Methoden

Im folgenden sind alle implementierten Methoden definiert. Sollten Sie Daten aus dem System benötigen, welche nicht über die Schnittstelle freigegeben werden, kontaktieren Sie bitte Ihren Ansprechpartner.

Meldescheine buchen

```
POST  import/meldeschein/buchen/
```

Dieser Aufruf erzeugt einen oder mehrere elektronischen Meldeschein im System.

Allgemein

Es ist möglich mehrere Meldescheine auf einmal zu buchen, indem weitere Objekte dem `meldescheine` Array hinzugefügt werden.

Parameter

Name	Datentyp	Verwendung
meta	object	Meta-Objekt
params	object	Optionale Parameter
meldescheine	array	Array mit Meldescheindaten
meldescheine[]	object	Die zu buchenden Daten des Meldescheins
meldescheine[].personen	array	Auflistung aller Gäste
meldescheine[].ms_type	string	Immer mit dem Wert "ems" zu befüllen

Pro Person

Name	Datentyp	Verwendung
meldescheine[].personen[].tarif_id	int	Die Tarif-ID des Gasts
meldescheine[].personen[].gast_type	string	Gast-Art (`person`, `gruppe`)
meldescheine[].personen[].arrival_date	date	Anreisedatum des Gasts

Name	Datentyp	Verwendung
meldescheine[].personen[].departure_date	date	Abreisedatum des Gasts

Einzel-Person

Name	Datentyp	Verwendung
meldescheine[].personen[].*	string	Weitere definierte Felder. (nach Definition)

> Die Definition der Felder hängt von der Gemeinde ab und können über den Aufruf `/get_current_user` abgefragt werden.

Reisegruppe

Name	Datentyp	Verwendung
meldescheine[].personen[].anzahl	int	Anzahl der Gäste

> Wenn das Feld meldescheine[].personen[].anzahl größer 1 ist ,wird der Meldeschein als Gruppenmeldeschein verbucht. Eventuell übergebene Einzelinformationen für den Gast (Vorname,Nachname etc.) werden nicht gespeichert.

Optionale Parameter (`params`)

Name	Datentyp	Beschreibung
append_pdf	bool	Für bei der Response jedem Meldeschein-Objekt den Parameter "pdf" .

`params.append_pdf`

Dieser Schalter ermöglicht es, jedem Meldeschein-Objekt direkt das PDF anzufügen.
In der Server-Antwort wird das Feld `pdf` angefügt, welches den Base64-kodierten Meldescheine enthält.

> Das aktivieren dieses Schalters gilt als Druckvorgang und wird entsprechend protokolliert - siehe Erklärung hier.

Beispiel

```
{
  "meta": {
    "gemeinde": 1,
    "requestId": "123-456-789",
    "idempotent": true,
    "objekt": 2049
  },
  "params": {
    "append_pdf": true
  },
     ....
 }
```

```
{
    "meta": {
        "gemeinde": 1,
        "requestId": "123-456-789",
        "idempotent": true,
        "objekt": 2049,
        "timestamp": "2020-12-18T11:51:41.154820",
        "user": {
            "id": 38,
            "username": "DemoV",
            "alias": "Lisa Mustermann"
        }
    },
    "status": "success",
    "meldescheine": [
        {
            "meldeschein": {
                "id": 28,
                //....
            },
            "id": 28,
            "pdf": "JVBERi0xLjMKMSAwIG9iago8PAov....."
        }
    ]
}
```

Server Antwort

Die Serverantwort gibt im Erfolgsfall ein Array aus `meldeschein` Objekten mit der dem gesamten Meldeschein-Objekt zurück.

> Der Anwort-Aufbau entspricht dem aus dem Meldeschein aktuaklisieren /get Aufruf

> Die Reihenfolge der Personen ist zwischen Anfrage und Antwort immer gleichbleibend. Somit können Sie die die ID's der Personen in Ihrem System für spätere Aktualisierungen mitführen

Response:

JSON


```
{
  "meta": {
    "requestId": "5ca0b885-11e3-4abd-9e60-cb8265ea9899",
    "timestamp": "2019-10-24T14:06:10.673840",
    "user": {
      "id": 14,
      "username": "api_demo",
      "first_name": "API",
      "last_name": "Demo",
      "email": "api_demo@apidemo.de",
      "is_active": true,
      "user_type": "beherberger",
      "alias": "API Demo",
      "last_login": "2019-10-24T11:35:18.015947+02:00"
    },
    "gemeinde": 1
  },
  "status": "success",
  "meldescheine": [
    {
      "id": 396,
      //.....
      "personen": [
        {
          "id":3334,
          "tarif_id": 1,
```

```
          "Gast_Vorname": "Max",
          "Gast_Name": "Mustermann",
          "arrival_date": "2019-01-01",
          "departure_date": "2019-01-15"
        }
      ]
    }
  ]
}
```


Beispiel-Buchungen

Beispiele für verschiedene Szenarien

Personen-Meldeschein

Folgende Beispiele zeigen jeweils die Buchung eines Scheins. Da die Rückantwort des Servers ist unabhängig des Typs immer gleich.

Request:

JSON

```
{
  "meta": {
    "vermieter": 1,
    "gemeinde": 1,
    "objekt": 1
  },
  "meldescheine": [
    {
      "input_id": "test-1",
      "personen": [
        {
          "gast_type": "person",
          "tarif_id": 1,
          "Gast_Vorname": "TEST_BF",
          "arrival_date": "2019-07-14",
          "departure_date": "2019-07-17"
        }
      ]
    }
  ]
}
```


XML

Gruppen-Meldeschein

Request:

JSON

```
{
  "meta": {
    "vermieter": 1,
    "gemeinde": 1,
    "objekt": 1
  },
  "meldescheine": [
    {
      "input_id": "test-1",
      "personen": [
        {
          "gast_type": "gruppe",
          "anzahl": 5,
          "tarif_id": 1,
          "arrival_date": "2019-07-14",
          "departure_date": "2019-07-17"
        }
      ]
    }
  ]
}
```

XML

Code-Beispiele

Folgendes Beispiel authentifiziert die Anfrage und bucht einen Meldeschein

Python 3

```
# This example uses the "requests" library for HTTP requests
(https://requests.readthedocs.io/en/latest/)
# You can install it via
# $ pip3 install requests
import requests
payload = {
    'username':'api_demo',
    'password':'demodemo',
```

```
}

r = requests.post('https://apiv2.meldescheine.de/api/token/',data=payload)
server_response_json = r.json()



jwt_token = server_response_json.get('token') # Aus dem server_response objekt den String-Wert
aus "token" extrahieren

# Jetzt buchen
meldeschein_data = {} # Leer für Test
headers = {
 'Authorization':  "JWT %s" % jwt_token
}
r =
requests.post("https://apiv2.meldescheine.de/api/import/meldeschein/buchen/",data=meldeschein_
data,headers=headers)
```

Meldescheine aktualisieren

Dokumentation unvollständig.

Allgemein

Dieser Aufruf aktualisiert einen existierenden elektronischen Meldeschein im System.

Es kann pro Aufruf immer nur ein (1) Meldeschein aktualisiert werden.
Wenn im unten erläuterten Array mehrere Meldescheine aufgeführt sind, wird der erste Meldeschein (Index:0) aktualisiert.

Existierende Meldescheine müssen immer mindestens einen Gast besitzen. Es ist nicht möglich einen Meldeschein zu "leeren", indem alle Gäste gelöscht werden.

Beschränkungen

Ein Meldeschein kann nur verändert werden, solange sich sein Status in "Entwurf" oder "gedruckt" befindet. Ist ein Meldeschein abgereist, sind keine weiteren Änderungen möglich.

"abgereist" ist wie folgt definiert: Alle Gäste des Meldescheins sind abgereist. Ein automatischer Prozess auf System-Ebene prüft alle aktuellen Meldescheine und Gäste anhand ihres Abreisedatums. Sind alle Gäste abgereist, so erhält der Meldeschein den Status "abgereist". Der Meldeschein gilt somit für die Gemeinde als "bereit zur Abrechnung."

Änderungen an diesen Meldescheinen können nur über die Gemeinde angefordert werden.

Parameter

```
POST import/meldeschein/aktualisieren/
```

Name	Datentyp	Verwendung

meta	object	Meta-Objekt
meldescheine	array	Array mit Meldescheindaten
meldescheine[]	object	Die zu buchenden Daten des Meldescheins
meldescheine[].personen	array	Auflistung aller Gäste

Der Aufbau der Anfrage entspricht exakt des Aufrufs "buchen" mit dem Unterschied, dass im Meldeschein-Objekt die ID aus dem Aufruf "buchen" mit übergeben werden muss, um den zu aktualisierenden Meldeschein eindeutig zu identifizieren.

Zum aktualisieren von Gästen muss das Feld "id" mit übergeben werden

Server Antwort

Die "aktualisieren" Antwort enthält den aktualisieren Meldeschein als vollständiges Objekt. Die Antwort ist unabhängig der ausgeführten Aktion immer im gleichen Format und spiegelt den aktuellen Status des Meldescheins im System wider.

Da nur derzeit ein Meldeschein aktualisiert werden kann, befindet sich der aktualisierten Meldeschein im Objekt `response.meldescheine[0]`

Die Antwort gibt den aktuellen Status des Meldescheins im System zurück - hier können Sie ggfls. prüfen, ob Änderungen erfolgreich übernommen wurden.


```
{
    "meta": {
        "requestId": "afc8673a-2a4c-11eb-b5a3-c04a00212a69",
        "timestamp": "2020-11-19T10:50:41.996029",
        "user": {
            "id": 38,
            "username": "DemoV",
            "alias": "Lisa Mustermann"
        },
        "gemeinde": 1
```

```
},
"status": "success",
"response": {
    "meldescheine": [
        {
            "id": 688,
            "personen": [
                {
                    "id": 2015,
                    "tarif_id": 1,
                    "meldeschein_id": 688,
                    "gast_type": "person",
                    "created": "2020-11-19T10:48:47.228504+01:00",
                    "modified": "2020-11-19T10:50:42.311202+01:00",
                    "anzahl": 1,
                    "arrival_date": "2019-07-14",
                    "departure_date": "2019-07-17",
                    "CSID": "6b59af3c-2a4c-11eb-aa07-c04a00212a69-WLclSDowcn",
                    "kurtaxe_calc": 6.0,
                    "Gast_Vorname": "aktualisiert?",
                    "Gast_Name": null,
                    "Gast_Geburtsdatum": null,
                    "Gast_Strasse": null,
                    "Gast_Postleitzahl": null,
                    "Gast_Wohnort": null,
                    "Gast_Land": null,
                    "Gast_Email": null,
                    "Gast_Ausweisnummer": null,
                    "Gast_Staatsangehoerigkeit": null,
                    "address_id": null,
                    "address_lat": null,
                    "address_lng": null,
                    "address_label": null
                },
                {
                    "id": 2016,
                    "tarif_id": 1,
                    "meldeschein_id": 688,
                    "gast_type": "person",
                    "created": "2020-11-19T10:48:47.571441+01:00",
```

```
        "modified": null,
        "anzahl": 1,
        "arrival_date": "2019-07-14",
        "departure_date": "2019-07-17",
        "CSID": "6b59af3c-2a4c-11eb-aa07-c04a00212a69-WLclSDowcn",
        "kurtaxe_calc": 6.0,
        "Gast_Vorname": "TEST_BF_2",
        "Gast_Name": null,
        "Gast_Geburtsdatum": null,
        "Gast_Strasse": null,
        "Gast_Postleitzahl": null,
        "Gast_Wohnort": null,
        "Gast_Land": null,
        "Gast_Email": null,
        "Gast_Ausweisnummer": null,
        "Gast_Staatsangehoerigkeit": null,
        "address_id": null,
        "address_lat": null,
        "address_lng": null,
        "address_label": null
    },
    {
        "id": 2017,
        "tarif_id": 1,
        "meldeschein_id": 688,
        "gast_type": "person",
        "created": "2020-11-19T10:48:47.611475+01:00",
        "modified": null,
        "anzahl": 1,
        "arrival_date": "2019-07-14",
        "departure_date": "2019-07-17",
        "CSID": "6b59af3c-2a4c-11eb-aa07-c04a00212a69-WLclSDowcn",
        "kurtaxe_calc": 6.0,
        "Gast_Vorname": "TEST_BF_3",
        "Gast_Name": null,
        "Gast_Geburtsdatum": null,
        "Gast_Strasse": null,
        "Gast_Postleitzahl": null,
        "Gast_Wohnort": null,
        "Gast_Land": null,
```

```
            "Gast_Email": null,
            "Gast_Ausweisnummer": null,
            "Gast_Staatsangehoerigkeit": null,
            "address_id": null,
            "address_lat": null,
            "address_lng": null,
            "address_label": null
        },
        {
            "id": 2018,
            "tarif_id": 1,
            "meldeschein_id": 688,
            "gast_type": "person",
            "created": "2020-11-19T10:48:47.655957+01:00",
            "modified": null,
            "anzahl": 1,
            "arrival_date": "2019-07-14",
            "departure_date": "2019-07-17",
            "CSID": "6b59af3c-2a4c-11eb-aa07-c04a00212a69-WLclSDowcn",
            "kurtaxe_calc": 6.0,
            "Gast_Vorname": "TEST_BF_4",
            "Gast_Name": null,
            "Gast_Geburtsdatum": null,
            "Gast_Strasse": null,
            "Gast_Postleitzahl": null,
            "Gast_Wohnort": null,
            "Gast_Land": null,
            "Gast_Email": null,
            "Gast_Ausweisnummer": null,
            "Gast_Staatsangehoerigkeit": null,
            "address_id": null,
            "address_lat": null,
            "address_lng": null,
            "address_label": null
        }
    ],
    "travel_span": {
        "arrival_date": "2019-07-14T00:00:00",
        "departure_date": "2019-07-17T00:00:00",
        "travel_days": 3,
```

```
        "uebernachtungen_sum": 12
    },
    "objekt": {
        "id": 2049,
        "beherberger_id": 13455,
        "beherberger": {
            "firma": "Ferienhof Müller",
            "vorname": "Melissa",
            "nachname": "Müller",
            "comment": "<br>",
            "debitor_nr": "08154711",
            "alias": "Ferienhof Müller (08154711)",
            "id": 13455,
            "ansprechpartner": 37,
            "gemeinde_id": 1
        },
        "beherberger_alias": "Ferienhof Müller (08154711)",
        "ortsteil": {
            "id": 1,
            "alias": "Standard",
            "gemeinde": 1
        },
        "created": "2020-03-30T08:34:03.052153+02:00",
        "modified": "2020-07-21T13:16:53.520759+02:00",
        "anzahl_betten": 4,
        "alias": "Haus Sonne",
        "strasse": "Edgar-Anstett-Straße 4",
        "plz": "66978",
        "ort": "Merzalben",
        "tarifzone": 1,
        "unterkunftsart": 1
    },
    "meldescheinnummer": "688",
    "objekt_id": 2049,
    "beherberger_id": 13455,
    "verwendete_kurtaxe": 24.0,
    "creation_user": {
        "id": 38,
        "username": "DemoV",
        "alias": "Lisa Mustermann"
```

```
                },
                "precheck_errors": [],
                "state_display": "Entwurf",
                "faktura_date": null,
                "CSID": "6b59af3c-2a4c-11eb-aa07-c04a00212a69-WLclSDowcn",
                "Form_Id": null,
                "BatchNo": null,
                "BatchRDate": "2020-11-19",
                "BatchPgDta": null,
                "BatchTrack": null,
                "ms_type": "ems",
                "state": "draft",
                "created": "2020-11-19T00:00:00+01:00",
                "modified": null,
                "user_note": null,
                "is_storniert": false,
                "source": null,
                "storno_bestaetigt": false,
                "kurtaxe_ist": null,
                "manueller_betrag": 0.0,
                "kz_manueller_betrag": false,
                "kurtaxe_calc": 24.0,
                "stat_alter_kind_1": null,
                "stat_alter_kind_2": null,
                "stat_alter_kind_3": null,
                "stat_alter_kind_4": null,
                "invoice": null
            }
        ]
    }
}
```


Anlegen von zusätzlichen Gästen

Um dem Meldeschein weitere Gäste hinzuzufügen, können diese ohne Angabe des Feldes "id" in das "personen" Array mit übergeben werden.

> Für das anlegen von zusätzlichen Gästen gelten die Regeln wie beim anlegen eines Meldescheins. Die Felder "tarif_id" , "arrival_date" sowie "departure_date" sind zwingend notwendig. Zusätzliche Felder analog zum buchen Aufruf und der eingerichteten Gemeinde.

Beispiel-Request: Anlegen eines zusätzlichen Gastes:

"Max Mustermann"
Tarif-ID "1" (Erwachsener)
Reisezeitraum: 15.10.2020 - 16.10.2020
in Meldeschein 650

```
{
  "meta":  {
    "objekt":2080,
    "gemeinde":1,
□□"requestId":"TEST"
  },
  "meldescheine": [
    {
□□□"id": 650,
□□□"personen":[
□□□□{
□□□□□"arrival_date": "2020-10-15",
□□□□□"departure_date":"2020-10-16",
□□□□□"tarif_id":1,
□□□□□"Gast_Vorname":"Max",
□□□□□"Gast_Name":"Mustermann"
□□□□}
□□□]
□□}
  ]
}
```

Aktualisieren von vorhandenen Gästen

Um Gäste zu aktualisieren, da sich z.B. Reisedaten geändert haben ist es notwendig, die "id" des Gastes mit zu übergeben.

Diese kann z.B. über den "get-data" Aufruf abgerufen werden oder kann beim buchen der Meldescheine sofort gespeichert werden.

Beispiel: Ändern des im vorherigen Beispiel "Anlegen von zusätzlichen Gästen" erzeugten Gastes "Max Mustermann":.

In diesem Fall wird der Reisezeitraum geändert: 16.10.2020 bis 19.10.2020 sowie der Nachname des Gastes.

Es werden nur Felder aktualisiert, die übergeben werden und im System vorhanden sind. Achten Sie daher bitte darauf, nur die notwendigen Felder zu übergeben und achten Sie auf korrekte Feldbezeichnungen

Zu beachten die ID "1945" des Gastes im unteren Beispiel.
- Wird die ID angegeben: Gast aktualisieren
- Keine Angabe der ID: Gast anlegen

```
{
  "meta":  {
    "objekt":2080,
    "gemeinde":1,
"requestId":"TEST"
  },
  "meldescheine": [
    {
"id": 650,
"personen":[
{
"id":1945, //<------------------ ID des Gastes (z.B. über /get-data abgerufen
"arrival_date": "2020-10-16",
"departure_date":"2020-10-18",
"tarif_id":1,
"Gast_Vorname":"Max",
"Gast_Name":"Mustermann (verändert)"
}
]
}
  ]
}
```

Löschen von vorhandenen Gästen

Hinweis zum aktualisieren von Gästen:
Bitte verwenden Sie immer den Ablauf "aktualisieren" um Gäste zu verändern. Bitte löschen Sie nicht alle Gäste und legen diese neu an.
Dies ist technisch in einem kombinierten Aufruf (s.u.) zwar möglich - jedoch erschwert dies

die Nachvollziehbarkeit auf System-Ebene, da zu jedem Gast eine Historie angelegt wird.

Gäste können gelöscht werden, in dem diese mit ihrer ID sowie dem Feld "delete:true" übergeben werden.

Beispiel-Request: Löschen des Gastes mit id `1937` aus Meldeschein mit ID `651`

```
{
  "meta":  {
    "objekt":2080,
    "gemeinde":1,
□□"requestId":"TEST"
  },
  "meldescheine": [
    {
□□□"id": 651,
□□□"personen":[
□□□□{
□□□□□"id": 1937,
□□□□□"delete":true  //<------ Gibt an, dass der Gast beim Aufruf gelöscht werden soll
□□□□}
□□□]
□□}
  ]
}
```

Wenn Sie versuchen einen Gast zu löschen, der nicht dem Schein zugeordnet ist, erhalten Sie eine entsprechende Fehlermeldung

Verschiedene Gast-Aktionen in einem Aufruf

Es ist möglich die oben beschriebenen Aktionen in einem Aufruf zu kombinieren:

Beispiel:

Eine Kombination aus den og. Beispielen

- Gast `1945` aktualisieren
- Gast `1937` löschen

- Einen neuen Gast anlegen


```
{
  "meta":  {
    "objekt":2080,
    "gemeinde":1,
"requestId":"TEST"
  },
  "meldescheine": [
    {
"id": 650,
"personen":[
{
"id":1945,
"arrival_date": "2020-10-16",
"departure_date":"2020-10-18",
"tarif_id":1,
"Gast_Vorname":"Max",
"Gast_Name":"Mustermann (verändert)"
},
{
"id": 1937,
"delete":true
},
{
"arrival_date": "2020-10-15",
"departure_date":"2020-10-16",
"tarif_id":1,
"Gast_Vorname":"Max",
"Gast_Name":"Mustermann"
}

]
}
  ]
}
```

Meldescheindaten abrufen

Dieser Aufruf frägt alle hinterlegten Daten eines oder mehrerer Meldescheine ab.

```
POST  import/meldeschein/get/
```

Parameter

Name	Datentyp	Verwendung
meta	Objekt	Meta-Objekt
meldescheine	Array	Array mit Meldeschein-IDs

Beispiele

Folgendes Beispiel ruft die Daten von zwei Meldescheinen ab.

Request

JSON

```
{
  "meta": {
  //....
  },
  "meldescheine": [
    123,
    456
  ]
}
```

Response

JSON


```
{
  "meta": {
    "requestId": "0324b440-c0ec-4828-a13c-5b1b81797d78",
    "timestamp": "2019-06-13T11:55:26.039028",
    "user": {
      "id": 6,
      "username": "demo_hotel"
    },
    "gemeinde": 1
  },
  "response": [
    {
      "id":123,
      "personen": [
        {
          "tarif_id": 1,
          "Gast_Vorname": "Max",
          "Gast_Name": "Mustermann",
          "arrival_date": "2019-01-01",
          "departure_date": "2019-01-15"
        }
      ]
    },
    {
      "id": "456",
      "personen": [
        {
          "tarif_id": 2,
          "anzahl": 12
        }
       //...
]
    }
  ]
}
```

Meldeschein-Kurkarte abrufen (PDF)

Gibt die Kurkarten der angefragten Meldescheine als base64 kodiertes PDF zurück. Die Kurkarte ist druckfertig und muss i.d.R. nur noch als PDF-Datei ausgegeben und gedruckt werden.

> **Vermeiden Sie mehrfache Aufrufe der Schnittstelle**:
> Jede Anfrage "get-pdf" wird als Druckvorgang gewertet und entsprechend protokolliert.
>
> Sie gilt somit als physikalischer Druck einer Kurkarte. Achten Sie daher bitte darauf, die Anfrage nur auszuführen, wenn ein Druck der Gästekarte umgehend folgt.
>
> Sollte die PDF-Kurkarte mehrfach benötigt werden, speichern Sie bitte das PDF auf Ihrem System für eventuelle Verwendung ab.

Parameter

```
POST import/meldeschein/get-pdf/
```

Name	Datentyp	Verwendung
meta	object	Meta-Objekt
meldescheine	array	Array mit Meldeschein-IDs

Request

Folgendes Beispiel ruft die Daten von zwei Meldescheinen ab.

Request:

JSON

```
{
  "meta":  {
    "objekt": 2049,
    "gemeinde": 1
  },
  "meldescheine": [
    553
  ]
}
```

Response

JSON

```
{
    "meta": {
        "requestId": "66f81f62-0942-11eb-bd68-c04a00212a69",
        "timestamp": "2020-10-08T10:43:56.496102",
        "user": {
            "id": 38,
            "username": "DemoV",
            "alias": "Lisa Mustermann"
        },
        "gemeinde": 1
    },
    "response": [
        {
            "id": 553,
            "encoding": "base64",
            "pdf": "JVBERi0xLjMKMSAwIG9iaHMgWyAzIDAg....<abgeschnitten>"
        }
    ]
}
```

PDF dekodieren und speichern

Die Rückgabe des Servers enthält die PDF-Daten als `base64` kodierte `string`-Werte in der Response. Der'Wert ist jeweil ins `response[].pdf` zu finden.

Die übliche Vorgehensweise ist es, den base64 kodierten Wert als PDF-Datei zu speichern und diese dann zur weiteren Verwendung zu geben (z.B. Ausdruck Kurkarte).
Möglichkeiten Base64 Zeicheketten als Binärdateien zu speichern bieten sich in vielen Programmiersprachen.

Code-Beispiele

Python 3

```
import base64


# ....
# Die variable Antwort des Servers ist hier in der variable "r" zu finden

for ms in r['response']:

  base64_string = ms.get('pdf')
  filename =  './Beispiel-Meldeschein-%s.pdf'  % ms.get('id') # Datei als "Beispiel-
Meldeschein-<id>.pdf" speichern
  fh = open(filename,'wb')   #lokale Datei im binären Modus zum schreiben öffnen / erstellen.

  decoded = base64.decodebytes(base64_string) # Base64 String binär dekodieren
  fh.write(decoded) #  dekodierte Daten in Datei schreiben
  fh.close() # Datei schliessen
  # Die Datei <filename> kann nun gedruckt werden
```

Konfiguration abrufen

```
GET  users/get_current_user
```

Dieser Aufruf frägt alle Informationen zum aktuellen Benutzer und damit verknüpften Gemeinden sowie Beherbergern.

Dies ermöglicht es, in der Drittsoftware definierte Tarife, Meldeschein-Felder sowie weitere Informationen abzurufen und bereitzustellen.

Parameter

Keine Parameter notwendig

Aufbau Antwort

Allgemein

Feld	Datentyp	Erklärung
user.*	object	Informationen zum aktuell angemeldeten Benutzer (über Token)
beherberger	array	Dem Account zugeordnete Vermieter
beherberger[].*	object	Sämtliche Informationen zum Vermieter
beherberger[].objekte	object	Alle Objekte des Vermieters, welche dem Account zugeordnet sind
gemeinden	array	Alle dem Account zugeordneten Gemeinden (i.d.R. eine)
gemeinden[].ortsteile	array	Alle definierten Ortsteile der Gemeinde
gemeinden[].tarifzonen	array	Alle definierten Tarifzonen der Gemeinde

Feld	Datentyp	Erklärung
gemeinden[].saisonzeiten	array	Alle definierten Saisonzeiten der Gemeinde
gemeinden[].meldeschein_fields	array	Alle definierten Felder des Elektronischen Meldescheins

Meldeschein-Felder

Die definierten Gast/Meldescheinfelder sind unter folgendem Pfad zu finden:

`gemeinden[GEMEINDE_ID].meldeschein_fields`, wobei `GEMEINDE_ID` der jeweiligen Gemeinde entspricht

Relevante Felder:

Feld	Datentyp	Erklärung
alias	string	Klartextbezeichnung
column_name	string	Der Wert der Datenbankspalte, dieser Wert wird bei der Übergabe in `/buchen` benötigt
data_type	string	Datentyp des Felds
required	boolean	Wird der Wert bei der Buchung eines Meldescheins benötigt?

Hinweis

Alle Felder mit `required = true` müssen bei der Buchung eines Meldescheins übergeben werden

Definierte Tarife, Saisonzeiten, Tarifzonen

Die definierten Tarife, Saisonzeiten und Tarifzonen folgen dem gleichen Aufbau

Tarife: `gemeinden[GEMEINDE_ID].tarife`, wobei `GEMEINDE_ID` der jeweiligen Gemeinde entspricht
Saisonzeiten: `gemeinden[GEMEINDE_ID].saisonzeiten`, wobei `GEMEINDE_ID` der jeweiligen Gemeinde entspricht Tarifzonen: `gemeinden[GEMEINDE_ID].tarifzonen`, wobei `GEMEINDE_ID` der jeweiligen Gemeinde entspricht

Beispiele

Folgendes Beispiel ruft die gesamte Konfiguration für einen Demo-Mandanten ab.

Response:

JSON


```
{
  "user": {
    "id": 14,
    "username": "api_demo",
    "first_name": "API",
    "last_name": "Demo",
    "email": "api_demo@apidemo.de",
    "is_active": true,
    "user_type": "beherberger",
    "alias": "API Demo",
    "last_login": "2019-10-24T11:35:18.015947+02:00"
  },
  "beherberger": [
    {
      "id": 9,
      "objekte": [
        {
          "id": 9,
          "tarifzone_id": 1,
          "ortsteil_id": 1,
          "beherberger_id": 9,
          "tarifzone": {
            "id": 1,
            "created": "2019-08-21T15:41:56.239951+02:00",
            "modified": "2019-08-21T15:41:56.240018+02:00",
            "alias": "Hauptzone",
            "gemeinde": 1
          },
          "ortsteil": {
            "id": 1,
            "created": "2019-08-21T15:42:02.596680+02:00",
            "modified": "2019-08-21T15:42:02.596757+02:00",
            "alias": "Fehmarn",
            "gemeinde": 1
          },
          "beherberger": {
            "firma": "DEMO",
            "vorname": "Max",
```

```
          "nachname": "Mustermann",
          "comment": "",
          "debitor_nr": null,
          "alias": "DEMO"
        },
        "created": "2019-09-04T09:51:53.201523+02:00",
        "modified": "2019-09-04T09:51:53.201585+02:00",
        "anzahl_betten": 1,
        "alias": "API-Objekt",
        "strasse": "API-Strasse",
        "plz": "86707",
        "ort": "Westendorf"
      }
    ],
    "alias": "DEMO",
    "created": "2019-09-04T09:49:52.052348+02:00",
    "modified": "2019-09-04T09:56:22.192552+02:00",
    "is_active": true,
    "firma": "DEMO",
    "anrede": "Herr",
    "vorname": "DEMO",
    "nachname": "DEMO",
    "strasse": "DEMO",
    "plz": "123456",
    "ort": "DEMO",
    "telefon": "12345",
    "email": "demo@demo.de",
    "allow_logins": true,
    "create_invoices": true,
    "custom_1": null,
    "custom_2": null,
    "custom_3": null,
    "custom_4": null,
    "debitor_nr": null,
    "vermieter_nr": "API-TEST",
    "comment": "",
    "gemeinde": 1
  }
],
"gemeinden": [
  {
```

```
"id": 1,
"ortsteile": [
  {
    "id": 1,
    "created": "2019-08-21T15:42:02.596680+02:00",
    "modified": "2019-08-21T15:42:02.596757+02:00",
    "alias": "Hauptbereich",
    "gemeinde": 1
  }
],
"tarifzonen": [
  {
    "id": 1,
    "created": "2019-08-21T15:41:56.239951+02:00",
    "modified": "2019-08-21T15:41:56.240018+02:00",
    "alias": "Hauptzone",
    "gemeinde": 1
  }
],
"saisonzeiten": [
  {
    "id": 1,
    "created": "2019-08-21T15:44:42.087596+02:00",
    "modified": "2019-08-21T15:44:42.087663+02:00",
    "alias": "Hauptsaison 2019",
    "valid_from": "2019-05-15",
    "valid_until": "2019-09-14",
    "gemeinde": 1
  },
  {
    "id": 2,
    "created": "2019-08-22T10:11:18.114641+02:00",
    "modified": "2019-08-22T10:11:18.114702+02:00",
    "alias": "Nebensaison 1",
    "valid_from": "2019-01-01",
    "valid_until": "2019-05-14",
    "gemeinde": 1
  },
  {
    "id": 3,
    "created": "2019-08-22T10:12:05.322441+02:00",
```

```
      "modified": "2019-08-22T10:12:05.322534+02:00",
      "alias": "Nebensaison 2",
      "valid_from": "2019-09-15",
      "valid_until": "2019-12-31",
      "gemeinde": 1
    }
  ],
  "tarife": [
    {
      "id": 2,
      "full_alias": "Behinderung 80% (SB 80 %)",
      "created": "2019-08-22T10:16:22.572120+02:00",
      "modified": "2019-08-22T10:16:22.572201+02:00",
      "is_active": true,
      "tarif_code": "SB 80 %",
      "alias": "Behinderung 80%",
      "allow_beherberger": true,
      "print_kurkarte": true,
      "default_selected": false,
      "gemeinde": 1
    },
    {
      "id": 3,
      "full_alias": "Behinderung Begleitung (SBB 80 %)",
      "created": "2019-08-22T10:16:58.052509+02:00",
      "modified": "2019-08-22T10:16:58.052578+02:00",
      "is_active": true,
      "tarif_code": "SBB 80 %",
      "alias": "Behinderung Begleitung",
      "allow_beherberger": true,
      "print_kurkarte": true,
      "default_selected": false,
      "gemeinde": 1
    },
    {
      "id": 7,
      "full_alias": "Beruflich Anwesende (BA)",
      "created": "2019-08-22T10:21:39.062066+02:00",
      "modified": "2019-08-22T10:21:39.062138+02:00",
      "is_active": true,
      "tarif_code": "BA",
```

```
  "alias": "Beruflich Anwesende",
  "allow_beherberger": true,
  "print_kurkarte": true,
  "default_selected": false,
  "gemeinde": 1
},
{
  "id": 1,
  "full_alias": "Erwachsener (EW)",
  "created": "2019-08-21T15:43:26.469087+02:00",
  "modified": "2019-09-05T14:54:45.644304+02:00",
  "is_active": true,
  "tarif_code": "EW",
  "alias": "Erwachsener",
  "allow_beherberger": true,
  "print_kurkarte": true,
  "default_selected": true,
  "gemeinde": 1
},
{
  "id": 5,
  "full_alias": "Gruppen ab 25 Personen (GG)",
  "created": "2019-08-22T10:17:47.179287+02:00",
  "modified": "2019-08-22T10:17:47.179348+02:00",
  "is_active": true,
  "tarif_code": "GG",
  "alias": "Gruppen ab 25 Personen",
  "allow_beherberger": true,
  "print_kurkarte": true,
  "default_selected": false,
  "gemeinde": 1
},
{
  "id": 4,
  "full_alias": "Gruppen unter 25 Personen (GK)",
  "created": "2019-08-22T10:17:30.252887+02:00",
  "modified": "2019-08-22T10:17:30.252952+02:00",
  "is_active": true,
  "tarif_code": "GK",
  "alias": "Gruppen unter 25 Personen",
  "allow_beherberger": true,
```

```
      "print_kurkarte": true,
      "default_selected": false,
      "gemeinde": 1
    },
    {
      "id": 8,
      "full_alias": "keine Kategorie (kK)",
      "created": "2019-08-22T10:22:02.186252+02:00",
      "modified": "2019-08-22T10:22:02.186552+02:00",
      "is_active": true,
      "tarif_code": "kK",
      "alias": "keine Kategorie",
      "allow_beherberger": true,
      "print_kurkarte": true,
      "default_selected": false,
      "gemeinde": 1
    },
    {
      "id": 6,
      "full_alias": "Kinder (Kind)",
      "created": "2019-08-22T10:20:33.649448+02:00",
      "modified": "2019-08-22T10:20:33.649504+02:00",
      "is_active": true,
      "tarif_code": "Kind",
      "alias": "Kinder",
      "allow_beherberger": true,
      "print_kurkarte": true,
      "default_selected": false,
      "gemeinde": 1
    }
  ],
  "meldeschein_fields": [
    {
      "id": 2,
      "created": "2019-08-21T15:46:11.393579+02:00",
      "modified": "2019-09-05T14:56:24.456160+02:00",
      "alias": "Nachname",
      "column_name": "Gast_Name",
      "data_type": "text",
      "size": 6,
      "transferable": false,
```

```
    "list_visible": true,
    "required": true,
    "priority": "high",
    "view_order": 1,
    "gemeinde": 1
  },
  {
    "id": 1,
    "created": "2019-08-21T15:45:57.452169+02:00",
    "modified": "2019-09-05T14:56:20.478747+02:00",
    "alias": "Vorname",
    "column_name": "Gast_Vorname",
    "data_type": "text",
    "size": 6,
    "transferable": false,
    "list_visible": true,
    "required": false,
    "priority": "high",
    "view_order": 2,
    "gemeinde": 1
  },
  {
    "id": 3,
    "created": "2019-08-22T10:26:05.736723+02:00",
    "modified": "2019-08-22T10:26:05.736825+02:00",
    "alias": "Strasse",
    "column_name": "Gast_Strasse",
    "data_type": "text",
    "size": 12,
    "transferable": false,
    "list_visible": false,
    "required": false,
    "priority": "low",
    "view_order": 3,
    "gemeinde": 1
  },
  {
    "id": 4,
    "created": "2019-08-22T10:26:29.499058+02:00",
    "modified": "2019-08-22T10:26:29.499139+02:00",
    "alias": "PLZ",
```

```
      "column_name": "Gast_Postleitzahl",
      "data_type": "text",
      "size": 4,
      "transferable": false,
      "list_visible": false,
      "required": false,
      "priority": "low",
      "view_order": 4,
      "gemeinde": 1
    },
    {
      "id": 5,
      "created": "2019-08-22T10:26:42.577394+02:00",
      "modified": "2019-08-22T10:27:14.479290+02:00",
      "alias": "Ort",
      "column_name": "Gast_Wohnort",
      "data_type": "text",
      "size": 8,
      "transferable": false,
      "list_visible": true,
      "required": false,
      "priority": "low",
      "view_order": 5,
      "gemeinde": 1
    },
    {
      "id": 6,
      "created": "2019-08-26T13:50:23.435905+02:00",
      "modified": "2019-09-05T15:02:12.258999+02:00",
      "alias": "Geburtsdatum",
      "column_name": "Gast_Geburtstag",
      "data_type": "date",
      "size": 4,
      "transferable": false,
      "list_visible": false,
      "required": false,
      "priority": "low",
      "view_order": 6,
      "gemeinde": 1
    }
  ],
```

```
      "created": "2019-08-21T15:00:59.512049+02:00",
      "modified": "2019-08-21T15:36:50.593090+02:00",
      "alias": "Fehmarn",
      "using_paper": true,
      "using_ems": true,
      "datatable_name": "dat_99999"
    }
  ],
  "created": "2019-09-04T09:52:03.825645+02:00",
  "modified": "2019-09-04T09:52:03.825762+02:00",
  "permissions": [
    "authentication.add_vermieteraccount",
    "authentication.change_vermieteraccount",
    "authentication.delete_vermieteraccount",
    "authentication.view_vermieteraccount",
    "common.view_beherberger",
    "common.view_objekt"
  ]
}
```

Meldescheine stornieren

Dieser Aufruf setzt den Status der übergebenen Meldescheine auf "storniert".

> Stornierte Meldescheine sind nicht länger veränderbar.

```
POST  import/meldeschein/stornieren/
```

Parameter

Name	Datentyp	Verwendung
meta	Objekt	Meta-Objekt
meldescheine	Array	Array mit Meldeschein-IDs

Beispiele

Folgendes Beispiel storniert zwei Meldescheine

Request

```
{
  "meta": {
    "gemeinde": 1,
    "objekt": 2049
  },
  "meldescheine": [
    347, 348
  ]
}
```

Response

JSON


```
{
    "meta": {
        "gemeinde": 1,
        "objekt": 2049,
        "requestId": "5546961c-8d5a-11eb-b450-c04a00212a69",
        "timestamp": "2021-03-25T12:07:48.219553",
        "user": {
            "id": 2,
            "username": "admin",
            "alias": "Administrator"
        }
    },
    "status": "success",
    "meldescheine": [
        347,348
    ]
}
```