WinCardCore
Schnittstelle zu externem Abwicklungssystem

Version 3.2

Inhalt

Historie

Version 3.2	28.09.2017	• allgemein positiver Wert für <InfoId>-Elemente • <MediaKey> der Kartennummer identisch mit <CardKey> • 1.5: Aufbau der URLs • Angepasste http-Status-Texte (1.3) • Wertebereich 0..255 für <DeviceNumber>-Elemente
Version 3.1	15.09.2017	• 1.4 Allgemeine Abarbeitung (nachgelagerte Integration der Stammdaten) • 2.5. Body-Aufbau aus 2.6 übernommen • 2.8: Übermittlung aller Kartendaten einzelner Kunden • 2.9. Aktualisierung von Kundendaten (ohne Übermittlung von Kartendaten)
Version 3.0	10.08.2017	• Umstellung von XML auf JSON • Umbenennung DiscountsOnPriceGross in DiscountOnPriceGross • 2.6. Body-Aufbau bei Übermittlung neuer zusätzlicher Kartendaten geändert • Datentypen aktualisiert
Version 2.3	14.07.2017	• Statuswerte der Elemente <DriveIn> und <DriveOut> einer Transaktion erweitert für manuelle Autorisierung an Ein-/Ausfahrt • Authentifizierungsbeispiel korrigiert • Beschränkung des http-Headers auf druckbare ASCII-Zeichen • Element <InfoId> an der Transaktionsschnittstelle • Status-Code 202 an der Kundendatenschnittstelle
Version 2.2	13.06.2017	• Element <EmailActivityActive> in <EmailActivity> umbenannt • Statuswerte der Elemente <DriveIn> und <DriveOut> einer Transaktion erweitert • LEGIC/MIFARE-Medientypen • 1.3 http Status Codes • Status-Code 409 an der Transaktionsschnittstelle
Version 2.1	01.06.2017	• Element <MediaKey> ist optional • Element <AccountKey> in der POST-Methode von cards • Element <CardVariant> für Testkarten • Kommandoschnittstelle • Elemente <DriveOut> und <Price> einer Transaktion nur bei entsprechendem Gesamtstatus • Gesamtstatus 3 (storniert) und Ausfahrtstatus 0 (ausstehend) in der Transaktion entfernt • Transaktionsbeispiel an Beschreibung angepasst • Sperren/Freigeben von Kunden entfernt • Präfix „/ica“ in der remote URL
Version 2.0	24.05.2017	• Erweiterung und Umstellung auf REST-API
Version 1.4	18.10.2016	
Version 1.3	07.10.2016	
Version 1.2	05.10.2016	
Version 1.1	08.09.2016	
Version 1.0	25.05.2016	

1. Allgemeine Schnittstellenbeschreibung

Die Schnittstelle wird über HTTPS-Requests angesprochen und ist an REST angelehnt. Im Header sind nur druckbare ASCII-Zeichen erlaubt. Die Daten im Body liegen im JSON-Format, UTF8 kodiert vor. Datum und Zeit-Werte werden im ISO 8601-Format dargestellt.

1.1. Authentifizierung

Zur Authentifizierung des Senders und Validierung der Daten werden im http-Header die folgenden Parameter übermittelt:

ClientId *Eindeutige Kennzeichnung des API-Aufrufers* z.B. 'parkhaus_betreiber_name'
LocalTime *Aktuelles Datum/Zeit (Großschrift), z.B.'* 2016-02-01T09:00:22'
AuthKey *Schlüssel als Bestandteil der generierten Signatur*
Signature *Generierte Signatur durch* HMAC-SHA256 erstellt

Die Signatur wird durch Verkettung der 3 Zeichenketten (*ClientId, LocalTime, AuthKey*) und Bildung eines HASH-Wertes unter Verwendung eines Signaturschlüssels (SigKey) erstellt. AuthKey und Sigkey müssen jeweils vom Empfänger einmalig bereitgestellt werden. ClientId, AuthKey und SigKey werden zwischen ICA und dem Betreiber des externen Abwicklungssystems vereinbart.

Signature = HMAC-SHA256(*ClientId* + *LocalTime* + *AuthKey*, *SigKey*)

Beispiel:

ClientId: garage_operator_identifier
LocalTime: 2017-01-01T18:00:00
AuthKey: bb86296747fa74384aaaeb129ca549d67c00d5284560a03ae6b9d67528456034
SigKey: 152f20ab789d25e947506ee6485d4cb1959066f970bb65560ad8b4b2112f9b8f

ergibt

Signature: a8b0d807c79f66e54666bb3836e43bee8b472a3dece433bbb02cfc959366e140

1.2. Feldgrößen und Formate

Datentyp	Format	Beispiel
DATETIME	ISO 8601	2016-02-01T09:00:22 = 1. Februar 2016 – 09:00:22 Uhr
AMOUNT	Dezimalzahl mit zwei Nach-kommastellen und einem Punkt (.) als Dezimaltrennzeichen	12.34
VAT	Dezimalzahl mit zwei Nach-kommastellen und einem Punkt (.) als Dezimaltrennzeichen	19.00
CURRENCY	CHAR(3)	‚EUR'
UUID	VARCHAR(200)	‚0800-84747-36441'

CARD_TYPE	SMALLINT	150
CARD_ID	VARCHAR(20)	‚0815‘
MEDIA_TYPE	SMALLINT	1
MEDIA_ID	VARCHAR(40)	Abcdef
TRANSACTION_ID	VARCHAR(50)	‚0006-0000000123‘ = 0006 Parkhaus-Code (4 Stellen) 0000000123 Datensatz-ID (10 Stellen)
TEXT	VARCHAR(255)	‚Text‘
EMAIL	VARCHAR(50)	
TITLE	VARCHAR(40)	
PHONE	VARCHAR(20)	
POSTAL_CODE	VARCHAR(10)	
DISPLAY_NAME	VARCHAR(40)	‚Hans Mustermann‘

1.3. http Status Codes

Zur Kennzeichnung einer erfolgreichen Ausführung werden folgende Status Codes verwendet:

- **200 „OK“**
- **202 „Accepted“**
- **204 „No Content“**

Zur Kennzeichnung einer nicht erfolgreichen Ausführung werden folgende Status Codes verwendet:

- **400 „Bad Request“**
  Aufbau der Daten nicht in Ordnung
- **401 „Unauthorized“**
  Signatur nicht in Ordnung
- **404 „Not Found"**
  Ressource nicht gefunden
- **409 „Conflict“**
  Status der Ressource passt nicht zur Anforderung
- **422 „Unprocessable Entity“**
  Inhalt der Daten ungültig
- **500 „Internal Server Error“**
  Fehler auf Serverseite. Ein erneuter Versuch ist erst nach einer Wartezeit sinnvoll.
- **Sonstige 5xx**
  Fehler auf Serverseite. Ein erneuter Versuch kann direkt erfolgen.

Im Fehlerfall sollte im Body zusätzlich ein lesbarer Text übermittelt werden. Allgemein sollen dem Fehler entsprechende Status-Codes verwendet werden. Sofern nähere Informationen zur Eingrenzung des Fehlers vorhanden sind, sollte ein angepasster Status-Text mit diesen Informationen verwendet werden.

1.4. Allgemeine Abarbeitung

Die Integration der übermittelten Daten in die eigentlichen Stammdaten erfolgt nachgelagert. Daher erfolgt bei der direkten Verarbeitung der Requests keine Kontrolle gegenüber dem aktuellen Bestand der Stammdaten. Sofern ein Request korrekt aufgebaut ist, wird dieser auch akzeptiert. Anforderungen zum Löschen von nicht vorhandenen Elementen führen zu keinem Fehler. Anforderungen zum Aktualisieren von nicht vorhandenen Elementen führen zum automatischen Anlegen der Elemente. Anforderungen zum Anlegen von schon vorhandenen Elementen führen zur Aktualisierung der Elemente.

1.5. Aufbau der URLs

Sowohl das in den URLs des ICA-Systems als auch das in den URLs des externen Abwicklungssystems verwendete Präfix ({PREIFX}) kann unabhängig voneinander konfiguriert werden. Als Standard wird in beiden Fällen „/api/ica“ angenommen.

2. Kundenstammdaten als Whitelist vom externen Abwicklungssystem

Damit nicht bei jeder Einfahrt eines Ausweises eine Online-Anfrage auf Gültigkeit an das externe Abwicklungssystem gestellt werden muss, wird vom externen Abwicklungssystem eine Zulassungsliste (Whitelist) bereitgestellt.

2.1. Übermittlung aller Kundenstammdaten

URL: {PREFIX}/accounts
HTTP-Methode: PUT
HTTP-Status-Code: 204 „No Content“ bzw. 202 „Accepted “

Mittels eines PUT-Request wird eine Komplettliste aller Kunden an das ICA-System gesendet.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Daten der Liste werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

Zur Aufteilung der Übermittlung einer Komplettliste in einzelne Blöcke können optional die Parameter First, Last und Count beim Request übergeben werden.

First: Nummer des ersten Datensatzes im Request
Last: Nummer des letzten Datensatzes im Request
Count: Anzahl aller Datensätze in der Komplettliste

PUT-Requests bei denen Last nicht Count entspricht werden aufgespeichert und erst verarbeitet, wenn ein PUT-Request empfangen wird, bei dem Last und Count den gleichen Wert haben. Ist die Übertragung einer Komplettliste noch nicht abgeschlossen, sind nur weitere PUT-Request zur Vervollständigung der Komplettliste erlaubt. Jegliche sonstigen Requests führen zur Löschung der schon aufgespeicherten PUT-Requests.

Bei Erfolg wird 204 „No Content“ zurück gemeldet, wenn der PUT-Request eine Komplettliste enthält oder es sich um den letzten Block handelt. Handelt es sich um andere Blöcke wird dagegen 202 „Accepted “ gemeldet.

Bsp.-URL: /api/ica/v1/accounts?first=1&last=100&count=1000

Body:

Eine Liste mit einem oder mehreren Kunden.

<Account>
Enthält alle Daten eines Kunden inklusive aller Karten.

Element <AccountKey>
Eine eindeutige ID des Kunden. Wird in der Parktransaktion an die Abrechnungsstelle zurückgegeben.
(Datentyp: UUID)

Element <Customer> (optional)
Angaben zum Kunden.

Element <CustomerNo> (optional)
Kundennummer.
(Datentyp: TEXT)

Element <Gender> (optional)
Geschlecht des Inhabers der Karte, bzw. Medien (männlich, weiblich, unbekannt).
0 = unbekannt
1 = männlich
2 = weiblich
(Datentyp: INTEGER)

Element <Title> (optional)
Anrede des Inhabers der Karte, bzw. Medien (Frau, Herr, Firma, …).
(Datentyp: TEXT)

Element <FirstName> (optional)
Vorname des Inhabers der Karte, bzw. Medien.
(Datentyp: TEXT)

Element <LastName> (optional)
Nachname des Inhabers der Karte, bzw. Medien.
(Datentyp: TEXT)

Element <CompanyName> (optional)
Der Firmenname des Inhabers der Karte.
(Datentyp: TEXT)

Element <DisplayName> (optional)
Der Name des Kartenbesitzers der bei Ein- oder Ausfahrt an der Station angezeigt werden kann. Ist dieses Element nicht vorhanden oder leer, dann werden hierfür die Elemente <LastName> und / oder <FirstName> verwendet, falls sie vorhanden sind.
(Datentyp: DISPLAY_NAME)

Element <DisplayPermission> (optional)
Konfiguration, ob der Besitzername (Inhaber der Karte) an der Ein- und/oder Ausfahrstation angezeigt werden kann.
0 = Systemstandard
1 = Anzeige erlaubt
2 = Anzeige nicht erlaubt
(Datentyp: INTEGER)

Element <PostalCode> (optional)
Postleitzahl zum Wohnort
(Datentyp: POSTAL_CODE)

Element <Location> (optional)
Wohnort
(Datentyp: TEXT)

Element <Street> (optional)
Straße/Hausnummer
(Datentyp: TEXT)

Element <Phone> (optional)
Telefonnummer
(Datentyp: PHONE)

Element <EmailAddress> (optional)
Email-Adresse
(Datentyp: EMAIL)

Element <EmailActivity> (optional)
Benachrichtigung von Ein- und Ausfahrten per Email.
0: Systemeinstellung
1: immer verschicken
2: nie verschicken
(Datentyp: INTEGER)

Element <Info1> .. <Info4> (optional)
Beliebiger Info-Text (Bedeutung je nach Projekt.
(Datentyp: TEXT)

Element <Bool1> .. <Bool4> (optional)
Beliebige TRUE/FALSE-Zustände (Bedeutung je nach Projekt).
(Datentyp: BOOL)

<CardList>
Eine Liste mit einer oder mehreren Karten.

Element <Card>
Enthält alle Daten einer Karte inklusive aller Medien. Es muss zwingend genau ein Media-Element mit dem Medientyp 255 und einer über alle Karten eindeutigen Kartennummer enthalten sein. Card-Elemente, die dies nicht erfüllen, werden nicht verarbeitet. Das MediaKey-Element der Kartennumer entspricht funktional dem CardKey-Element und wird daher automatisch durch den CardKey-Wert überschrieben.

Element <CardKey>
Die eindeutige ID der Karte.
(Datentyp: UUID)

Element <CardVariant> (optional)
Information über die Kartenvariante:
0: Standardkarte
1: Testkarte

Ist das Element nicht vorhanden, wird Variante ‚0' vorgegeben.

Parkgebühren von Testkarten laufen im ICA-System normal auf. Im Transaktionenreport werden die Varianten aber getrennt ausgewiesen. Im externen Abwicklungssystem sollten Testkarten nicht in Rechnung gestellt werden.
(Datentyp: INTEGER)

Element <Blocked>
Information über Sperrzustand der Karte
0: nicht gesperrt (darf einfahren)
1: gesperrt (Standard)
2: gesperrt, Grund 2
3: gesperrt, Grund 3
4: …
(Datentyp: INTEGER)

Element <BlockedInfo>
Detailinformationen zum Sperrzustand der Karte, z.B. warum die Karte gesperrt wurde.
(Datentyp: TEXT)

Element <Category> (optional)
Über die Kategorie können zu verwendender Tarif, Zeitprofil, Beleggruppe, Kunde und Parkbereiche bestimmt werden. Ist das Element nicht vorhanden, wird Kategorie ‚0' vorgegeben.
(Datentyp: INTEGER)

Element <CarParkId> (optional)
Der von ICA vergebene Parkhaus-Code des Parkhauses, für das die Karte ausgegeben wurde.
(Datentyp: INTEGER)

Element <CardBrand> (optional)
Markenname der Karte bzw. Medien des Inhabers, z.B. ‚Müller-Flottenkarte, Meier-Kundenkarte'
(Datentyp: TEXT)

Element <ValidFrom> (optional)
Die Karte hat ab diesem Datum Gültigkeit. Ist das Element nicht vorhanden, so ist das Medium ab sofort gültig.
(Datentyp: DATETIME)

Element <ValidTo> (optional)
Die Karte hat bis zu diesem Datum Gültigkeit. Ist das Element nicht vorhanden, so ist das Medium unbegrenzt gültig.
(Datentyp: DATETIME)

<MediaList>
Eine Liste von Media-Elementen, die für die Karte zugelassen sind.

Element <Media>
Element unterhalb des Elementes <MediaList>, das ein zugelassenes Medium beschreibt.

Element <MediaType>
Verwendeter Medientyp
1: UHF-Tag
2: Kfz-Kennzeichen
21: MIFARE Classic
22: MIFARE Ultralight
23: MIFARE DESFire
31: LEGIC prime
32: LEGIC advant
255: Kartennummer
(Datentyp: MEDIA_TYPE)

Element <MediaId>
Je nach Medientyp eine eindeutige Zeichenkette. Groß-/Kleinschreibung wird NICHT berücksichtigt (eventuelle Ausnahmen werden unter Element <MediaType> gesondert aufgelistet).
(Datentyp: MEDIA_ID)

Element <MediaKey> (optional)
Eindeutige Bezeichner der Abrechungsstelle für dieses Medium. Wird falls vorhanden in Transaction zurückgeliefert.
(Datentyp: UUID)

Beispiel:


```
[{
    "AccountKey": "ce87c87a87d28b33",
    "Customer": {
        "CustomerNo": "69",
        "Gender": 1,
        "Title": "Herr",
        "FirstName": "Hans",
        "LastName": "Mustermann",
        "CompanyName": "Car Company",
        "DisplayName": "Mustermann",
        "DisplayPermission": 0,
        "PostalCode": "46000",
        "Location": "Dortmund",
        "Street": "Am Wendehammer",
        "Phone": "0231/123456",
        "EmailAddress": "mustermann@google.com",
        "EmailActivity": 0,
        "Info1": "Text",
        "Bool1": false,
        "Bool2": false,
        "Bool3": false,
        "Bool4": false
    },
    "Card": [{
        "CardKey": "ce87c87a87d28b34",
        "CardVariant": 0,
        "Blocked": 1,
        "BlockedInfo": "Failed to pay the last invoice",
        "Category": 3,
        "CarParkId": 3,
        "CardBrand": "Car Sharing",
        "ValidFrom": "2016-02-01T09:00:22",
        "ValidTo": "2016-03-30T23:59:59",
        "Media": [{
            "MediaType": 255,
            "MediaId": "Kartennummer",
            "MediaKey": "12345"
        },
        {
            "MediaType": 1,
            "MediaId": "UHF-UID-2",
            "MediaKey": "0800-84747-36441"
        },
        {
            "MediaType": 1,
            "MediaId": "UHF-UID-3",
            "MediaKey": "0800-84747-36442"
        }]
    }]
}]
```

2.2. Aktualisierung bestehender Kundenstammdaten

URL: {PREFIX}/v1/accounts
HTTP-Methode: PATCH
HTTP-Status-Code: 204 „No Content“

Mittels eines PATCH-Request werden vorhandenen Kundenstammdaten modifiziert
Der Body entspricht der PUT-Methode. Dementsprechend werden vorhandene Karten eines Kunden gelöscht, wenn sie im Request nicht übermittelt werden.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Aktualisierungen werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

2.3. Übermittlung neuer zusätzlicher Kundenstammdaten

URL: {PREFIX}/v1/accounts
HTTP-Methode: POST
HTTP-Status-Code: 204 „No Content“

Mittels eines POST-Request werden neue zusätzliche Kundenstammdaten übermittelt.
Der Body entspricht der PUT-Methode.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Daten der Liste werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

2.4. Löschen von Kundenstammdaten

URL: {PREFIX}/v1/accounts
HTTP-Methode: DELETE
HTTP-Status-Code: 204 „No Content“

Mittels eines DELETE-Request werden Kundenstammdaten gelöscht.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Der Request wird nur ausgeführt, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

Body:

Eine Liste mit einer oder mehreren Kunden-IDs.

<AccountKey>
Eine eindeutige ID des Kunden.
(Datentyp: UUID)

Beispiel:

```
[
  "ce87c87a87d28b33",
  "ce87c87a87d28b34",
  "ce87c87a87d28b35"
]
```

2.5. Aktualisierung bestehender Kartendaten

URL: {PREFIX}/v1/cards
HTTP-Methode: PATCH
HTTP-Status-Code: 204 „No Content“

Mittels eines PATCH-Request werden vorhandene Kartendaten modifiziert. Nur übermittelte Karten werden verändert. Es können somit einzelne Karten eines Kunden modifiziert werden, ohne dass alle Karten des Kunden übermittelt werden müssen. Die Kundenzuordnung ist ebenfalls Bestandteil der Kartendaten. D.h. ist eine Karte schon vorhanden, aber einem anderen Kunden zugeordnet, wird sie automatisch dem neuen Kunden zugeordnet. Der Body entspricht der PUT-Methode der Kundenstammdaten ohne das <Customer>-Element.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Aktualisierungen werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

Body:

Eine Liste von Kartenlisten je Kunde.

<AccountCards>
Enthält eine Kartenliste eines Kunden

Element <AccountKey>
Die eindeutige ID des Kunden.
(Datentyp: UUID)

<CardList>
Eine Liste mit einer oder mehreren Karten.
Entspricht im Aufbau <CardList> der Kundendaten.

Beispiel:

```
[{
    "AccountKey": "ce87c87a87d28b33",
    "Card": [{
        "CardKey": "ce87c87a87d28b34",
        "CardVariant": 0,
        "Blocked": 1,
        "BlockedInfo": "Failed to pay the last invoice",
        "Category": 3,
        "CarParkId": 3,
        "CardBrand": "Car Sharing",
        "ValidFrom": "2016-02-01T09:00:22",
        "ValidTo": "2016-03-30T23:59:59",
        "Media": [{
            "MediaType": 255,
            "MediaId": "Kartennummer",
            "MediaKey": "12345"
        },
        {
            "MediaType": 1,
            "MediaId": "UHF-UID-2",
            "MediaKey": "0800-84747-36441"
        },
        {
            "MediaType": 1,
            "MediaId": "UHF-UID-3",
            "MediaKey": "0800-84747-36442"
        }]
    }]
}]
```

2.6. Übermittlung neuer zusätzlicher Kartendaten

URL: {PREFIX}/v1/cards
HTTP-Methode: POST
HTTP-Status-Code: 204 „No Content“

Mittels eines POST-Request werden neue zusätzliche Kartendaten für bereits bestehende Kunden übermittelt. Der Body entspricht der PATCH-Methode.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Daten der Liste werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

2.7. Löschen von Kartendaten

URL: {PREFIX}/v1/cards
HTTP-Methode: DELETE
HTTP-Status-Code: 204 „No Content“

Mittels eines DELETE-Request werden Kartendaten gelöscht.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Der Request wird nur ausgeführt, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.
Body:

Eine Liste mit einer oder mehreren Karten-IDs.

<CardKey>
Die eindeutige ID der Karte.
(Datentyp: UUID)

Beispiel:
```
[
  "ce87c87a87d28b37",
  "ce87c87a87d28b38",
  "ce87c87a87d28b39"
]
```

2.8. Übermittlung aller Kartendaten einzelner Kunden

URL: {PREFIX}/v1/cards
HTTP-Methode: PUT
HTTP-Status-Code: 204 „No Content“

Mittels eines PUT-Request werden jeweils alle Kartendaten für bereits bestehende Kunden übermittelt. Vorhandene Karten eines übermittelten Kunden, die nicht im Request enthalten sind, werden gelöscht. Der Body entspricht der PATCH-Methode.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Daten der Liste werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

2.9. Aktualisierung bestehender Kundendaten

URL: {PREFIX}/v1/customers
HTTP-Methode: PATCH
HTTP-Status-Code: 204 „No Content“

Mittels eines PATCH-Request werden vorhandene Kundendaten modifiziert. Der Body entspricht der PUT-Methode der Kundenstammdaten ohne eine Liste der <Card>-Elemente.

Es wird vom ICA-System kein Antwort-Telegramm gesendet, sondern nur ein http-Statuscode. Die Daten der Liste werden nur dann in das ICA-System übernommen, wenn die gesamte Liste fehlerfrei verarbeitet werden konnte, andernfalls bleibt der alte Datenbestand erhalten.

Body:

Eine Liste von Daten je Kunde.

<AccountCustomer>
Enthält die Daten eines Kunden

Element <AccountKey>
Die eindeutige ID des Kunden.
(Datentyp: UUID)

Element <Customer>
Angaben zum Kunden.
Entspricht im Aufbau dem <Customer>-Element der Kundendaten.

Beispiel:

```
[{
    "AccountKey": "ce87c87a87d28b33",
    "Customer": {
        "CustomerNo": "69",
        "Gender": 1,
        "Title": "Herr",
        "FirstName": "Hans",
        "LastName": "Mustermann",
        "CompanyName": "Car Company",
        "DisplayName": "Mustermann",
        "DisplayPermission": 0,
        "PostalCode": "46000",
        "Location": "Dortmund",
        "Street": "Am Wendehammer",
        "Phone": "0231/123456",
        "EmailAddress": "mustermann@google.com",
        "EmailActivity": 0,
        "Info1": "Text",
        "Bool1": false,
        "Bool2": false,
        "Bool3": false,
        "Bool4": false
    }
}]
```

3. Abfrage von Kundenstammdaten am externen Abwicklungssystem

Es ist möglich am externen Abwicklungssystem Kundenstammdaten abzufragen. So ist es z.B. möglich, dass beim Verwenden eines in der Whitelist als gesperrt hinterlegten Ausweismedium, der aktuelle Sperrzustand ‚Online' am externen Abwicklungssystem angefragt werden kann.

3.1. Abfrage aller Kundenstammdaten vom externen Abwicklungssystem

URL (remote): {PREFIX}/v1/accounts
HTTP-Methode: GET
HTTP-Status-Code: 200 „OK"

Mittels eines GET-Request wird eine Komplettliste aller Kunden abgefragt.

Als Antwort wird vom externen Abwicklungssystem eine Komplettliste (<AccountList>) im Aufbau entsprechend dem PUT-Request für Kundenstammdaten übermittelt.

Im Fehlerfall wird ein http-Fehlercode vom externen Abwicklungssystem übermittelt.

3.2. Abfrage der Stammdaten eines Kunden vom externen Abwicklungssystem

URL (remote): {PREFIX}/v1/accounts/{AccountKey}
HTTP-Methode: GET
HTTP-Status-Code: 200 „OK"

Mittels eines GET-Request mit angegebener Kunden-ID werden die Stammdaten eines Kunden abgefragt.

Als Antwort wird vom externen Abwicklungssystem ein Kundendatensatz (<Account>) eines einzelnen Kunden im Aufbau entsprechend dem PUT-Request für Kundenstammdaten übermittelt.

Im Fehlerfall wird ein http-Fehlercode vom externen Abwicklungssystem übermittelt.

3.3. Abfrage der Daten einer Karte vom externen Abwicklungssystem

URL (remote): {PREFIX}/v1/cards/{CardKey}
HTTP-Methode: GET
HTTP-Status-Code: 200 „OK"

Mittels eines GET-Request mit angegebener Karten-ID werden die Daten einer Karte abgefragt.

Als Antwort wird vom externen Abwicklungssystem ein Kartendatensatz (<Card>) einer einzelnen Karte im Aufbau entsprechend dem PUT-Request für Kundenstammdaten übermittelt.

Im Fehlerfall wird ein http-Fehlercode vom externen Abwicklungssystem übermittelt.

4. Senden von Ein-/Ausfahrt-Transaktionen an das externe Abwicklungssystem

Die Übermittlung von Transaktionen an das externe Abwicklungssystem erfolgt über PUT/PATCH-Requests. Die zu übertragenden JSON-Daten enthalten abhängig davon, ob es sich um eine Ein- oder Ausfahrt handelt, das Element <DriveOut> oder nur <DriveIn>. Bei einer Einfahrt entfällt außerdem <Price>.

4.1. Übermittlung einer neuen Transaktion

URL (remote): {PREFIX}/v1/transactions/{TransactionId}
HTTP-Methode: PUT
HTTP-Status-Code: 204 „No Content“

Mittels eines PUT-Request wird eine neue Transaktion an das externe Abwicklungssystem gesendet. {TransactionID} ist die 15 stellige Transaktions-ID des Parksystems und besteht aus dem Parkhaus-Code und der Datensatz-ID der Transaktion.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Sollte es die Transaktion schon mit identischen Daten im externen Abwicklungssystem geben, weil z.B. die Antwort des externen Abwicklungssystems bei einem vorherigen erfolgreichen Einlieferungsversuch auf Grund eines Verbindungsabbruchs nicht empfangen wurde, so muss ebenfalls der Status-Code 204 „No Content“ übermittelt werden. Sollten die Daten abweichen, muss stattdessen der Status-Code 409 „Conflict“ übermittelt werden.

Body:

<Transaction>
Transaction enthält alle Daten eines kompletten Parkvorgangs.

Element <CarParkOperatorName>
Der Betreibername des Parksystems.
(Datentyp: TEXT)

Element <CarParkId>
Der von ICA vergebene Parkhaus-Code des Parkhauses, in dem der Parkvorgang erfolgte.
(Datentyp: INTEGER)

Element <CarParkName>
Name des Parkhauses
(Datentyp: TEXT)

Element <AccountKey>
Die eindeutige ID des Kunden.
(Datentyp: UUID)

Element <Media>
Enthält die Kartennummer als Medium.

Element <MediaType>
Verwendeter Medientyp
255: Kartentyp
(Datentyp: MEDIA_TYPE)

Element <MediaId>
Die Kartennummer.
(Datentyp: MEDIA_ID)

Element <MediaKey> (entspricht <CardKey>)
Eindeutige ID dieser Karte.
(Datentyp: UUID)

Element <Status>
Status des gesamten Parkvorgangs.
0: Offen (das Ausweismedium befindet sich im Parkhaus)
1: Beendet
2: Abgebrochen
(Datentyp: INTEGER)

Element <DriveIn>
Enthält alle Elemente zur Einfahrt eines Ausweismediums.

Element <DeviceNumber>
Die ID der Station die bei der Einfahrt benutzt wurde.
0..255
(Datentyp: INTEGER)

Element <DateTime>
Zeitpunkt der Einfahrt.
(Datentyp: DATETIME)

Element <Media>
Enthält die Elemente die das benutzte Ausweismedium beschreiben.

Element <MediaType>
Verwendeter Medientyp
1: UHF-Tag
2: Kfz-Kennzeichen
21: MIFARE Classic
22: MIFARE Ultralight
23: MIFARE DESFire
31: LEGIC prime
32: LEGIC advant
(Datentyp: MEDIA_TYPE)

Element <MediaId>
Je nach Medientyp eine eindeutige Nummer.
(Datentyp: MEDIA_ID)

Element <MediaKey> (sofern bekannt)
Eindeutige ID dieses Mediums im externen Abwicklungssystem.
(Datentyp: UUID)

Element <Status>
Status der Einfahrt
1: Gestartet durch das System
2: Gestartet durch den Bediener
3: Gestartet durch den Bediener als Ersatz (neue Einfahrtzeit)
4: Gestartet durch das System mit Autorisierung durch den Bediener (an der Einfahrt)
(Datentyp: INTEGER)

Element <InfoId> (optional)
Zusätzliche Information/Begründung für den Vorgang
1: manuelle Autorisierung, da Karte an Station nicht lesbar
2: ….
Hier sollen allgemein positive Werte unterstützt werden, da sich im Praxisbetrieb verschiedene Standard-Informationen/Begründungen ergeben werden.
(Datentyp: INTEGER)

Element <DriveOut> (nur bei Gesamtstatus 1=beendet oder 2=abgebrochen)
Enthält alle Elemente zur Ausfahrt eines Ausweismediums (entspricht Element <DriveIn>).

Element <Status>
Status der Ausfahrt
1: Beendet durch das System (normale Parkpreisberechnung)
2: Beendet durch den Bediener (normale Parkpreisberechnung)
3: Beendet durch das System mit Preisvorgabe, da max. Parkzeit überschritten
4: Beendet durch den Bediener mit Preisvorgabe
5: Beendet durch das System mit Autorisierung durch den Bediener
(an Ausfahrt, normale Parkpreisberechnung)
6: Beendet durch das System mit Autorisierung und Preisvorgabe durch den Bediener
(an Ausfahrt)
101: Abgebrochen durch das System (das Ausweismedium ist ohne Ausfahrt erneut eingefahren)
102: Abgebrochen durch den Bediener
103: Abgebrochen durch Rollback (Fahrzeug ist nicht eingefahren)
104: Abgebrochen durch das System, da max. Parkzeit überschritten
105: Abgebrochen durch den Bediener für eine Neuanlage (neue Einfahrzeit)
(Datentyp: INTEGER)

Element <InfoId> (optional)
Zusätzliche Information/Begründung für den Vorgang
1: manuelle Autorisierung, da Karte an Station nicht lesbar
2: ….
Hier sollen allgemein positive Werte unterstützt werden, da sich im Praxisbetrieb verschiedene Standard-Informationen/Begründungen ergeben werden.
(Datentyp: INTEGER)

Element <Price> (nur bei Gesamtstatus 1=beendet)
Enthält alle Elemente zur Beschreibung des gezahlten Betrags.

Element <PriceGross>
Der Preis für den Parkvorgang. Dieses Feld ist nur beim Ausfahrt-Request gefüllt. Die Rabattierungen sind bereits abgezogen.
(Datentyp: AMOUNT)

Element <VatPercentage>
Der Prozentsatz der Mehrwertsteuer die in <PriceGross> enthalten ist.
(Datentyp: VAT)

<DiscountOnPriceGrossList>
Eine Liste von Rabattierungen die aus <DiscountOnPriceGross> Elementen besteht.

Element <DiscountOnPriceGross>
Enthält alle Elemente die eine Rabattierung beschreiben.

Element <Amount>
Betrag des gewährten Rabatts.
(Datentyp: AMOUNT)

Element <Source>
Anlass oder Verwendung des gewährten Rabatts.
(Datentyp: TEXT)

Element <LocationID>
Kennung wodurch der Rabatt gewährt wurde (Kostenstelle und Seriennummer des Rabattierers)
(Datentyp: TEXT)

Element <LocationName>
Kennung wo der Rabatt gewährt wurde (z.B. Name des Kunden dem der Rabattierer gehört).
(Datentyp: TEXT)

Element <Comment>
Beliebiger Kommentar.
(Datentyp: TEXT)

Element <Currency>
Die verwendete Währung, z.Z. immer ‚EUR‘.
(Datentyp: CURRENCY)

Beispiel:


```
{
    "CarParkOperatorName": "Schmidt",
    "CarParkId": 123,
    "CarParkName": "Schmidt-Parkhaus-1",
    "AccountKey": "ce87c87a87d28b33",
    "Media": {
        "MediaType": 255,
        "MediaId": "0815",
        "MediaKey": "0800-84747-36441"
    },
    "Status": 1,
    "DriveIn": {
        "DeviceNumber": 1,
        "DateTime": "2016-02-01T09:00",
        "Media": {
            "MediaType": 1,
            "MediaId": "UHF-UID-2",
            "MediaKey": "0800-84747-36442"
        },
        "Status": 1,
        "InfoId": 1
    },
    "DriveOut": {
        "DeviceNumber": 51,
        "DateTime": "2016-02-01T11:00",
        "Media": {
            "MediaType": 1,
            "MediaId": "UHF-UID-2",
            "MediaKey": "0800-84747-36442"
        },
        "Status": 1,
        "InfoId": 1
    },
    "Price": {
        "PriceGross": 0.50,
        "VatPercentage": 19.0,
        "DiscountOnPriceGross": [{
            "Amount": 1.50,
            "Source": "Anlass oder Verwendung",
            "LocationID": "5:5",
            "LocationName": "Supermarkt Müller",
            "Comment": "Ein Kommentar"
        }],
        "Currency": "EUR"
    }
}
```

4.2. Aktualisierung einer schon übermittelten Transaktion

URL (remote): {PREFIX}/v1/transactions/{TransactionId}
HTTP-Methode: PATCH
HTTP-Status-Code: 204 „No Content“

Mittels eines PATCH-Request kann eine schon übermittelte Transaktion im externen Abwicklungssystem aktualisiert werden. {TransactionId} ist die im Parksystem vergebene 15 stellige Transaktions-ID.

Hierüber werden Ausfahrtdaten einer schon vorhandenen Transaktion mit ausschließlich Einfahrtdaten hinzugefügt.

Der Aufbau des Bodys entspricht dem aus dem PUT-Request.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Sollte es die Transaktion schon mit identischen Daten im externen Abwicklungssystem geben, weil z.B. die Antwort des externen Abwicklungssystems bei einem vorherigen erfolgreichen Einlieferungsversuch auf Grund eines Verbindungsabbruchs nicht empfangen wurde, so muss ebenfalls der Status-Code 204 „No Content“ übermittelt werden. Sollten die Daten abweichen, muss stattdessen der Status-Code 409 „Conflict“ übermittelt werden.

4.3. Stornierung einer schon übermittelten Transaktion

URL (remote): {PREFIX}/v1/transactions/{TransactionId}
HTTP-Methode: DELETE
HTTP-Status-Code: 204 „No Content“ (stornierbar) oder 409 „Conflict“ (nicht stornierbar)

Mittels eines DELETE-Request kann eine schon übermittelte Transaktion im externen Abwicklungssystem storniert werden. {TransactionId} ist die im Parksystem vergebene 15 stellige Transaktions-ID.

Body:

Element <Amount> optional
Mit diesem Element kann ein Teilbetrag der Transaktion erstattet werden. Ist das Element nicht vorhanden, wird der gesamte Betrag der Transaktion erstattet.
(Datentyp: AMOUNT)

Element <InfoId> (optional)
Zusätzliche Information/Begründung für den Vorgang
1: …
2: …
Hier sollen allgemein positive Werte unterstützt werden, da sich im Praxisbetrieb verschiedene Standard Informationen/Begründungen ergeben werden.
(Datentyp: INTEGER)

Element <Info> optional
Ein erklärender Text, warum die Rückerstattung erfolgte.
(Datentyp: TEXT)

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Sollte die Transaktion nicht mehr storniert werden können, weil sie schon abgerechnet wurde, so muss der Status-Code 409 „Conflict“ übermittelt werden.

Die positive/negative Antwort des externen Abwicklungssystems wird vom ICA-System als „endgültig“ bewertet, auch wenn der eigentliche Stornierungsvorgang im externen Abwicklungssystem erst später verarbeitet wird.

Beispiel:

```
{
    "Amount": 1.5,
    "Info": "Error",
    "InfoId": 5
}
```

5. Sperren/Freigeben von Karten im externen Abwicklungssystem durch Bediener

Karten sollen im externen Abwicklungssystem durch einen Bediener freigegeben oder gesperrt werden können.

5.1. Sperren einer Karte

URL (remote): {PREFIX}/v1/lock/cards/{CardKey}
HTTP-Methode: POST
HTTP-Status-Code: 204 „No Content“

Mittels eines POST-Request wird die über CardKey referenzierte Karte gesperrt.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Body:

Element <Info> optional
Ein erklärender Text, warum die Sperrung erfolgte.
(Datentyp: TEXT)

Beispiel:
```
{
    "Info": "Card lost"
}
```

5.2. Freigabe einer Karte

URL (remote): {PREFIX}/v1/lock/cards/{CardKey}
HTTP-Methode: DELETE
HTTP-Status-Code: 204 „No Content“

Mittels eines DELETE-Request wird die Sperrung der über CardKey referenzierten Karte aufgehoben.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Body:

Element <Info> optional
Ein erklärender Text, warum die Freigabe erfolgte.
(Datentyp: TEXT)

Beispiel:
```
{
    "Info": "Card regained"
}
```

6. Aktionen im externen Abwicklungssystem

Verschiedene Aktionen können im externen Abwicklungssystem ausgelöst werden.

6.1. Kommandos zum externen System

URL (remote): {PREFIX}/v1/command
HTTP-Methode: POST
HTTP-Status-Code: 204 „No Content“

Mittels eines POST-Request wird eine Aktion im externen System ausgelöst.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.

Body:

Element <Command>
Die gewünschte Aktion:
1: hiermit wird eine erneute Übermittlung aller Kundendaten über einen PUT-Request angefordert.
…
(Datentyp: INTEGER)

Beispiel:
```
{
    "Command": 1
}
```

6.2. Alive-Info zum externen System

URL (remote): {PREFIX}/v1/command
HTTP-Methode: GET
HTTP-Status-Code: 200 „OK“

Mittels eines GET-Request ermittelt das ICA-System, ob das externe Abwicklungssystem noch erreichbar ist und gleichzeitig erfährt dieses, dass das ICA-System betriebsbereit ist.

Vom externen Abwicklungssystem wird ein http-Antwortcode übermittelt.