E-POSTBUSINESS API

Login-API Referenz

Version 1.01

Impressum

Inhalt

1 Einführung

Die Login-API ist Teil der E-POSTBUSINESS API, das die REST-Schnittstelle des E-POST Systems für Anbieter von Software darstellt. Mit Hilfe der Login-API ermöglichen Sie es E-POST Nutzern sich über Ihre Applikation am E-POST System anzumelden. Eine solche Anmeldung ist Voraussetzung, um E-POST Funktionalitäten in Ihre Software zu integrieren.

Integration von E-POST Funktionalitäten

Für die Integration der E-POST Funktionalitäten sprechen Sie nach dem Login die jeweiligen spezifischen APIs an:

Versand-API

Mit der Versand-API entwickeln Sie Applikationen, mit denen Sie E-POSTBRIEFE versenden können. Weitere Informationen siehe *Versand-API Referenz*.

Mailbox-API

Mit der Mailbox-API entwickeln Sie Applikationen, mit denen Sie E-POSTBRIEFE empfangen und verwalten können. Weitere Informationen siehe *Mailbox-API Referenz*.

Safe-API

Mit der Safe-API entwickeln Sie Applikationen, die im E-POSTSAFE Dateien ablegen sowie verwalten können. Weitere Informationen siehe *Safe-API Referenz*.

Funktionsweise

Dieses Dokument beschreibt die Möglichkeiten für Ihre Software, sich bei der Login-API via OAuth 2.0 anzumelden. Für die Anmeldung wird „Authorization-Code-Grant-Flow“ und „Resource-Owner-Password-Credentials-Grant-Flow“ von OAuth 2.0 unterstützt. Die Login-API stellt die Möglichkeit zur Aktualisierung von *Access Tokens* gemäß „Refreshing an Access Token“ bereit. Bei der Login-API handelt es sich um eine REST-API, deren gesamte Kommunikation ausschließlich über HTTPS erfolgt.

Zielgruppe

Diese API-Referenz richtet sich an Entwickler, die in folgenden Themengebieten Kenntnisse haben:

- HTTP und HTTPS
- REST
- JSON
- OAuth 2.0

Partnerportal

Im E-POSTBUSINESS API Partnerportal unter http://partner.epost.de/ finden Sie alle Informationen zur Erstellung Ihrer E-POST Anwendung.

2 Grundbegriffe

Anfrage und Antwort

Anfrage und Antwort werden nach dem HTTP-Standard behandelt. Beachten Sie darüber hinaus folgende Anforderungen:

UTF-8

Für die Verarbeitung von Strings wird die UTF-8-Kodierung vorausgesetzt.

Datums- und Zeitangaben

Alle Datums- und Zeitangaben erfolgen in ISO 8601 (z. B. `2014-07-16T17:05:39+02:00`).

HTTPS-Verschlüsselung

Die Login-API verwendet ausschließlich HTTPS-Verbindungen. Die Identifikationsdaten der Teilnehmenden werden somit im gesamten Kommunikationsprozess geschützt.

HINWEIS

Verifizieren Sie bei jedem Aufruf die Korrektheit des Zertifikats auf der Gegenseite.

JSON-Datenformat

Für den Datenaustausch zwischen dem Client und dem E-POST System wird JSON verwendet, das ein kompaktes Datenformat in für Mensch und Maschine einfach lesbarer Textform ist.

REST-API-Endpoint

Folgender REST-Endpoint wird von der Login-API verwendet:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

Hypermedia-Ansatz

Der REST-API-Endpoint implementiert zur Kommunikation von URIs mit Clients den Hypermedia-Ansatz (HATEOAS: Hypertext As The Engine Of Application State, siehe https://en.wikipedia.org/wiki/HATEOAS). Der Einstiegspunkt der API liefert dabei Verweise (Links) auf die angebotenen Ressourcen. Diese Sammlung von Links wird im Kontext der E-POSTBUSINESS API als Service-Dokument bezeichnet.

Über die Links kann die URI zur Ressource ermittelt werden. Ressourcen können ihrerseits wieder Links zu verknüpften (Sub)-Ressourcen anbieten. Dadurch entsteht eine navigierbare API, die ohne Kenntnis der konkreten und aktuellen URI-Pfade nutzbar ist. Dies sorgt für eine Entkopplung von API und Client und ermöglicht es dem Client durch Prüfung der Existenz von Links zu ermitteln, ob bestimmte API-Funktionen nutzbar sind.

HTTP-Statuscodes

Antworten werden in HTTP-Statuscode-Syntax ausgegeben. Jede zusätzliche Information wird im Body der Antwort zurückgegeben und als JSON-Dokument formatiert.

Neben den Ressourcen-spezifischen Antwortcodes gelten folgende allgemeine Antwortcodes:

405

Method Not Allowed

Die HTTP-Methode ist nicht unterstützt.

415

Unsupported Media Type

Der übergebene `Content-Type` entspricht nicht dem erwarteten Wert.

500

Internal Server Error

Interner Verarbeitungsfehler

503

Service Unavailable

Interner Verarbeitungsfehler

3 Identitätsniveau von Privatkunden

Übersicht

Das Identitätsniveau legt für Privatkunden fest, ob die Nutzung bestimmter E-POST Funktionen möglich ist. Durch das Verfahren wird jeder Privatkunde sofort durch Ausfüllen des Online-Registrierungsformulars medienbruchfrei, d. h. ohne POSTIDENT-Verfahren, registriert und mit direktem Portalzugang für E-POST freigeschaltet. Es gibt vier verschiedene Identitätsniveaus, denen ein Privatkunde abhängig von der Verifizierung seiner Identität und seiner postalischen Adresse zugeordnet wird.

HINWEIS

Gegenüberstellung von Identitätsniveau und Authentifizierungsniveau

Identitätsnivau

Das Identitätsniveau gibt an, welche Kenntnisse das E-POST System über einen Kunden hat (dessen Identität) und was davon im Rahmen der Registrierung verifiziert worden ist.

Authentifizierungsniveau (siehe 4. Authentisierung)

Das Authentifizierungsniveau gibt an, in welcher Weise ein Kunde seine Identität während der Nutzung der E-POST Produkte nachgewiesen hat:

- einfach per Benutzername/Passwort

oder

- erweitert mit zweitem Faktor (*HandyTAN*). Der zweite Faktor bietet dem System eine höhere Gewissheit, dass es sich wirklich um die jeweilige Identität handelt.

Registrierung und Identitätsniveau

Ein Privatkunde erhält ein bestimmtes Identitätsniveau, das sich abhängig von der Verifizierung seiner Identität und seiner postalischen Adresse im Rahmen des Registrierungsprozesses ergibt.

Basic

Durch Ausfüllen des Online-Registrierungsformulars registriert sich ein Privatkunde sofort ohne POSTIDENT-Verfahren und erhält das Identitätsniveau *Basic*.

Premium

Ein *Basic*-Kunde, der sich im Nachgang zur Registrierung per POSTIDENT-Verfahren verifiziert, erhält das Niveau *Premium*.

Darüber hinaus kann sowohl ein *Basic*- als auch ein *Premium*-Kunde seine Adresse verifizieren und wechselt damit in einen weiteren Status:

Basic+

Ein *Basic*-Kunde, der zusätzlich seine Adresse mittels AdressTAN verifiziert, erhält das Niveau *Basic+*.

Premium+

Ein *Premium*-Kunde, der zusätzlich zu seiner POSTIDENT-Verifizierung seine Adresse mittels AdressTAN verifiziert, erhält das Niveau *Premium+*.

HINWEIS

Die AdressTAN wird in einem physischen Brief an die angegebene Adresse versendet.

E-POST Funktionen in Abhängigkeit zum Identitätsniveau

Ein Privatkunde kann **abhängig von seinem Identitätsniveau** bestimmte E-POST Funktionen **nutzen** (✓) bzw. **nicht nutzen** (⊘):

Funktion	Basic	Basic+	Premium	Premium+
E-POSTBRIEFE physisch versenden	✓	✓	✓	✓
E-POSTSAFE verwenden	✓	✓	✓	✓
Faxe schreiben und empfangen	✓	✓	✓	✓
Elektronische E-POSTBRIEFE empfangen	⊘	⊘	✓	✓
E-POSTBRIEFE elektronisch versenden	⊘	⊘	✓	✓
E-POSTZAHLUNG verwenden	⊘	⊘	✓	✓
E-POSTIDENT verwenden	⊘	⊘	✓	✓
E-POSTSCAN nutzen	⊘	✓	⊘	✓

Tabelle 3-1 E-POST Funktionen in Abhängigkeit zum Identitätsniveau

Vorgehensweise: Implementierung mit der Versand-API

Nach erfolgreichem Login eines Nutzers bekommt Ihre Anwendung das Identitätsniveau (Parameter `id_level`) von diesem in der Antwort mitgeteilt (siehe 4.1.2 Access Token anfordern).

Bei der Umsetzung des Versandprozesses stellen Sie sicher, dass nur die für das Identitätsniveau des Nutzers zugelassenen Funktionen aufgerufen werden können. Darüber hinaus verarbeiten Sie entsprechende Fehlermeldungen, falls eine nicht zulässige Funktion aufgerufen wurde.

Beispiel:

Ihre Applikation kann entweder proaktiv verhindern, dass ein *Basic*-Nutzer einen elektronischen E-POSTBRIEF verschickt, oder diesen Vorgang generell zulassen und die entsprechende Fehlermeldung verarbeiten und einen Hinweis anzeigen.

HINWEIS

Geschäftskunden haben kein Identitätsniveau und es sind für diese alle mit der E-POSTBUSINESS API umsetzbaren E-POST Funktionen nutzbar.

4 Authentisierung

Mit der Login-API authentisiert sich Ihre Applikation für das E-POST System.

OAuth 2.0

Die Anmeldung Ihrer Applikation am E-POST System erfolgt mit OAuth 2.0. Dieses nutzt die Mittel von HTTP, so dass die Anmeldung über einen Browser erfolgen kann. Alternativ kann eine Anmeldung auch ohne einen Browser erfolgen.

Registrierung

Für die Registrierung wenden Sie sich zunächst an unseren Support im Partnerportal (http://partner.epost.de/registrieren/).

Im Laufe der Registrierung zu E-POSTBUSINESS API werden folgende Daten aufgenommen:

- eine `DevId`, die Ihre Firma eindeutig identifiziert.
- eine `AppId`, die Ihre Applikation eindeutig identifiziert.
- eine *Redirect-URL*, die im Rahmen des OAuth-2.0-Prozesses zurückgegeben wird.

HINWEIS

Beachten Sie bei der Eingabe der Daten Folgendes:

- Beachten Sie bei `DevId` und `AppId` folgende Einschränkungen bei den erlaubten Zeichen:
  - Erlaubt sind nur **Buchstaben** (A-Z und a-z), Ziffern (0-9) und **Unterstrich** („_“).
  - **Nicht** erlaubt sind Leer- oder Sonderzeichen.
  - Erlaubt sind **maximal 80 Zeichen**.
- Geben Sie für eine eindeutige Identifizierung für die `DevId` den vollständigen Firmennamen an (z. B. „FirmennameGmbH“).
- Beschreiben Sie die Funktionalität Ihrer Applikation für die `AppId` (z. B. „VersandApp“).
- Die *Redirect-URL* definieren Sie selbst, z. B. `https://ihredomain.de/epost/send`.

Nach erfolgreicher Registrierung bekommen Sie von der Deutschen Post AG Testdaten für die Test- und Integrationsumgebung zugesandt.

License-File (LIF-Datei)

Mit den Registrierungsdaten wird ein *License-File* (LIF-Datei) erstellt und Ihnen zugesendet. Bei der LIF-Datei handelt es sich um einen kryptographischen Schlüssel, der beim Authentisierungsprozess benötigt wird. Die LIF-Datei stellt sicher, dass Ihre Applikation die notwendigen Berechtigungen besitzt, um das E-POST System zu verwenden.

HINWEIS

Im Folgenden werden nur die Schritte aufgezeigt, die Ihre Applikation implementieren muss, um sich über die Login-API authentifizieren zu können. Die OAuth-2.0-spezifischen Schritte, die zwischen Browser und Versand-API, Mailbox-API oder Safe-API stattfinden, werden nicht beschrieben, da in Ihrer Applikation in diesem Zusammenhang keine Implementierung stattfinden muss.

Wahl der Authentisierungs-Spezifikation

Sie haben die Wahl, eine oder beide der folgenden OAuth-2.0-Spezifikationen für die Anmeldung zu verwenden:

- Authentisierung mit *Authorization Code Grant* (mit Benutzeraktion) (siehe 4.1 Authentisierungs-Prozess nach Authorization Code Grant-Spezifikation)
- Authentisierung mit *Resource Owner Password Credentials Grant* (optionale Benutzeraktion) (siehe 4.2 Resource Owner Password Credentials Grant)

4.1 Authentisierungs-Prozess nach Authorization Code Grant-Spezifikation

Der Authentisierungs-Prozess erfolgt nach *Authorization Code Grant*-Spezifikation, wenn eine Implementierung mit Benutzeraktion umgesetzt wird.

Weitere Informationen zum *Authorization Code Grant* finden Sie unter http://tools.ietf.org/html/rfc6749#section-4.1. Die folgenden Schritte beschreiben sequentiell den Ablauf des Authentisierungs-Prozesses.

1. Für den Zugriff einer Applikation auf eine Ressource eines Benutzers für einen bestimmten Zweck (gekennzeichnet durch Scope, z. B. Versenden eines E-POSTBRIEFS, Empfang eines E-POSTBRIEFS, Zugriff auf den E-POSTSAFE).
   - öffnet die Applikation einen Browser und teilt diesem mit, dem Benutzer die E-POST Login-Seite anzuzeigen,
   - übergibt die Applikation dem Browser die benötigten Parameter (z. B. Scope).
2. Der Browser zeigt dem Benutzer die Login-Seite an.

   Der Browser teilt dem Benutzer mit, dass die Applikation mit seinen Ressourcen interagieren möchte.
3. Der Benutzer gibt auf der Login-Seite seine Nutzerdaten ein.
4. Die Login-API validiert die Nutzerdaten.

   Falls der Scope ein **hohes** Authentifizierungsniveau benötigt
   - veranlasst die Login-API die Zusendung einer *HandyTAN* per SMS an den Benutzer,
   - antwortet die Login-API mit einem Redirect auf die E-POST *TAN-Seite*.

   Falls der Scope ein **normales** Authentifizierungsniveau erfordert
   - wird Schritt 8 fortgesetzt.
5. Der Browser zeigt dem Benutzer die *TAN-Seite* an.
6. Der Benutzer gibt die erhaltene *HandyTAN* auf der *TAN-Seite* ein und sendet diese ab.
7. Die Login-API validiert die *HandyTAN*.
8. Der Browser gibt die Kontrolle an die Applikation zurück, indem er die Redirect-URL aufruft und einen Autorisierungscode übergibt.
9. Die Applikation extrahiert den Autorisierungscode und fordert ein Access Token an.

Die Applikation verwendet das Access Token für die Zugriffe auf die Ressourcen des Benutzers.

4.1.1 Login-Seite anfordern

Durch das Anfordern der Login-Seite gibt die Applikation dem Nutzer die Möglichkeit sich am E-POST System zu authentifizieren und der Applikation Rechte zu gewähren. Die Authentifizierung der Applikation erfolgt erst beim Tausch des Authorization Code in ein Access Token.

Ihre Applikation beantragt per separatem Browser eine Authentisierung (Login oder Login mit *HandyTAN*). Der Browser wickelt danach den Authentisierungsprozess mit der Login-API ab.

Anfrage

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

URL-Struktur

```
GET /oauth2/auth
```

Pfad-Parameter

response_type (erforderlich)

Ein String der festlegt, welche Berechtigung der Client für die Autorisierung benötigt. Zurzeit wird ausschließlich der Wert `code` unterstützt.

client_id (erforderlich)

Ein String, der die Client-Applikation identifiziert.

HINWEIS

Die `client_id` ergibt sich aus `DevId` und `AppId`, die Sie bei der Registrierung für E-POSTBUSINESS API angeben. Die Bildungsvorschrift der `client_id` ist folgende:

- clientid = devid "," appid

state (optional)

Ein beliebiger String von maximal 512 Zeichen Länge, der den Zustand der Applikation beschreibt. Der Wert wird der Applikation im Login-Ergebnis zurückgegeben.

ACHTUNG

Ihre Applikation muss diesen Parameter angeben, um *Cross-Site-Request-Forgery* zu verhindern, auch wenn der Parameter nicht als Pflicht-Wert erwartet wird. Weitere Informationen siehe http://tools.ietf.org/html/draft-ietf-oauth-v2-30#section-10.12.

TIPP

Es kann mit diesem Parameter z. B. eine *gehashte* Session-ID oder ein zufallsgenerierter Wert übergeben werden.

scope (erforderlich)

Der Scope legt fest, inwieweit der Client auf die Ressource(n) des Nutzers zugreift.

- Folgende Scopes sind möglich:

  send_letter
  - Erlaubt das Versenden von elektronischen E-POSTBRIEFEN
  - Erfordert ein hohes Authentifizierungsniveau
  - Weitere Informationen siehe *Versand-API Referenz*

  send_hybrid
  - Erlaubt das Versenden von physischen E-POSTBRIEFEN
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Versand-API Referenz*

  read_letter
  - Erlaubt das Lesen von E-POSTBRIEFEN
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Mailbox-API Referenz*

  create_letter
  - Erlaubt das Erstellen von E-POSTBRIEF Entwürfen
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Mailbox-API Referenz*

  delete_letter
  - Erlaubt das Löschen von E-POSTBRIEF Entwürfen
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Mailbox-API Referenz*

  safe
  - Erlaubt den Zugriff auf den E-POSTSAFE
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Safe-API Referenz*

  change_account_settings
  - Ermöglicht die Änderung von Nutzerinformationen über die Account-Service-API
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Account-Service-API Referenz*

  read_account_info
  - Ermöglicht das Lesen von Nutzerinformationen über die Account-Service-API
  - Erfordert ein normales Authentifizierungsniveau
  - Weitere Informationen siehe *Account-Service-API Referenz*

HINWEIS

Das Authentifizierungsniveau gibt an, ob sich der Nutzer des E-POST Systems mit einer HandyTAN authentifizieren muss (hoch) oder nicht (normal).

- Es ist möglich, mehrere Scopes gleichzeitig anzugeben. Mehrere Scopes werden mit einem Leerzeichen getrennt.

  Beispiele:

  - `scope=send_letter create_letter`
  - `scope=safe read_letter`
  - `scope=delete_letter read_letter`

redirect_uri (erforderlich)

Eine URI, an welche das Login-Ergebnis geleitet wird. Die `redirect_uri` muss dem E-POST System bekannt sein. Die finale vom System erzeugte `redirect_uri` darf nicht länger als 2083 Zeichen sein, inklusive Protokollangabe und weiterer Parameter, andernfalls kommt es zu einem Darstellungsproblem sowohl beim Browser als auch beim Server.

Beispiel

```
GET /oauth2/auth?
 response_type=code&
 client_id=TestClientId&
 state=TestState&
 scope=send_letter&
 redirect_uri=https://localhost:8888/briefkasten
 HTTP/1.1
Host: login.epost.de
```

Antwort

302 Found (Erfolgsfall)

Im Erfolgsfall wird eine Redirect-URL mit Autorisierungscode zurückgegeben. Der Server übergibt der Applikation dabei die nachfolgenden Parameter als *Query-String* der Redirect-URL.

Die Antwort enthält folgende Parameter:

code

Der zurückgegebene Autoriorisierungscode.

state

Der bei der Anfrage übergebene Wert wird unverändert durchgereicht.

302 Found (Fehlerfall)

Im Fehlerfall wird Ihrer Applikation eine Redirect-URL mit der entsprechenden Fehlermeldung zurückgegeben. Es wird dabei ein Fehlercode (`error`), eine optionale Fehlerbeschreibung (`error_description`), ein optionaler Hinweis, unter welcher URL eine detaillierte Fehlerbeschreibung abgerufen werden kann (`error_uri`) sowie der in der Anfrage übergebene Wert (`state`) zurückgegeben.

error

access_denied

Ungültige TAN oder falsche `redirect_uri`

invalid_niveau

Die Anmeldung darf nur im hohen Authentifizierungsniveau erfolgen (diese Einstellung kann der Benutzer im Portal setzen)

invalid_request

Fehlende oder fehlerhafte Parameter oder Referer

invalid_scope

Falsches Listenformat oder Liste beinhaltet eine oder mehrere unbekannte oder ungültige Scopes

server_error

Unerwarteter serverseitiger Fehler

temporarily_unavailable

Temporärer serverseitiger Fehler

unauthorized_client

`client_id` ist falsch

unsupported_response_type

Response Type entspricht nicht dem `code`

error_description

Für Menschen lesbarer Text mit zusätzlichen Informationen, um den Entwickler beim Verstehen des aufgetretenen Fehlers zu unterstützen.

error_uri

Eine URI, die eine für Menschen lesbare Webseite mit Informationen über den Fehler informiert.

state

Der in der Anfrage übergebene Wert wird unverändert durchgereicht.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Beispiel: Antwort im Erfolgsfall

```
HTTP/1.1 302 Found
Location: https://localhost:8888/briefkasten
 ?code=dGhpcyBpcyBhbiBiYXN
 &state=TestState
```

Beispiel: Antwort im Fehlerfall

```
HTTP/1.1 302 Found
Location: https://localhost:8888/briefkasten?
 error=access_denied&
 error_description="The resource owner or authorization server denied the
request."&
 error_uri="http://tools.ietf.org/html/rfc6749#section-4.1.2"&
 state=TestState
```

4.1.2 Access Token anfordern

Falls im vorangegangenen Schritt die Anfrage nach dem Autorisierungscode erfolgreich beantwortet wurde und Ihre Applikation diesen erhalten hat, ist der zweite Schritt die Anforderung des Access Tokens über die Login-API.

Dieser Vorgang wird durch den Austausch des Autorisierungscodes zum Access Token durchgeführt, was durch eine weitere Anfrage in Ihrer Applikation realisiert wird.

Anfrage

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

URL-Struktur

```
POST /oauth2/tokens/
```

HTTP-Header

Content-Type (erforderlich)

`application/x-www-form-urlencoded`

Authorization (erforderlich)

Autorisierungs-Kopfzeile gemäß *HTTP Basic authentication scheme* (siehe http://tools.ietf.org/html/rfc6749#section-2.3.1 und http://www.ietf.org/rfc/rfc2617.txt Kapitel 2). Die verwendeten Werte entstammen der bei der Registrierung für E-POSTBUSINESS API angegebenen Daten.

Die Identifikation des Benutzers wird durch das Zusammenfügen des JSON-Feldes `DevId` mit einem Komma und dem JSON-Feld `AppId` gebildet. Der resultierende String wird mit dem `application/x-www-form-urlencoded`-Algorithmus aus http://tools.ietf.org/html/rfc6749#appendix-B kodiert. Dieser kodierte Wert wird als Benutzername verwendet, der Inhalt der LIF-Datei wird mit demselben Algorithmus kodiert und als Passwort benutzt.

Nach dem Zusammenfügen des Benutzernamens, einem „:“-Zeichen und dem Passwort wird der Wert mit Base64-Algorithmus kodiert. Dem resultierenden String wird das Wort `Basic` mit einem Leerzeichen vorangestellt und als Wert für den HTTP-Header `Authorization` benutzt.

Bildungsvorschrift:

- Basic = basic-credentials
- basic-credentials = base64-client-pass
- base64-client-pass = <base64 encoding of client-pass>
- client-pass = encoded-clientid ":" encoded-password
- encoded-clientid = <x-www-form-urlencoding of clientid>

- encoded-password = <x-www-form-urlencoding of password>
- clientid = devid "," appid
- devid = <DevId aus LIF-Datei>
- appid = <AppId aus LIF-Datei>
- password = <Inhalt der LIF-Datei>

HTTP-Body Die Client-Applikation sendet die folgenden Parameter im Format `application/x-www-form-urlencoded` mit `UTF-8`-Codierung im *entity-body* der HTTP-Anfrage.

grant_type (erforderlich)

Legt fest, welchen Code-Typ der Client für den Token-Tausch anbietet. Für die Authentisierung mit Authorization Code wird als Typ-Angabe nur `authorization_code` unterstützt.

code (erforderlich)

Der vom sich authentisierenden Benutzer erhaltene Autorisierungscode.

scope (erforderlich)

Der bei der Autorisierung angeforderte Scope. Dieser muss dem Scope entsprechen, der bei der Anforderung der Login-Seite angegeben wurde.

redirect_uri (erforderlich)

Die bei der Autorisierung verwendete Redirect-URL (siehe http://tools.ietf.org/html/rfc6749#section-4.1.3). Nach OAuth2.0-Spezifikation muss die `redirect_uri` mit übertragen werden, damit die Konsistenz der Anfrage überprüft werden kann.

Beispiel

```
POST /oauth2/tokens/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

grant_type=authorization_code&
 code=dGhpcyBpcyBhbiBiYXN&
 scope=send_letter%20create_letter&
 redirect_uri=https://localhost:8888/briefkasten/
```

Antwort

200 OK

Im Erfolgsfall wird Ihrer Applikation ein Ticket in Form eines JSON-Objekts zurückgegeben, welches ein Access Token enthält.

- Je nach angegebenen Scope können Sie die Versand-API oder Mailbox-API Ressourcen für den Versand bzw. Zugriff verwenden sowie auf Dokumente oder Verzeichnisse über die Safe-API zugreifen oder ändern.

HINWEIS

Die Antwort enthält im Gegensatz zur OAuth2.0-Spezifikation kein eigenes Refresh Token für die Aktualisierung. Um dies umzusetzen, wird das bereits bestehende Access Token erneuert.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit dem Ticket.

Das Ticket ist folgendermaßen aufgebaut:

access_token

Das Access Token in Form eines Strings.

expires_in

Gültigkeitsdauer des Access Tokens in Sekunden.

Das Access Token hat eine begrenzte Gültigkeitsdauer, die von der Login-API bestimmt wird.

token_type

Der Parameter wird vom Standard verlangt und ist bei der E-POSTBUSINESS API immer `Bearer`.

id_level

Das Identitätsniveau des angemeldeten E-POST Nutzers (siehe 3. Identitätsniveau von Privatkunden).

Mögliche Werte sind `basic`, `basicplus`, `premium`, `premiumplus` und `null`. Der Wert `null` tritt auf, wenn es sich um einen Geschäftskunden handelt.

HINWEIS

Wenn das Access Token seine Gültigkeitsdauer überschritten hat, wird die Anfrage mit dem entsprechenden Hinweis `invalid_token` abgelehnt. Danach kann Ihre Applikation das Token für einen längeren Zeitraum bei der Login-API erneuern. Dafür muss Ihre Applikation das bestehende Access Token verwenden. Ein gesondertes Refresh Token wird nicht ausgehändigt.

ACHTUNG

Stellen Sie sicher, dass das Access Token sicher in der Applikation gehalten wird und kein Fremdzugriff darauf erfolgen kann.

400 Bad Request

Die Anfrage ist fehlerhaft aufgebaut.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_request

Parameter-Fehler

unsupported_grant_type

`grant_type` ist ungleich `authorization_code`

invalid_grant

Ungültiger *authorization_code*

unauthorized_client

Der `grant_type` ist für den authentifizierten Client nicht erlaubt

invalid_scope

Verwendung eines falschen Listenformats oder die angegebene Liste enthält eine oder mehrere unbekannte oder ungültige Scope bzw. die übergebene Liste gleicht nicht der Liste, die bei der Beantragung des verwendeten Autorisierungscodes verwendet wurde.

401 Unauthorized

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_client

Credentials fehlen, Client-ID ist ungültig oder Credentials sind nicht valide.

403 Forbidden

Es besteht keine Berechtigung, um auf die Ressource zuzugreifen.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_niveau

Die Anmeldung darf nur im hohen Authentifizierungsniveau erfolgen (diese Einstellung kann der Benutzer im Portal setzen)

415 Unsupported Media Type

Der Medientyp der Datei ist nicht erlaubt.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

unsupported_media_type

Der Content-Type der Anfrage wird nicht unterstützt.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Beispiel: Antwort im Erfolgsfall

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
   "access_token": "V0aG9yaXphdGlvbiBjb2Rl",
   "token_type": "Bearer",
   "expires_in": 3600,
   "id_level": "basicplus"
}
```

Beispiel: Antwort im Fehlerfall

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=UTF-8

{
   "error": "invalid_request",
   "error_description": "See URI below to get detailed information about
the error.",
   "error_uri": "http://tools.ietf.org/html/rfc6749#section-5.2"
}
```

4.1.3 Gültigkeitsdauer von Access Tokens

Ein Access Token hat folgende Parameter, die definieren, wie lange es erneuert werden kann. Weitere Informationen finden Sie unter 4.1.4 Access Token erneuern.

Parameter	Wert (h:min)	Beschreibung
`validity`	0:10	Zeitspanne, innerhalb deren ein Access Token (unabhängig vom Authentifizierungsniveau) gültig ist und genutzt werden kann. Nach Ablauf dieser Zeit kann das Access Token nicht mehr für Operationen auf der API verwendet werden. Es kann aber erneuert werden.

Parameter	Wert (h:min)	Beschreibung
`available`	1:00	Zeitspanne, innerhalb deren ein Access Token auf **normalem** Authentifizierungsniveau spätestens erneuert werden muss. Wird es nicht innerhalb dieser Zeitspanne erneuert, wird es ungültig und lässt sich auch nicht mehr erneuern.
`downgrade`	0:15	Zeitspanne, innerhalb deren ein Access Token auf **hohem** Authentifizierungsniveau erneuert werden muss. Wird es nicht innerhalb dieser Zeit erneuert, wechselt es ins normale Authentifizierungsniveau. Da die `validity`-Zeit dann bereits verstrichen ist, muss es erneuert werden, damit es wieder (jetzt auf **normalem** Authentifizierungsniveau) verwendet werden kann. Auch die `available`-Zeit ist zu diesem Zeitpunkt bereits teilweise verstrichen. Es bleiben also noch 45 min übrig, bis das Access Token ungültig wird.
`max.validity`	24:00	Maximale Gültigkeitsdauer eines Access Token trotz kontinuierlichen Erneuerns. Nach Ablauf ist kein Erneuern mehr möglich.

Tabelle 4.1-1 Parameter von Access Tokens

Grafisch lassen sich die verschiedenen Parameter von Access-Tokens wie folgt darstellen.

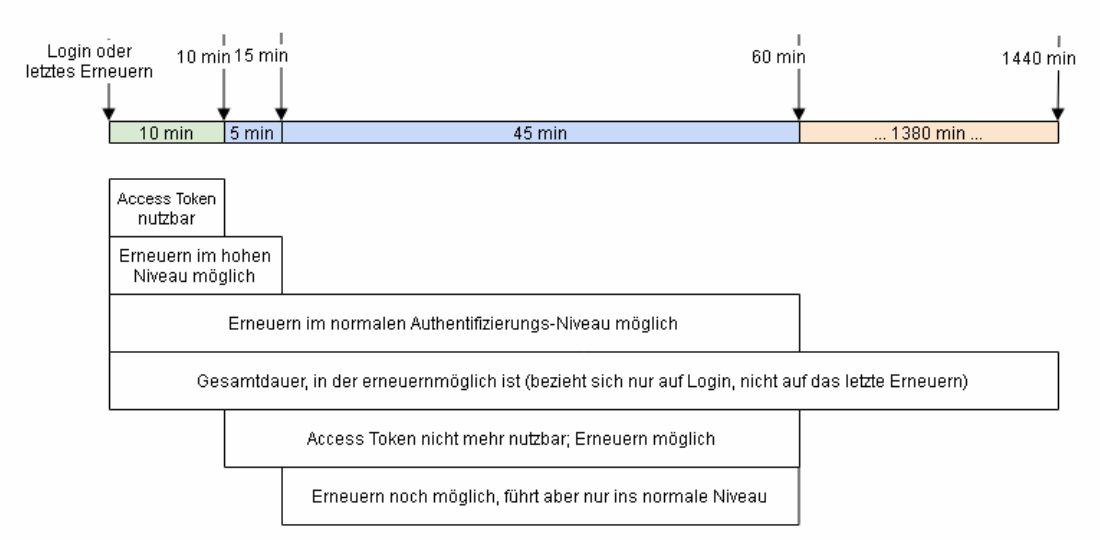

Abbildung 4.1-1 Übersicht: Gültigkeitszustände eines Access Tokens

4.1.4 Access Token erneuern

Wenn das Access Token nicht mehr gültig ist, kann Ihre Applikation dieses unter Verwendung Ihrer Registrierungsdaten gegenüber dem Autorisierungsserver aktualisieren.

Die Möglichkeit, das Access Token zu erneuern, hängt von den Parametern ab, die seine Gültigkeit definieren. Weitere Informationen finden Sie unter 4.1.3 Gültigkeitsdauer von Access Tokens.

Die Schnittstelle entspricht dem Kapitel „Refreshing an Access Token“ aus RFC 6749 (siehe http://tools.ietf.org/html/rfc6749#section-3) und dem Kapitel „The WWW-Authenticate Response Header Field“ aus RFC 6750 (siehe http://tools.ietf.org/html/rfc6750#section-3). Als Refresh Token wird kein separates *Refresh Token* sondern das Access Token verwendet.

Im folgenden Diagramm sind die einzelnen Fälle dargestellt.

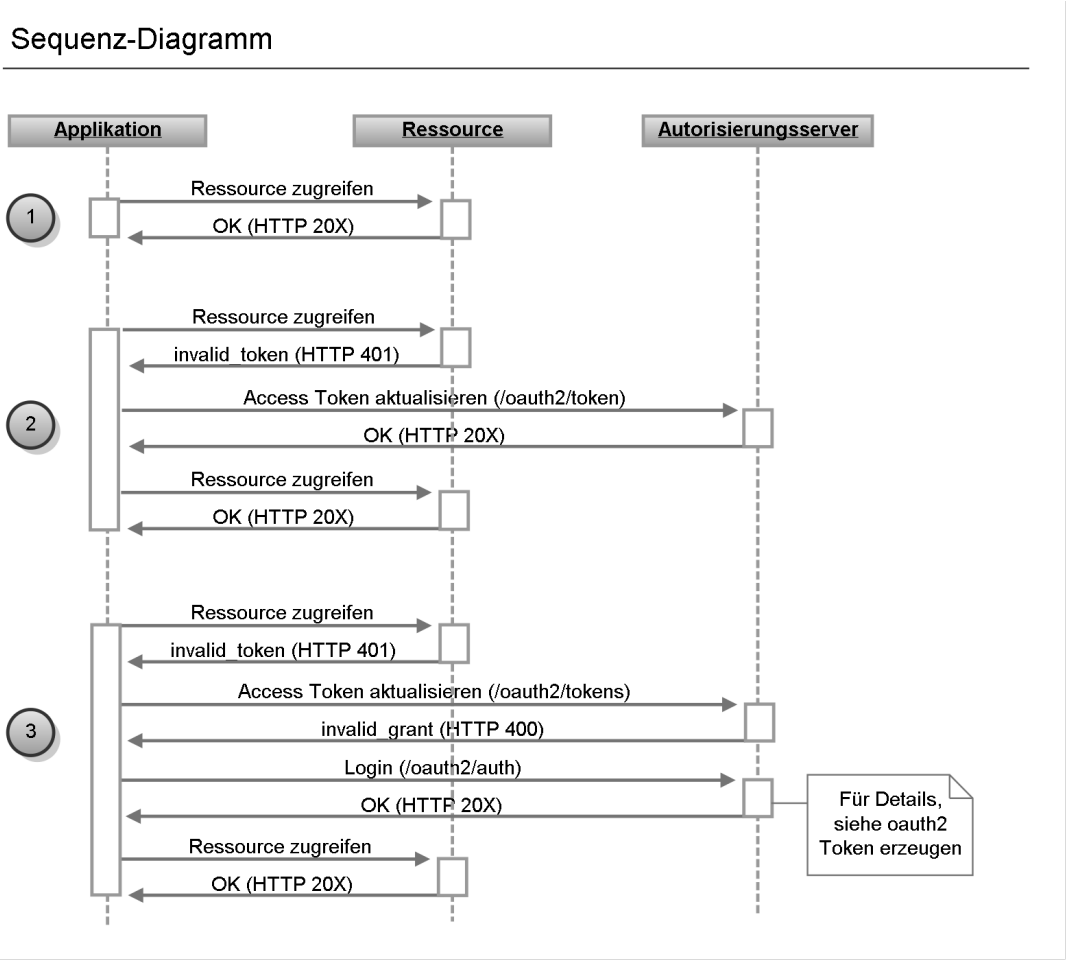

Abbildung 4.1-2 Sequenz-Diagramm für die Erneuerung des Access Tokens

1. Das Access Token ist gültig:

   Auf die jeweilige Ressource kann normal zugegriffen werden.
2. Das Access Token ist ungültig:
   - Der Aufrufende der Ressource erhält als Antwort BAD REQUEST mit dem Fehler `invalid_token` und dem Statuscode 401. Die Applikation kann nun versuchen das Access Token zu aktualisieren.
   - Die Aktualisierung erfolgt durch einen POST an den Autorisierungsserver mit der LIF-Datei und dem aktuellen Access Token als Anfrage-Parameter.
   - Wenn das Access Token erfolgreich aktualisiert werden konnte, dann antwortet der Autorisierungsserver mit dem aktualisierten Access Token. Dieses kann dann verwendet werden, um die ursprüngliche Ressource zu nutzen.
3. Das Access Token ist ungültig und kann nicht aktualisiert werden:

- Der Aufrufende der Ressource erhält als Antwort BAD REQUEST mit dem Fehler `invalid_token` und dem Statuscode 401. Die Applikation kann nun versuchen das Access Token zu aktualisieren.
- Die Aktualisierung erfolgt durch einen POST an den Autorisierungsserver mit der LIF-Datei und dem aktuellen Access Token als Anfrage-Parameter.
- Wenn das Access Token nicht aktualisiert werden konnte, antwortet der Autorisierungsserver mit dem Fehler `invalid_grant` und dem Statuscode 400. Die Applikation kann nun versuchen ein neues Access Token zu erzeugen.
- Die Applikation beauftragt über die Login-API einen neuen Login. Nachdem der Nutzer sich wieder eingeloggt hat, kann die Applikation den *Authorization Code* wieder gegen ein *Access Token* tauschen und die Ressource weiter benutzen.

Anfrage

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

URL-Struktur

```
POST /oauth2/tokens/
```

Content-Type (erforderlich)

`application/x-www-form-urlencoded;charset=UTF-8`

Authorization (erforderlich)

Autorisierungs-Kopfzeile gemäß *HTTP Basic authentication scheme*

HTTP-Body

Die Client-Applikation sendet die folgenden Parameter im Format `application/x-www-form-urlencoded` mit `UTF-8`-Codierung im *entity-body* der HTTP-Anfrage.

grant_type (erforderlich)

Der Wert `refresh_token` muss gesetzt werden (siehe http://tools.ietf.org/html/rfc6749#section-6).

refresh_token (erforderlich)

Das aktuelle Access Token.

Beispiel

```
POST /oauth2/tokens/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

grant_type=refresh_token&refresh_token=dGhpcyBpcyBhbiBiYXN
```

Antwort

200 OK

Im Erfolgsfall wird Ihrer Applikation ein Ticket in Form eines JSON-Objekts zurückgegeben, welches ein Access Token enthält.

- Je nach angegebenen Scope können Sie die Versand-API oder Mailbox-API Ressourcen für den Versand bzw. Zugriff verwenden sowie auf Dokumente oder Verzeichnisse über die Safe-API zugreifen oder ändern.

HINWEIS

Die Antwort enthält im Gegensatz zur OAuth2.0-Spezifikation kein eigenes Refresh Token für die Aktualisierung. Um dies umzusetzen, wird das bereits bestehende Access Token erneuert.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit dem Ticket.

Das Ticket ist folgendermaßen aufgebaut:

access_token

Das Access Token in Form eines Strings.

expires_in

Gültigkeitsdauer des Access Tokens in Sekunden.

Das Access Token hat eine begrenzte Gültigkeitsdauer, die von der Login-API bestimmt wird.

token_type

Der Parameter wird vom Standard verlangt und ist bei der E-POSTBUSINESS API immer `Bearer`.

id_level

Das Identitätsniveau des angemeldeten E-POST Nutzers (siehe 3. Identitätsniveau von Privatkunden).

Mögliche Werte sind `basic`, `basicplus`, `premium`, `premiumplus` und `null`. Der Wert `null` tritt auf, wenn es sich um einen Geschäftskunden handelt.

HINWEIS

Wenn das Access Token seine Gültigkeitsdauer überschritten hat, wird die Anfrage mit dem entsprechenden Hinweis `invalid_token` abgelehnt. Danach kann Ihre Applikation das Token für einen längeren Zeitraum bei der Login-API erneuern. Dafür muss Ihre Applikation das bestehende Access Token verwenden. Ein gesondertes Refresh Token wird nicht ausgehändigt.

ACHTUNG

Stellen Sie sicher, dass das Access Token sicher in der Applikation gehalten wird und kein Fremdzugriff darauf erfolgen kann.

400 Bad Request

Die Anfrage ist fehlerhaft aufgebaut.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_request

Parameter-Fehler

unsupported_grant_type

`grant_type` ist ungleich `authorization_code`

invalid_grant

Das übergebene `refresh_token` ist ungültig oder abgelaufen.

unauthorized_client

Der `grant_type` ist für den authentifizierten Client nicht erlaubt

invalid_scope

Verwendung eines falschen Listenformats oder die angegebene Liste enthält eine oder mehrere unbekannte oder ungültige Scope bzw. die übergebene Liste gleicht nicht der Liste, die bei der Beantragung des verwendeten Autorisierungscodes verwendet wurde.

401 Unauthorized

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_client

Credentials fehlen, Client-ID ist ungültig oder Credentials sind nicht valide.

403 Forbidden

Es besteht keine Berechtigung, um auf die Ressource zuzugreifen.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_niveau

Die Anmeldung darf nur im hohen Authentifizierungsniveau erfolgen (diese Einstellung kann der Benutzer im Portal setzen)

415 Unsupported Media Type

Der Medientyp der Datei ist nicht erlaubt.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

unsupported_media_type

Der Content-Type der Anfrage wird nicht unterstützt.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Beispiel: Antwort im Erfolgsfall

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
   "access_token": "V0aG9yaXphdGlvbiBjb2Rl",
   "token_type": "Bearer",
   "expires_in": 3600,
   "id_level": "basicplus"
}
```

Beispiel: Antwort im Fehlerfall

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=UTF-8

{
   "error": "invalid_request",
   "error_description": "See URI below to get detailed information about
the error.",
   "error_uri": "http://tools.ietf.org/html/rfc6749#section-5.2"
}
```

4.2 Resource Owner Password Credentials Grant

Das Authentisierungs-Verfahren nach *Resource Owner Password Credentials Grant* wird angewendet, wenn zwischen dem Nutzer und der jeweiligen Anwendung (Client) eine vertrauensvolle Verbindung besteht, da die Zugangsdaten (Nutzername und Passwort) anwendungsseitig hinterlegt werden.

Voraussetzung: Vorlage eines Sicherheitskonzepts

Für die Implementierung dieses Verfahrens ist die Vorlage eines Sicherheitskonzepts notwendig:

- Wenden Sie sich an epost-partner@deutschepost.de oder nutzen Sie das Kontaktformular im Partnerportal (https://partner.epost.de/kontakt).

Sie erhalten daraufhin detaillierte Informationen für die Ausgestaltung des Sicherheitskonzepts.

Implementierung

Die Zugangsdaten bestehen aus der E-POSTBRIEF Adresse und dem zugehörigen Passwort. Durch die Authentisierung nach *Resource Owner Password Credentials Grant* ist es möglich, dass eine Applikation den Nutzer bei Bedarf und ohne dessen Einbindung anmelden kann.

HINWEIS

Beachten Sie Folgendes beim Implementieren des Verfahrens für die Versand-API:

Mit diesem Authentisierungs-Verfahren erfolgt eine Anmeldung mit dem normalen Authentifizierungsniveau. Für das Versenden von elektronischen E-POSTBRIEFEN ist allerdings ein hohes Authentifizierungsniveau notwendig. Wenn Sie dieses Authentisierungs-Verfahren nutzen, müssen Sie für elektronische E-POSTBRIEFE einen Vorgang zur Erhöhung des Authentifizierungs-Niveaus implementieren (siehe 4.3 Authentifizierungsniveau erhöhen).

ACHTUNG

Wenn Sie benutzerspezifische Daten wie den Nutzernamen und das Passwort lokal oder remote speichern, müssen Sie die Daten mit einer Verschlüsselung versehen.

Anfrage

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

Ziel der Anfrage

Das Ziel der Anfrage ist der *Token Endpoint* (siehe http://tools.ietf.org/html/rfc6749#section-3.2), der unter dem Pfad `/oauth2/tokens/` erreichbar ist.

URL-Struktur

```
POST /oauth2/tokens/
```

HTTP-Header

Content-Type (erforderlich)

`application/x-www-form-urlencoded`

Authorization (erforderlich)

Autorisierungs-Kopfzeile gemäß *HTTP Basic authentication scheme* (siehe http://tools.ietf.org/html/rfc6749#section-2.3.1 und http://www.ietf.org/rfc/rfc2617.txt Kapitel 2). Die verwendeten Werte entstammen der bei der Registrierung für E-POSTBUSINESS API angegebenen Daten.

Die Identifikation des Benutzers wird durch das Zusammenfügen des JSON-Feldes `DevId` mit einem Komma und dem JSON-Feld `AppId` gebildet. Der resultierende String wird mit dem `application/x-www-form-urlencoded`-Algorithmus aus http://tools.ietf.org/html/rfc6749#appendix-B kodiert. Dieser kodierte Wert wird als Benutzername verwendet, der Inhalt der LIF-Datei wird mit demselben Algorithmus kodiert und als Passwort benutzt.

Nach dem Zusammenfügen des Benutzernamens, einem „:“-Zeichen und dem Passwort wird der Wert mit Base64-Algorithmus kodiert. Dem resultierenden String wird das Wort `Basic` mit einem Leerzeichen vorangestellt und als Wert für den HTTP-Header `Authorization` benutzt.

Bildungsvorschrift:

- Basic = basic-credentials
- basic-credentials = base64-client-pass
- base64-client-pass = <base64 encoding of client-pass>

- client-pass = encoded-clientid ":" encoded-password
- encoded-clientid = <x-www-form-urlencoding of clientid>
- encoded-password = <x-www-form-urlencoding of password>
- clientid = devid "," appid
- devid = <DevId aus LIF-Datei>
- appid = <AppId aus LIF-Datei>
- password = <Inhalt der LIF-Datei>

HTTP-Body Die Client-Applikation sendet die folgenden Parameter im Format `application/x-www-form-urlencoded` mit `UTF-8`-Codierung im *entity-body* der HTTP-Anfrage.

ACHTUNG

Der Inhalt des Bodys muss URL-kodiert werden (siehe: http://de.wikipedia.org/wiki/URL-Encoding). Dies ist notwendig, da sonst Passwörter mit Sonderzeichen falsch übertragen werden und somit eine Authentifizierung nicht möglich ist. Gut gewählte Passwörter bestehen üblicherweise aus einer Reihe von Sonderzeichen wie z. B. **%**. Da **%** eine besondere Bedeutung hat, muss das Zeichen mit **%25** kodiert werden. Somit wird z. B. aus dem Passwort **G$eHelmNi%S** die URL-kodierte Zeichenkette **G%24eHelmNi%25S**.

grant_type (erforderlich)

Hier muss der feste Wert `password` gesetzt werden, wie in RFC 6749 spezifiziert (siehe http://tools.ietf.org/html/rfc6749#section-4.3.2).

username (erforderlich)

Nutzername

password (erforderlich)

Passwort des Nutzers

scope (erforderlich)

Entgegen der RFC 6749-Spezifikation ist dies ein zwingender Parameter. Es können mit dem Authentisierungs-Verfahren nach *Resource Owner Password Credentials Grant* folgende Werte angegeben werden:

- `send_hybrid`
- `read_letter`
- `create_letter`
- `delete_letter`
- `safe`

HINWEIS

Das Setzen des Wertes `send_letter` ist nicht möglich, da mit dieser Methode zurzeit kein hohes Authentifizierungsniveau möglich ist.

Beispiel (ein Scope)

```
POST /oauth2/tokens/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

grant_type=password&username=john.doe@testingepost.de&password=********&scope=send_hybrid
```

Beispiel (mehrere Scopes)

```
POST /oauth2/tokens/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

grant_type=password&username=john.doe@testingepost.de&password=********&scope=send_hybrid+delete_letter
```

Antwort

200 OK

Im Erfolgsfall wird Ihrer Applikation ein Ticket in Form eines JSON-Objekts zurückgegeben, welches ein Access Token enthält.

- Je nach angegebenen Scope können Sie die Versand-API oder Mailbox-API Ressourcen für den Versand bzw. Zugriff verwenden sowie auf Dokumente oder Verzeichnisse über die Safe-API zugreifen oder ändern.

HINWEIS

Die Antwort enthält im Gegensatz zur OAuth2.0-Spezifikation kein eigenes Refresh Token für die Aktualisierung. Um dies umzusetzen, wird das bereits bestehende Access Token erneuert.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit dem Ticket.

Das Ticket ist folgendermaßen aufgebaut:

access_token

Das Access Token in Form eines Strings.

expires_in

Gültigkeitsdauer des Access Tokens in Sekunden.

Das Access Token hat eine begrenzte Gültigkeitsdauer, die von der Login-API bestimmt wird.

token_type

Der Parameter wird vom Standard verlangt und ist bei der E-POSTBUSINESS API immer `Bearer`.

id_level

Das Identitätsniveau des angemeldeten E-POST Nutzers (siehe 3. Identitätsniveau von Privatkunden).

Mögliche Werte sind `basic`, `basicplus`, `premium`, `premiumplus` und `null`. Der Wert `null` tritt auf, wenn es sich um einen Geschäftskunden handelt.

HINWEIS

Wenn das Access Token seine Gültigkeitsdauer überschritten hat, wird die Anfrage mit dem entsprechenden Hinweis `invalid_token` abgelehnt. Danach kann Ihre Applikation das Token für einen längeren Zeitraum bei der Login-API erneuern. Dafür muss Ihre Applikation das bestehende Access Token verwenden. Ein gesondertes Refresh Token wird nicht ausgehändigt.

ACHTUNG

Stellen Sie sicher, dass das Access Token sicher in der Applikation gehalten wird und kein Fremdzugriff darauf erfolgen kann.

400 Bad Request

Die Anfrage ist fehlerhaft aufgebaut.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_request

Parameter-Fehler

unsupported_grant_type

`grant_type` ist ungleich `authorization_code`

invalid_grant

Die übergebenen Nutzer-Zugangsdaten sind ungültig.

unauthorized_client

Der `grant_type` ist für den authentifizierten Client nicht erlaubt

invalid_scope

Verwendung eines falschen Listenformats oder die angegebene Liste enthält eine oder mehrere unbekannte oder ungültige Scope bzw. die übergebene Liste gleicht nicht der Liste, die bei der Beantragung des verwendeten Autorisierungscodes verwendet wurde.

401 Unauthorized

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_client

Credentials fehlen, Client-ID ist ungültig oder Credentials sind nicht valide.

403 Forbidden

Es besteht keine Berechtigung, um auf die Ressource zuzugreifen.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_niveau

Die Anmeldung darf nur im hohen Authentifizierungsniveau erfolgen (diese Einstellung kann der Benutzer im Portal setzen)

415 Unsupported Media Type

Der Medientyp der Datei ist nicht erlaubt.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

unsupported_media_type

Der Content-Type der Anfrage wird nicht unterstützt.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Beispiel: Antwort im Erfolgsfall

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{
   "access_token": "V0aG9yaXphdGlvbiBjb2Rl",
   "token_type": "Bearer",
   "expires_in": 3600,
   "id_level": "basicplus"
}
```

Beispiel: Antwort im Fehlerfall

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=UTF-8

{
   "error": "invalid_request",
   "error_description": "See URI below to get detailed information about
the error.",
   "error_uri": "http://tools.ietf.org/html/rfc6749#section-5.2"
}
```

4.3 Authentifizierungsniveau erhöhen

Um mit der Versand-API elektronische E-POSTBRIEFE versenden zu können, muss der Nutzer mit hohem Authentifizierungsniveau angemeldet sein. Der Begriff „hohes Authentifizierungsniveau“ bedeutet dabei, dass der Nutzer für den Versand von E-POSTBRIEFEN neben Benutzername und Passwort zusätzlich eine *HandyTAN* eingeben muss.

Zu einem bestehenden Access Token erhält der Nutzer dabei einen Authorization Code mit erweitertem Scope. Mit diesem Authorization Code kann das E-POST System ein neues Access Token ausstellen.

Das Ergebnis dieser Anfrage ist zunächst ein Redirect auf das HandyTAN-Formular. Dieses Formular muss dem Nutzer zur Eingabe angezeigt werden. Der Scope ergibt sich dabei aus dem bestehenden Scope des ursprünglichen Access Token und dem hinzugefügten Scope (`send_letter`). Der Code wird mittels des gegebenen Access Token erzeugt.

Der Nutzer gibt dann die HandyTAN in das Formular ein und sendet sie. Die Login-API validiert die HandyTAN, antwortet der Applikation mit einer Redirect-URL und übergibt einen Autorisierungscode. Die Applikation extrahiert den Autorisierungscode und fordert ein Access Token an.

Anfrage

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

URL-Struktur

```
POST /oauth2/auth/extension
```

HTTP-Header

Content-Type (erforderlich)

`application/x-www-form-urlencoded;charset=UTF-8`

HTTP-Body

Die Client-Applikation sendet die folgenden Parameter im Format `application/x-www-form-urlencoded` mit `UTF-8`-Codierung im *entity-body* der HTTP-Anfrage:

response_type (erforderlich)

Ein String der festlegt, welche Berechtigung der Client für die Autorisierung benötigt. Zurzeit wird ausschließlich der Wert `code` unterstützt.

client_id (erforderlich)

Ein String, der die Client-Applikation identifiziert.

HINWEIS

Die `client_id` ergibt sich aus `DevId` und `AppId`, die Sie bei der Registrierung für E-POSTBUSINESS API angeben. Die Bildungsvorschrift der `client_id` ist folgende:

- clientid = devid "," appid

redirect_uri (erforderlich)

Die registrierte (auf dem Server hinterlegte) Redirect-URI des Partners.

state (optional)

Ein beliebiger String von maximal 512 Zeichen Länge, der den Zustand der Applikation beschreibt. Der Wert wird der Applikation im Login-Ergebnis zurückgegeben.

ACHTUNG

Ihre Applikation muss diesen Parameter angeben, um *Cross-Site-Request-Forgery* zu verhindern, auch wenn der Parameter nicht als Pflicht-Wert erwartet wird. Weitere Informationen siehe http://tools.ietf.org/html/draft-ietf-oauth-v2-30#section-10.12.

TIPP

Es kann mit diesem Parameter z. B. eine *gehashte* Session-ID oder ein zufallsgenerierter Wert übergeben werden.

scope (optional)

Ein String, der die Erweiterung des Scopes um den Standardwert `send_letter` angibt. Der Parameter muss nicht eigens angegeben werden.

access_token (erforderlich)

Das Access Token des normalen Authentifizierungs-Niveaus in Form eines Strings.

Beispiel

```
POST /oauth2/auth/extension HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Content-Length: 999
  response_type=code
    &client_id=TestClientId
    &state=TestState
    &scope=send_letter
    &redirect_uri=https://localhost:8888/briefkasten
    &access_token=dGhpcyBpcyBhbiBiYXN
```

Antwort

302 Found (Erfolgsfall)

Im Erfolgsfall antwortet die Login-API mit einem Redirect auf die E-POST *TAN-Seite*.

Die Antwort enthält folgende Parameter:

client_id

Der zurückgegebene Autoriorisierungscode.

state

Der bei der Anfrage übergebene Wert wird unverändert durchgereicht.

scope

Scopes des Access Token im normalen Authentifizierungsniveau sowie die Erweiterung `send_letter`.

redirect_uri

Redirect zum Eingabefenster für die HandyTAN.

Rückgabe des Access Token:

Im Erfolgsfall geht der Prozess wie folgt weiter:

1. Ihre Applikation zeigt dem Nutzer die E-POST TAN-Seite.
2. Der Nutzer gibt die erhaltene HandyTAN ein und sendet sie.
3. Die Login-API validiert die HandyTAN, antwortet Ihrer Applikation mit einer Redirect-URL und liefert einen Autorisierungscode zurück.
4. Ihre Applikation fordert den Access Token über die Login-API an (siehe 4.1.2 Access Token anfordern).

302 Found (Fehlerfall)

Im Fehlerfall wird Ihrer Applikation eine Redirect-URL mit der entsprechenden Fehlermeldung zurückgegeben. Es wird dabei ein Fehlercode (`error`), eine optionale Fehlerbeschreibung (`error_description`), ein optionaler Hinweis, unter welcher URL eine detaillierte Fehlerbeschreibung abgerufen werden kann (`error_uri`) sowie der in der Anfrage übergebene Wert (`state`) zurückgegeben.

error

invalid_scope

Keiner der angegebenen Scopes erfordert das hohe Authentifizierungsniveau.

access_denied

Das übergebene Access Token ist abgelaufen.

error_description

Für Menschen lesbarer Text mit zusätzlichen Informationen, um den Entwickler beim Verstehen des aufgetretenen Fehlers zu unterstützen.

error_uri

Eine URI, die eine für Menschen lesbare Webseite mit Informationen über den Fehler informiert.

state

Der in der Anfrage übergebene Wert wird unverändert durchgereicht.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Beispiel: Antwort im Erfolgsfall

```
HTTP/1.1 302 Found
Set-Cookie: code=SDFGSDHTZJw45ggw
Location: https://login.epost.de/oauth2/auth/tan?response_type=code
 &client_id=TestClientId
 &state=TestState
 &scope=send_letter send_hybrid create_letter
 &redirect_uri=https://localhost:8888/link/zum/tan_formular
```

Beispiel: Antwort im Fehlerfall

```
HTTP/1.1 302 Found
Location: https://localhost:8888/briefkasten?
 error=access_denied&
 error_description="The resource owner or authorization server denied the
request."&
 error_uri="http://tools.ietf.org/html/rfc6749#section-4.1.2"&
 state=TestState
```

4.4 Abmeldung

Ein gültiges Access Token kann von der Anwendung zur Abmeldung (Invalidierung) eingereicht werden. Das gesendete Access Token kann anschließend nicht mehr für Anfragen gegen die Versand-API, Mailbox-API oder Safe-API verwendet werden.

Anfrage

Die Anfrage zur Abmeldung enthält das abzumeldende Token als Parameter `access_token`. Der Aufruf ist ebenfalls für bereits abgelaufende Access Tokens erfolgreich.

Zugriff auf Login-URL

Ihre Applikation muss Zugriff auf die Login-URL haben:

- https://login.epost.de (Produktionsumgebung)
- https://login.epost-gka.de (Test- und Integrationsumgebung)

URL-Struktur

```
POST /oauth2/tokens/logout
```

HTTP-Header

Content-Type (erforderlich)

```
application/x-www-form-urlencoded;charset=UTF-8
```

Pfad-Parameter

access_token (erforderlich)

Das aktuelle Access Token.

Beispiel

```
POST /oauth2/tokens/logout HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=UTF-8

access_token=dGhpcyBpcyBhbiBiYXN...
```

Antwort

204 No Content

Als Antwort im Erfolgsfall wird eine HTTP-Nachricht mit Statuscode 204 und leerem Body zurückgegeben. Damit ist der Logout erfolgt.

400 Bad Request

Die Anfrage ist fehlerhaft.

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

invalid_input

Das übergebene Access Token fehlt oder ist im falschen Format.

401 Unauthorized

HTTP-Header

```
Content-Type: application/json; charset=UTF-8
```

HTTP-Body

Der Body enthält ein JSON-Objekt mit Fehlerinformationen.

Mögliche Werte für `error`:

unauthorized

Das Access Token ist abgelaufen.

500 Internal Server Error

Interner Verarbeitungsfehler

503 Service Unavailable

Interner Verarbeitungsfehler

Bei Fragen zur E-POSTBUSINESS API finden Sie weitere Informationen auf dem E-POST Partnerportal unter http://partner.epost.de
Bei allgemeinen Fragen zur E-POST unterstützt Sie gerne der Kundenservice der Deutschen Post AG:

- Tel.: +49 228 76 36 76 06, Mo – Fr 8.00 – 20.00 Uhr (außer an bundeseinheitlichen Feiertagen)
- E-POSTBRIEF: Service@dpdhl.epost.de
- E-Mail: service@deutschepost.de

Deutsche Post AG
Charles-de-Gaulle-Straße 20
53113 Bonn

www.deutschepost.de

Stand 07/2015