslesen zu können, stellt die easy Marketing GmbH eine Webservice-API zur Verfügung. Diese liefert zu einer EMID (= `_id` eines Dokuments der MongoDB-Collection `tracking.action`) den kompletten Touchpoint zurück: Zeitpunkt, Typ, Triplet (Kampagne/Projekt/Werbemittel), Weiterleitungs-URL sowie **alle GET-Parameter** des ursprünglichen Requests.

Typischer Use-Case: Zu einer Liste von aaaids/EMIDs die GET-Parameter (z. B. Partner-ID) ermitteln, ohne manuellen Einzel-Download über TripleA.

Jedem User stehen hierfür ein **Authentifizierungs-Token** und eine **Login_id** zur Verfügung, die über das Frontend abgerufen werden können.

- **User-ID:** Wo ist die User-ID hinterlegt? (Siehe Frontend-Profil)
- **Access-Token:** Wo ist der Access-Token hinterlegt? (Siehe API-Einstellungen)

Endpunkte

Zielgruppe	URL
Admin	`https://SUBDOMAIN.de/ws/V6/admin/JSON/Emids/Read`
Advertiser	`https://SUBDOMAIN.de/ws/V6/advertiser/JSON/Emids/Read`

Beispiele

Beispiel mit cURL

```
curl -X POST \
  -H "Content-Type: application/json" \
  -H "X-Network-ID: -1" \
  -H "X-Auth-Token: ADMIN_APIUSER_TOKEN" \
```

```
  -H "X-Auth-ID: ADMIN_APIUSER_LOGIN_ID" \
  -d '{ "emid": "6512a1f0c3e4b2a1d0f9e8a7" }' \
  https://DOMAIN/ws/V6/admin/JSON/Emids/Read
```

Codebeispiel

Mit der folgenden Methode kann zu einer EMID der Touchpoint ausgelesen werden. Ein Beispielaufruf sieht folgendermaßen aus:

Headers

Variable	Wert / Bedeutung
Content-Type	`application/json`
X-Network-ID	`-1`
X-Auth-Token	`ADMIN_APIUSER_TOKEN`
X-Auth-ID	`ADMIN_APIUSER_LOGIN_ID`

Body

```
{
    "emid": "1314e1f0c3a4b2a1d0f3e5a7"
}
```

Variablenerläuterung

Headers

Variable	Bedeutung	Datentyp
Content-Type	Der Content-Type des Requests	String
X-NETWORKID	Hier wird die ID des Mandanten eingetragen. Wenn nur ein Mandant vorhanden ist oder mandantenübergreifend gearbeitet wird, muss der Wert “-1” eingetragen werden.	Integer
X-AUTH-TOKEN	Hier wird der API-Authentifizierungs-Token des Admin Nutzers hinterlegt.	String
X-AUTH-ID	Hier wird die ID des Admin Nutzers hinterlegt.	Integer

Body

Parameter	Beschreibung	Datentyp	Pflichtfeld
emid	Klick-/Action-ID (24-stellig hex) = `_id` des Dokuments in `tracking.action`. Alternativ als `id` übergebbar.	String	Ja

Response

Die Antwort enthält das komplette Action-Dokument im Feld `data`.

Beispiel (Erfolg)


```
{
    "data": {
        "_id": "6512a1f0c3e4b2a1d0f9e8a7",
        "sessionid": "…",
        "time": "2026-09-15 06:51:30",
        "type": "click",
        "triplet": {
            "campaign_id": 146,
            "project_id": 42,
            "admedia_id": 7
        },
        "iphash": "…",
        "deeplink": "https://ziel-shop.de/…",
        "Request": {
            "Get": {
                "partner_id": "12345",
                "subid": "…"
            },
            "Server": {
                "HTTP_REFERER": "https://…",
                "HTTP_USER_AGENT": "Mozilla/5.0 …"
            }
        }
    },
```

```
    "log": "no logmessage provided",
    "sessionId": null,
    "session": null
}
```

Felder im `data`-Objekt

Feld	Beschreibung	Datentyp
_id	EMID (ObjectId des Action-Dokuments)	String
sessionid	Session-ID des Users	String
time	Zeitpunkt des Touchpoints	Datetime
type	Typ des Touchpoints (`click`, `view`, …)	String
triplet.campaign_id	Kampagnen-ID	Integer
triplet.project_id	Projekt-ID	Integer
triplet.admedia_id	Werbemittel-ID	Integer
iphash	Hash der IP-Adresse	String
deeplink	Weiterleitungs-/Ziel-URL	String
Request.Get	Alle GET-Parameter des Requests (z. B. Partner-ID)	Object
Request.Server	Server-Variablen des Requests (`HTTP_REFERER`, `HTTP_USER_AGENT`, …)	Object

> **Hinweis:** Der Endpunkt liefert **IDs**, keine Klartext-Namen. Kampagnen-/Projekt-/Werbemittel-**Titel** müssen bei Bedarf separat aufgelöst werden.

Fehler

Fehler werden als JSON mit `error: true` zurückgegeben.

Typ	`message` (Beispiel)	Ursache
Ws\NotFoundException	`The requested entity 'Action <emid> not found' couldn't be found.`	EMID existiert nicht in `tracking.action`
InvalidArgumentException	`internal server error. Details: Invalid emid format`	`emid` ist keine gültige 24-stellige Hex-ObjectId

Typ	`message` (Beispiel)	Ursache
InvalidArgumentException	`Missing emid parameter`	`emid` / `id` fehlt im Body
403	–	Authentifizierung fehlgeschlagen (Token/Session ungültig)

Beispiel (nicht gefunden)


```
{
    "data": false,
    "error": true,
    "type": "Ws\\NotFoundException",
    "message": "The requested entity 'Action 6512a1f0c3e4b2a1d0f9e8a7 not found' couldn't be found.",
    "session": null,
    "sessionid": null
}
```


***) Hinweis:** Der Endpunkt verarbeitet **eine EMID pro Request**. Für eine Liste bitte pro EMID einen Request absetzen; die CSV-Aufbereitung erfolgt clientseitig aus der JSON-Antwort.

Webservice Nachbuchungen

Nachbuchungsanfragen per API abrufen und bearbeiten (V6-REST-API)

Über die V6-REST-API können Sie Nachbuchungsanfragen Ihrer Publisher automatisiert abrufen, prüfen, annehmen, ablehnen und bearbeiten. Die API bietet dieselben Funktionen wie das **Nachbuchungstool** in der Oberfläche. Jede Entscheidung, die Sie per API treffen, erscheint dort ebenfalls und wird in der Historie der Anfrage mit Zeitpunkt und Benutzer protokolliert.

---

1. Zugang

Basis-URL

Rolle	Basis-URL
Advertiser-Login	`https://<Ihre-Domain>/ws/V6/advertiser/REST/`
Admin-Login (Netzwerkbetreiber)	`https://<Ihre-Domain>/ws/V6/admin/REST/`

`<Ihre-Domain>` ist die Domain Ihrer Installation, also dieselbe Adresse, unter der Sie sich in der Oberfläche anmelden.

Authentifizierung

Jeder Aufruf braucht drei HTTP-Header:

Header	Wert
`X-Auth-ID`	ID Ihres Benutzers bzw. Advertiser-Logins
`X-Auth-Token`	API-Token dieses Benutzers
`X-Network-ID`	`1`

Wo Sie den Token finden: Im Menü unter *API → Daten-API*. Dort wird Ihr API-Token angezeigt.

Berechtigung

Der Benutzer braucht das Recht **Nachbuchungstool** (`bookingrequests_tool`). Zum Abrufen genügt Leserecht, für Entscheidungen und Änderungen ist Lese- und Schreibrecht nötig. Die API arbeitet immer mit genau den Rechten des Benutzers, dessen Token Sie verwenden. Ein Advertiser-Login sieht nur die Nachbuchungen seiner eigenen Kampagnen.

---

2. Endpunkte im Überblick

Methode	Pfad	Zweck
`GET`	`Bookingrequests`	Liste der Nachbuchungsanfragen (mit Filtern), optional als CSV
`GET`	`Bookingrequests/{id}`	eine Anfrage mit Dubletten, Gutschein-Prüfung, Trigger-Liste und Historie
`POST`	`Bookingrequests`	Aktionen: annehmen, ablehnen, Trigger zuweisen, Provisionsvorschau, CSV-Import
`PUT`	`Bookingrequests/{id}`	eine Anfrage bearbeiten (Trigger, Warenkorbwert, Sale/Lead, Provision)
`GET` / `PUT` / `POST` / `DELETE`	`Bookingrequesthints`	Hinweise für Publisher (Text und Screenshots) pflegen

Die vollständige technische Beschreibung mit allen Feldern finden Sie in der OpenAPI-/Swagger-Dokumentation unter **`https://<Ihre-Domain>/ws/docs/v6`**. Dort melden Sie sich mit Login-ID und Token an.

---

3. Nachbuchungsanfragen abrufen

```
curl "https://<Ihre-
Domain>/ws/V6/advertiser/REST/Bookingrequests?params[state]=open&params[limit]=200" \
  -H "X-Auth-ID: <ID>" \
  -H "X-Auth-Token: <API-TOKEN>" \
  -H "X-Network-ID: 1"
```

Filter

Parameter	Werte	Bedeutung
`params[state]`	`open`, `accepted`, `rejected`, `all` (auch als Liste, z. B. `open,rejected`)	offen (wartet auf Entscheidung), angenommen, abgelehnt, alle
`params[date_field]`	`request` (Standard), `order`	worauf sich `from`/`to` beziehen: Eingang der Anfrage oder Kaufdatum
`params[from]`, `params[to]`	`YYYY-MM-DD`	Zeitraum
`params[search]`	Text	Order-ID, Bestellnummer (auch Teil), Publisher-ID, Projekt-ID oder Anfrage-Nr.
`params[campaign_id]`	ID oder Liste	nur bestimmte Kampagnen (Programme)
`params[publisher_id]`	ID oder Liste	nur bestimmte Publisher
`params[project_id]`	ID oder Liste	nur bestimmte Publisher-Projekte
`params[request_id]`	ID oder Liste	nur bestimmte Anfragen (hochgeladene Dateien)
`params[duplicates]`	`only`, `none`	nur Dubletten / keine Dubletten
`params[voucher]`	`with`	nur Anfragen mit Gutscheincode
`params[sort]`	`request_timestamp` (Standard), `order_timestamp`, `turnover`, `commission`, `id`	Sortierfeld
`params[dir]`	`desc` (Standard), `asc`	Sortierrichtung
`params[limit]`	Standard `50`, höchstens `1000`	Anzahl je Seite
`params[offset]`	Standard `0`	Blättern
`params[format]`	`json` (Standard), `csv`	`csv` liefert alle passenden Zeilen als CSV-Text im Feld `csv`

Jede Zeile enthält Hinweise (`warnings`), die Ihnen bei der Entscheidung helfen:

Hinweis	Bedeutung
`duplicate`	Es gibt bereits eine Transaktion mit derselben Bestellnummer, demselben Trigger und derselben Kampagne
`voucher_unknown`	Gutscheincode ist beim Programm nicht hinterlegt
`voucher_other_publisher`	Gutscheincode ist einem anderen Publisher zugeordnet
`voucher_expired`	Gutscheincode war zum Kaufzeitpunkt nicht gültig

Hinweis	Bedeutung
`voucher_used`	Gutscheincode wurde bereits für eine andere Bestellung genutzt
`commission_changed`	Die neu berechnete Provision weicht von der angefragten ab
`commission_rule_missing`	Für diesen Trigger ist keine Provision hinterlegt

Trigger einer Kampagne abfragen

```
curl "https://<Ihre-
Domain>/ws/V6/advertiser/REST/Bookingrequests?params[view]=triggers&params[campaign_id]=1" \
  -H "X-Auth-ID: <ID>" -H "X-Auth-Token: <API-TOKEN>" -H "X-Network-ID: 1"
```

Eine einzelne Anfrage

```
curl "https://<Ihre-Domain>/ws/V6/advertiser/REST/Bookingrequests/123456" \
  -H "X-Auth-ID: <ID>" -H "X-Auth-Token: <API-TOKEN>" -H "X-Network-ID: 1"
```

Neben den Grunddaten liefert die Detailansicht:

- **Dubletten**: bestehende Transaktionen mit derselben Bestellnummer, demselben Trigger und derselben Kampagne, unabhängig vom Publisher,
- **Gutschein-Prüfung**: ob der Code unbekannt ist, einem anderen Publisher gehört, zum Kaufzeitpunkt ungültig war oder schon eingelöst wurde,
- die **verfügbaren Trigger** der Kampagne,
- die **Historie** aller Änderungen.

> Diese Prüfungen sind Hinweise für Ihre Entscheidung. Sie blockieren nichts.

---

4. Annehmen, ablehnen, Trigger zuweisen

Alle Aktionen laufen über `POST Bookingrequests` mit einem JSON-Body. In `order_ids` können Sie mehrere Anfragen auf einmal übergeben.

Annehmen

```
curl -X POST "https://<Ihre-Domain>/ws/V6/advertiser/REST/Bookingrequests" \
  -H "X-Auth-ID: <ID>" -H "X-Auth-Token: <API-TOKEN>" -H "X-Network-ID: 1" \
  -H "Content-Type: application/json" \
  -d '{ "action": "accept", "order_ids": [123456, 123457], "status": 1 }'
```

`status`	**Ergebnis**
`0`	übernommen als **offene** Transaktion
`1`	übernommen als **bestätigte** Transaktion

Beim Annehmen wird die Provision nach den Kampagneneinstellungen zum Kaufzeitpunkt neu berechnet. Das gilt nicht, wenn sie vorher manuell überschrieben wurde (siehe Abschnitt 5).

Ablehnen

```
curl -X POST "https://<Ihre-Domain>/ws/V6/advertiser/REST/Bookingrequests" \
  -H "X-Auth-ID: <ID>" -H "X-Auth-Token: <API-TOKEN>" -H "X-Network-ID: 1" \
  -H "Content-Type: application/json" \
  -d '{ "action": "reject", "order_ids": [123458], "reject_code": "already_booked", "reason":
"Bestellung bereits über Partner X getrackt" }'
```

Ein Ablehnungsgrund (`reject_code`) ist Pflicht:

`reject_code`	**Bedeutung**
`already_booked`	bereits vorhanden / schon gebucht
`cancelled`	Bestellung storniert
`other_channel`	anderer Kanal
`no_valid_sale`	kein valider Sale
`other`	Sonstiges (dann ist `reason` als Freitext Pflicht)

Der Publisher sieht Grund und Text.

Trigger zuweisen (mehrere Anfragen derselben Kampagne)

```
{ "action": "trigger", "order_ids": [123456, 123457], "trigger_id": 2 }
```

Provisionsvorschau (ohne zu speichern)

```
{ "action": "preview", "order_id": 123456, "trigger_id": 2, "turnover": 89.90 }
```

Die Vorschau liefert die Provision, die sich mit diesem Trigger und Warenkorbwert ergeben würde. Mit `"lead": true` wird sie für einen Lead ohne Warenkorbwert berechnet. Für die Vorschau genügt Leserecht.

Die Aktionen `accept`, `reject` und `trigger` liefern je Order ein eigenes Ergebnis. So sehen Sie, welche Orders verarbeitet wurden und welche nicht.

---

5. Eine Anfrage bearbeiten

```
curl -X PUT "https://<Ihre-Domain>/ws/V6/advertiser/REST/Bookingrequests/123456" \
  -H "X-Auth-ID: <ID>" -H "X-Auth-Token: <API-TOKEN>" -H "X-Network-ID: 1" \
  -H "Content-Type: application/json" \
  -d '{ "trigger_id": 1, "turnover": 79.90 }'
```

Feld	Bedeutung
`trigger_id`	Trigger/Aktion ändern
`turnover`	Warenkorbwert ändern
`lead`	als Lead statt Sale behandeln
`commission_manual` + `commission`	Provision manuell überschreiben (für Sonderfälle)

> Eine manuell überschriebene Provision wird markiert und beim Annehmen **nicht** neu berechnet.

---

6. Entscheidungen gesammelt per CSV übergeben

Viele Entscheidungen auf einmal können Sie auch als CSV übergeben. Das Format ist dasselbe wie beim Import im Nachbuchungstool:

Spalte	Inhalt
`order_id`	ID der Nachbuchungsanfrage
`action`	`accept`, `reject` oder `trigger`
`status`	bei `accept`: `0` oder `1`
`reject_code`, `reason`	bei `reject`
`trigger_id`	bei `trigger`

```
{ "action": "import", "content":
"order_id;action;status;reject_code;reason;trigger_id\n123456;accept;1;;;\n123458;reject;;canc
elled;;", "apply": true }
```

Ohne `apply` (Standard `false`) wird die Datei nur geprüft. Übernommen wird erst ein Aufruf mit `"apply": true`. Wir empfehlen, immer zuerst einen Prüflauf zu machen.

---

7. Hinweise für Publisher pflegen

Über `Bookingrequesthints` (`GET` / `PUT` / `POST` / `DELETE`) pflegen Sie den Hinweistext mit bis zu sechs Screenshots. Publisher sehen ihn beim Einreichen einer Nachbuchung, zum Beispiel mit der Angabe, wo die Bestellnummer steht oder welche Daten nicht mitgeschickt werden sollen. Es gibt einen allgemeinen Hinweis für alle Kampagnen und optional einen eigenen je Kampagne, der dann Vorrang hat.

---

8. Benachrichtigung bei neuen Anfragen

Unabhängig von der API erhalten Sie bei jeder neuen Nachbuchungsanfrage eine E-Mail an Ihre Kontaktadresse. Sie enthält Kampagne, Publisher, Anzahl, Warenkorbwert, mögliche Dubletten, Gutschein-Hinweise und die Zahl aller offenen Anfragen. Die Mail eignet sich gut als Auslöser für einen API-Abruf. Alternativ fragen Sie die API regelmäßig ab, zum Beispiel einmal pro Stunde mit `params[state]=open`.

---

9. Typischer Ablauf einer Integration

1. **Abrufen:** `GET Bookingrequests?params[state]=open`
2. **Abgleichen:** Bestellnummer und Warenkorbwert mit Ihrem Shop- oder ERP-System vergleichen. Optional die Detailansicht `GET Bookingrequests/{id}` für Dubletten und Gutscheine nutzen.
3. **Korrigieren (optional):** `PUT Bookingrequests/{id}` bei abweichendem Warenkorbwert oder Trigger.
4. **Entscheiden:** `POST Bookingrequests` mit `accept` (Status 0 oder 1) bzw. `reject` und Grund.

---

10. Fehlercodes

HTTP	Bedeutung
`400`	Eingabe ungültig; die Antwort nennt den Grund
`403`	Zugangsdaten falsch, fehlendes Recht „Nachbuchungstool“ oder Funktion für die Installation nicht freigeschaltet
`404`	Anfrage nicht gefunden bzw. nicht für diesen Zugang sichtbar
`405`	Methode nicht unterstützt: Nachbuchungen werden abgelehnt (`action: reject`), nicht gelöscht

Fehlerantworten haben immer einen JSON-Body mit `error: true` und einer Meldung in `msg`. Bei ungültigen Zeilen steht zusätzlich die Liste `errors` dabei.

Bei Fragen zur Einrichtung wenden Sie sich gern an unseren Support.