VeriFone Way2i^{TM} API Reference Guide, ver. 1.17

August 2013

This is a pre-release documentation of the VeriFone Way2i API.

All methods, interfaces, and protocols are subject to change.

Overview

The VeriFone Way2i services enable complex interaction with the VeriFone taxi ecosystem. This interaction may be from within a mobile application or from a service provider owned enterprise service/server. The services are developed for secure communication as it pertains to session and payment services.

Way2i services are split into several broad categories, including.

- Wallet services
- Payment services
- Taxi services

All services are provided in the form of RESTful Web services that use Google Protocol Buffers.

Glossary

The following terminology will be used throughout this document:

Way2i	Way2i will be used to describe the overall VeriFone API architecture as well as individual servers and services that implement it. Way2i, Way2 interface, and Way2i services will be used interchangeably.
Payment Instrument (Card)	A payment instrument is a credit card, a checking account, or any other financial instrument that can be registered by Way2i server. Payment Instrument, Payment Card or Card will be used interchangeably throughout this document.
PAN	Personal Account Number – an account number formatted according to the ISO/IEC 7812 standard.
Account	A subscriber's payment instrument, alternatively called a "card" or a "payment card", even if the specific payment instrument does not have a physical plastic card associated with it. Examples of accounts include credit cards, debit cards, checking accounts, and PayPal accounts.
Card Token	Card token is an unsecure identifier of a payment card that can be stored by the client application. Card tokens can be used by Way2i server to uniquely identify a stored card. Way2i services will then have access to the PAN or other identifying information.
Service Provider (SP)	Consumer/customer of the VeriFone Way2i API, typically a company that provides a set of applications or services (mobile or Web-based) to its subscribers.
Subscriber	An individual that has an account with the Service Provider that signed up for using Way2i interface. A subscriber is typically an individual who uses a mobile app or a Web application to access the services provided by the Service Provider.

VeriFone Way2i API

The VeriFone Way2i API is based on HTTP RESTful services that expose all Way2i services for consumption by server applications and mobile apps of Service Providers.

Way2i services are organized in several logical groups, with layers of abstraction and access based on subscription levels. The core services are represented on the following diagram:

Access to the Way2i APIs requires signing a Developer Agreement and a Service Contract between VeriFone and the Service Provider.

Your Developer Agreement and Service Contract will determine which API methods from documented in this Guide will be available for a your organization. Not all methods specified may be available for all Service Providers.

To sign up for VeriFone Way2i developer agreement, send an email inquiry to *way2i-developer@verifoneway.com* or download the sign up form from *http://verifoneway.com/way2i-developer*

Accessing Way2i Services

Upon entering into the developer agreement and a service contract with VeriFone, a representative from the VeriFone Way2i developer support team will contact you with the specific details accessing the services. To get started you will need to create an SSL certificate and submit your Certificate Signing Request to VeriFone.

A sandbox for integration testing will be provided, and access to the production Way2i services will be granted to Service Provider upon successful completion of the certification of its application.

As soon as your developer account is active, you will receive the welcome email from VeriFone that will include details about your sandbox environment. To access it, you will need the following:

- The URI of the service endpoint for your sandbox environment.
- Connection details. The endpoint may be accessible either through the public Internet or through a VPN link provided by VeriFone, which is determined in each individual case
- Your SSL certificate signed by VeriFone.
- Some additional information including the phone number and the client ID for accessing developer support may be also provided during service onboarding.

Way2i services are architected with consideration for security from ground up as vast majority of these services are within the scope of PCI DSS.

Latency and security are balanced in how the calls are made and the data is serialized. Way2i uses open source Google Protocol Buffers for efficient data serialization.

Architecture of the Way2i interface

The Way2i interface assumes that it will be accessed by a multi-tier application that includes the client and the server component.

The client component is a mobile app that provides some utility functionality and services to subscribers. The server application supports the functionality of the mobile app.

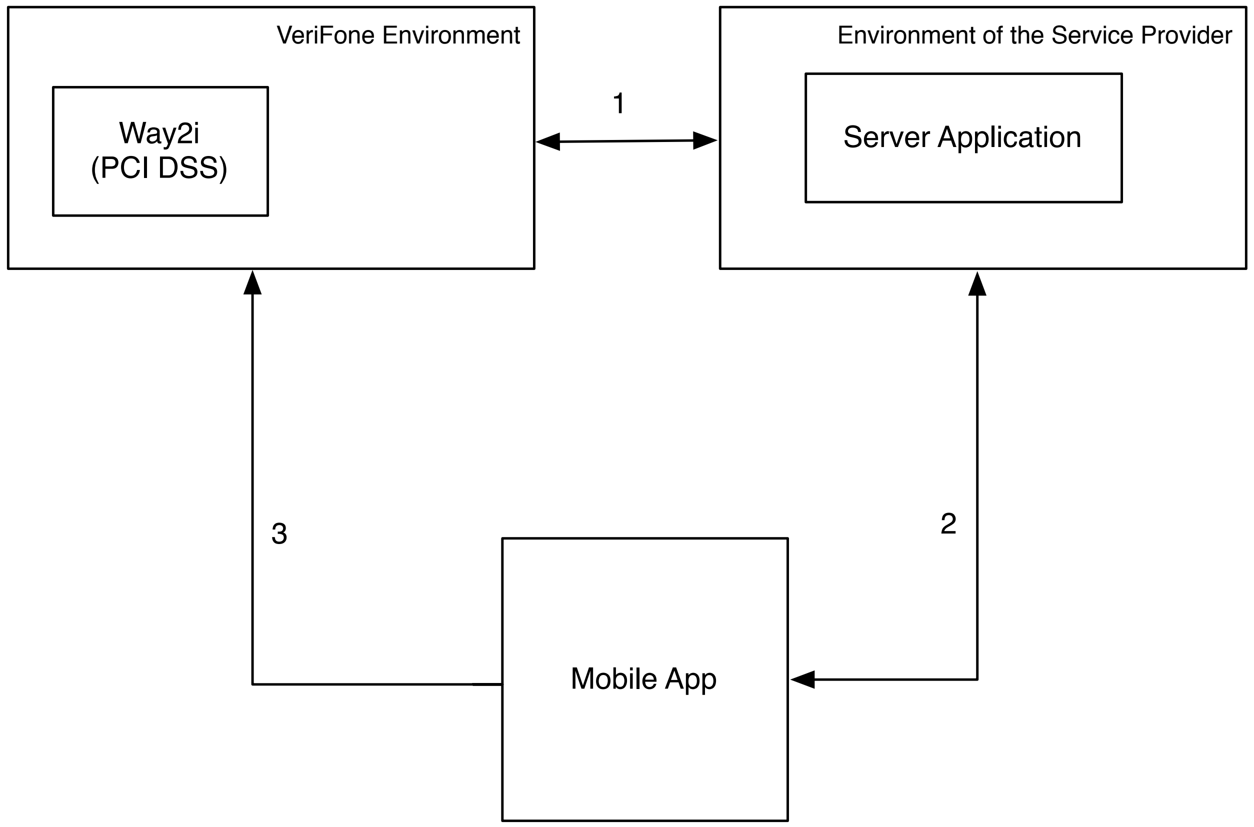

FIG 1. VeriFone Way2i Architecture Diagram

The Way2i Architecture Diagram illustrates the following components:

- The VeriFone environment in which the server infrastructure for Way2i services resides. The infrastructure includes two components
    - Way2i component within PCI scope (for operations with credit cards)
    - Way2i general service components (for all other operations)
- The Mobile App and the Server Applications are typically provided by the same company, client of VeriFone Way2i services. The Mobile App and the Server Applications are tightly integrated and we make the following assumptions:
    - The protocol between the Mobile App and the Server Application is efficient and secure
    - The Server Application can authenticate the Mobile App users and authorize them to access specific services exposed by the Server Application

- The ***secure interface 1*** is established for Server Application to access the VeriFone Way2i services. The interface is protected by the certificate pair, provisioned during the certificate exchange and license setup. While the VeriFone Way2i services are typically accessible over a public Internet, only authorized clients with valid certificates can access these services through the secure interface. VeriFone Way2i services are also available over a VPN channel.

- Requests that are received by VeriFone Way2i services over the ***secure interface 1*** are assumed to be *trusted* for Mobile App users. That means that the Mobile App users do not have to be authenticated by the Way2i server as it is expected that these users will be authenticated by the Service Provider's Server Application.

- Requests that are received by VeriFone Way2i services over ***interface 3*** are untrusted. They are not executed by the Way2i server unless they are confirmed by the Customer's server.

In such a model, we assume that the majority of the Way2i API functions will be accessed by the Server Application over the trusted interface 1, while some certain functions, for example card enrollment, will be accessed directly from the mobile app (for example card enrollment.)

Once the Service Provider is onboarded, interaction between the SP and the Way2i server takes place as follows:

- The Service Provider's server sends requests by using RESTful requests with provided URI over a trusted interface secured by the customer's certificate provided by VeriFone. The Way2i server knows the identifying information of the customer based on their certificate.

- When the mobile client needs to access the Way2i server (for example for credit card enrollment), the mobile client sends the RESTful service URI directly over an untrusted interface and supplies the customer token provided by VeriFone. The Way2i server receives the requests and adds it to a queue until it is confirmed by the customer's server over a trusted interface. Data in the queue is expired and discarded after certain time frame.

Architecture of the Way2i-enabled application

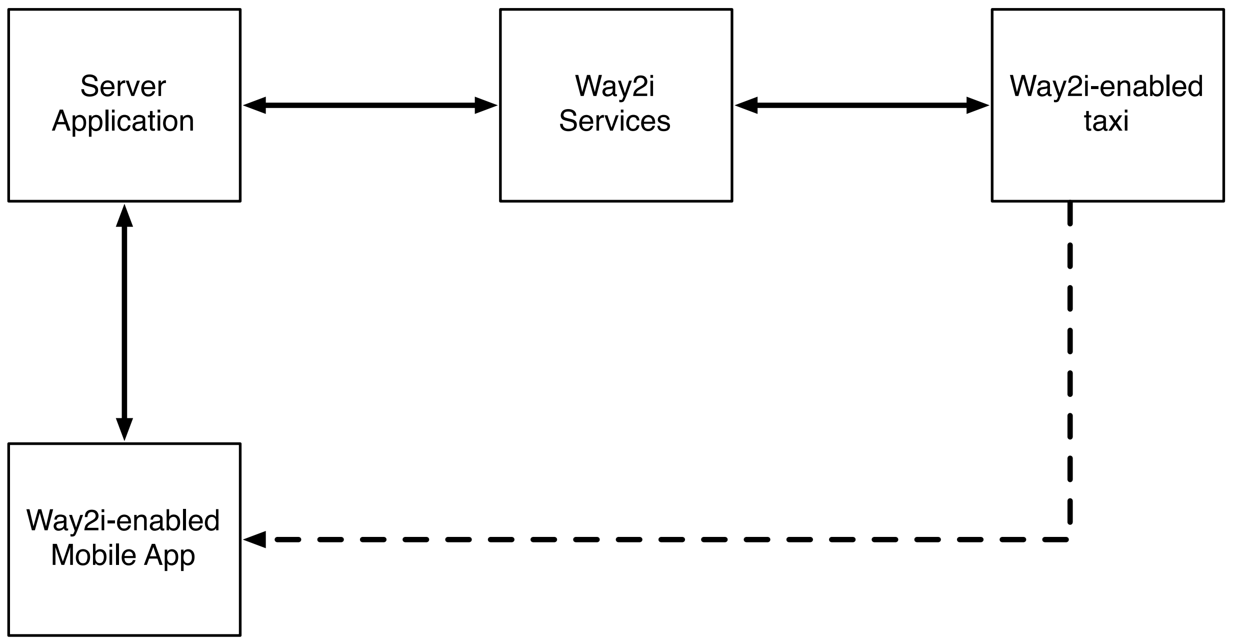

FIG 2. Way2i-enabled application architecture

Way2i-enabled application may use Way2i services both within and outside sessions established with Way2i-enabled taxis. Wallet and card management services are an example of services that do not require a session.

Payment for taxi rides, on the other hand, requires establishing a session with a taxi. The architecture of session creation is presented on the FIG2 above.

1. The Way2i-enabled taxi presents a "check-in code" associated with each new taxi ride. The check-in code is displayed on the passenger-facing screen within the taxi and is also being broadcasted wirelessly using VeriFone patented audio-based technology.

2. The Way2i-enabled mobile app needs to capture the code. For example a subscriber using the app can enter the code in the app manually.

3. The Way2i-enabled mobile app them passes over the check-in code to its server application.

4. The server application calls the Way2i method to establish a new session associated with the check-in code. It passes over to the Way2i server a URL of the callback RESTful service implemented by the Service Provider and a list of events for which it desires to receive callbacks.

5. The Way2i server calls the registered callback method when an appropriate event occurs.

1. Common Data Types

There are many different methods for serializing and de-serializing data for RESTful services, for example JSON and XML. Both of these formats are widely used, however both of them suffer from the same drawback – they are based on textual (ASCII or Unicode) representation of binary data which creates unnecessary burden for a high-performance, low latency transaction system, such as Way2i.

VeriFone Way2i interface instead chose to use the Google protocol buffers for serializing data.

Google protocol buffers ("protobuf") are "a flexible, efficient, automated mechanism for serializing structured data" and are licensed under a BSD open source license. Protocol buffers enable a very fast and efficient implementation of data exchange between the server application of a service provider and Way2i.

You can read more about Google protocol buffers at: *developers.google.com/protocol-buffers*

Way2i services can be accessed from applications implemented in a variety of different languages and frameworks. VeriFone provides a library for accessing Way2i services from Java. The library abstracts out RESTful protocol and provides Java classes and methods to access the Way2i API.

Description of common data types used in several methods follows.

All strings can contain up to 255 characters.

ResponseData is returned by most Way2i RESTful requests. It includes the following fields:

• responseTime:	timestamp of the response in the ISO 8601 format (string)
• status:	return status (int32)
• errorData:	extended error information is status is not 0 (string)
• responseID:	unique ID of the response (string)

ErrorData is included in the ResponseData if status is not 0. It includes:

• errorCode:	code of the error (int32)
• errorMessage:	human-readable message associated with the error code (string)

Attribute – a generic key-value pair

• key	Key (string)
• value	Value (string)

CardType – an enumeration of supported card types

- VISA
- AMEX
- MC
- DINERS
- DISCOVER
- JCB

PaymentInstrument: provides information about the payment instrument

• paymentInstrumentID:	ID of the payment instrument (string)
• type:	type of the payment instrument (enum)
• shortName:	automatically generated short name of the payment instrument (string)
• customName:	name assigned to this payment instrument by the customer (string, optional)
• cardType:	type of the payment card if the payment instrument is a card (enum, optional)

PaymentType: provides information on supported (by this instance/version of Way2i) payment types

▪ PAY_VTS_NYC_TAXI:	payment to NYC taxi drivers (or fleets) supported by VeriFone Transportation Systems (VTS)
▪ PAY_WAY2I:	internal payment from one subscriber to another subscriber

2. The Way2i Public Interface

The Way2i Public Interface provides access to server-side tokenization and to the first step of a two step card registration process.

Tokenization provides an interface for server-side tokenization of various payment instruments, including PAN-based credit or debit cards, bank account numbers, or PayPal accounts. Tokens have the following properties:

- Tokens are represented as 256 bit binary strings in base64 encoding and are not format preserving.
- Tokens are unique cryptographically secure nonces from which the account number cannot be calculated.
- The same payment instrument is always tokenized to the same token.

Method Name

tokenizeCard

This method implements secure server-side tokenization for any card numbers that conform to the ISO/IEC 7812 standard.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /cardToken
Payload:       PAN (passed as a string)
Response:      CardTokenResponse
```

CardTokenRequest:

• PAN:	Primary account number (the number of the credit card)

CardTokenResponse:

• Token:	256 bit based64 encoded string that represents the token
• ResponseData	

Method Name

registerCreditCard

This method registers a credit card in a wallet. This method is the first method of a two-step card registration process, and it should be confirmed by another Way2i method: confirmCreditCardRegistration

RESTful mapping:

```
HTTP Method:   POST
Resource:      /paymentInstrument/creditcard
Payload:       RegisterCreditCardRequest
Response:      RegisterCreditCardResponse
```

3. The Way2i Wallet Management Interface

Way2i provides a wallet management API that enables Service Providers to logically organize multiple payment instruments of their subscribers into abstract container objects called wallets. A wallet is identified by the wallet ID. A wallet may contain one or more payment instruments, and typically there is a one-to-one relationship between a subscriber and their wallet, however there is nothing that prevents the Way2i customer to create multiple wallets for the same subscriber or organize payment instruments in whatever fashion that suits the subscribers' business needs.

The Wallet Management API defines the following methods:

- createWallet
- createPrepaid
- registerCreditCard
- registerCreditCardWithoutValidation
- confirmCreditCardRegistration
- getRegisteredPaymentInstruments
- unregisterPaymentInstrument
- getSupportedPaymentTypes

Method Name

createWallet

This method creates a new wallet for the customer that can be associated by the customer with its subscriber. The wallet is initially empty and does not contain any payment instruments. Other wallet management API methods are used to add payment instruments to the wallet. The value of the walletID must be supplied by the customer. We recommend using a 256 bit base64 encoded strings, but it can really be any string that Way2i will associate with the newly created wallet.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /mgmt/wallet
Payload:       CreateWalletRequest
Response:      ResponseData
```

CreateWalletRequest:

- customerWalletID   provides the ID assigned to the wallet by the customer (string)

Method Name

createPrepaid

This method issues a new prepaid account and associates it with the current wallet. A prepaid account has a valid PAN and can be used as the source of funds or a destination of payment within Way2i.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /mgmt/paymentInstrument/prepaid
Payload:       CreatePrepaidRequest
Response:      CreatePrepaidResponse
```

CreatePrepaidRequest:

• walletID	ID of the wallet (string)
• name	Optional human readable name

`CreatePrepaidResponse`

- `PaymentInstrument:` `provides information about this payment instrument (see above for fields)`
- ResponseData

Method Name

registerCreditCard

This method associates an existing credit card with the wallet. During the card registration Way2i verifies the cardholder details (expiration date, CVV, and the billing address) using address verification services (AVS) if they are available for the specific card. Since this method needs to transmit the cardholder data including the PAN of the card, this method might be called directly from the mobile app over an untrusted interface when the subscriber enrolls a new card into their wallet. This method may be also called from the trusted interface, however in this case, the server from which this method is called needs to be PCI DSS certified.

If called over an untrusted interface, registerCreditCard, when received by the Way2i server is not executed but is placed in the queue until confirmed by a call confirmCreditCardRegistration over a trusted interface.

To properly identify which customer is the request associated with, the customer ID is supplied in this request when it is processed over an untrusted interface.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /mgmt/paymentInstrument/creditcard
Payload:       RegisterCreditCardRequest
Response:      RegisterCreditCardResponse
```

RegisterCreditCardRequest:

• serviceProvideID:	ID of the service provider (Way 2i customer)
• walletID:	ID of the wallet in which the card will be registered
• cardType:	type of the card
• PAN:	Card number
• CVV:	CVV
• expMonth:	expiration month
• expYear:	expiration year
• cardHolderName:	name of the cardholder (as it appears on the card)
• cardZIP:	ZIP code of the billing address
• cardName:	Human readable name that the customer wants to associate with this card

`RegisteredPaymentInstrumentResponse:`

- ResponseData: (see above)
  PaymentInstrument: (see above)

Method Name

registerCreditCardWithoutValidation

This method associates an existing credit card with the wallet. During the card registration Way2i does not perform the address verification services (AVS) on the card data.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /mgmt/paymentInstrument/creditcard
Payload:       RegisterCreditCardRequest
Response:      RegisterCreditCardResponse
```

RegisterCreditCardRequest: (see above)

`eredPaymentInstrumentResponse:`

- ResponseData: (see above)
- PaymentInstrument: (see above)

Method Name

confirmCreditCardRegistration

This method is called over a trusted interface to authenticate the previously called registerCreditCard method.

RESTful mapping:

```
HTTP Method:   PUT
Resource:      /mgmt/paymentInstrument/{walletId}creditcard
Payload:       string confirmationCode
Response:      ResponseData
```

Method Name

getRegisteredPaymentInstruments

This method returns a list (enumerates) all payment instruments registered for the current wallet.

RESTful mapping:

```
HTTP Method:   GET
Resource:      /mgmt/wallet/{walletId}/paymentInstruments
Response:      RegisteredPaymentInstrumentsResponse
```

RegisteredPaymentInstrumentResponse:

- array of PaymentInstrument structures
- ResponseData

Method Name

unregisterPaymentInstrument

Removes the association between the wallet and the payment instrument.

RESTful mapping:

```
HTTP Method:   DELETE
Resource:      /mgmt/paymentInstrument/{walletId}/{paymentInstrumentID}
Response:      ResponseData
```

Method Name

getSupportedPaymentTypes

This method returns the list of the available payments that can be made using payment instruments within this wallet. In v 1.0 of the Way2i interface the only supported payment type is VTS_NYC_TAXI, which is a payment type for paying NYC taxis as supported by the VTS eFleet system.

Additional payment types may be supported in the future Way2i versions

RESTful mapping:

```
HTTP Method:   GET
Resource:      /mgmt/wallet/{walletId}/paymentTypes
Response:      SupportedPaymentTypesResponse
```

SupportedPaymentTypesResponse:

- `paymentType`   `array of PaymentType values (see avbove)`
- `ResponseData`

4. The Way2i Session Interface

Way2i sessions exists within the context (and duration) of one taxi ride. The lifetime of a session starts when the passenger gets into a taxi and the taxi driver “hires” the meter. The lifetime of a session ends either implicitly, when the payment for the ride is processed by the Way2i system (or by the driver), or explicitly if the Way2i client calls the Way2i method to terminate the session.

Each session is linked with the check-in code of a taxi ride. When the new session is created the Way2i customer supplies to the Way2i server a session-specific URL for a RESTful endpoint. This URL will be called by the Way2i server asynchronously.

Way2i session interface uses the TaxiSessionRequest structure to pass information to Way2i

TaxiSessionRequest: an argument to session management API methods. Includes the following fields:

• medallionNumber:	the medallion of the taxi (string)
• callbackEndpooint:	the URL of the client’s callback method
• lat, lon:	the coordinates of the point from which the mobile client creates the new session (float)
• platform:	optional designator of the mobile platform. Currently defined platforms are: IOS, ANDROID, WINCE, WINRT, BLACKBERRY
• deviceID:	optional identifier of the mobile device from which the new session is established
• eventTypes:	array of TAXI_EVENT_TYPE strings that identifies to which events the client wants to subscribe.
• callbackType:	type of a callback: CHECK_IN_CODE or SESSION_ID. Depending on its type, either the check-in code or the session ID will be passed to the callback method.
• service Fee	service fee
• tipsPercentage	tips as a percentage

The following eventTypes are currently defined:

- METER_TIME_OFF – ride has completed or paused (the driver pressed the “meter time off” button)
- FARE_UPDATED – fare might get updated several times even after the driver pushed the “meter time off” button, for example when the car keeps moving.
- METER_REHIRED – the ride was resumed after the time off
- METER_PAYMENT_ACKNOWLEDGED – the fare was paid in the taxi with cash or credit card
  METER_HIRED – this event is sent when the ehail session starts

TaxiSessionResponse

- taxiSessionId: unique session identifier
- response: ResponseData

ServiceFee

- code: code (string)
- description: description (string)
- amount: amount (double)

PayTaxiRequest

- medallionNumber: the medallion of the taxi (string)
- walletId: wallet id (string)
- paymentInstrument: payment token (string)
- tips: tips (double)
- serviceFee: list of ServiceFee elements
- printReceipt: flag that indicates whether the meter should print a hardcopy receipt (boolean)

PaymentToken

- walletId: wallet id (string)
- paymentInstrument: payment token (string)
- share: fraction of the total amount that will be charged to this payment instrument should be > 0 and <= 1

PayTaxiRequestExt

- tokens: list of PaymentToken elements
- medallionNumber: the medallion of the taxi (string)
- tips: tips (double)
- serviceFee: list of ServiceFee elements
- printReceipt: flag that indicates whether the meter should print a hardcopy receipt (boolean)

ConfigurePaymentRequest

- medallionNumber: the medallion of the taxi (string)
- tips: tips (double)
- serviceFee: list of ServiceFee elements
- printReceipt: flag that indicates whether the meter should print a hardcopy receipt (boolean)

ConfigurePaymentResponse

- sessionId: session id
- response: ResponseData
- PayPreparedTaxiSessionWithTokenRequestwalletId: wallet Id
- paymentInstrument: payment instrument token

Method Name

isDriverLoggedIn

Checks if a driver is logged into a TPEP system of a taxi with the specified medallion ID.

RESTful mapping:

```
HTTP Method:   GET
Resource:      /taxi/driver/is-logged-in/{hack}/{medallionNumber}
Payload:       none
Response:      boolean
```

Method Name

createSession

Establishes a new taxi session and registers the callback URL. The newly created session is identified by the check-in code. Specific session parameters can be subsequently updated with the ***updateSession***. The session can be explicitly terminated by the Way2i client by calling ***dropSession.***

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/session/{checkInCode}
Payload:       TaxiSessionRequest
Response:      TaxiSessionResponse
```

Method Name

createSessionByMedallionNumber

This method is identical to createSession but it does not require a check-in code.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/session-by-medallion/{medallionNumber}
Payload:       TaxiSessionRequest
Response:      TaxiSessionResponse
```

Method Name

createEHailSession

This method is identical to createSessionbyMedallionNumber but it creates pending session which waits for the METER_HIRED event from the taxi. The actual session starts when the METER_HIRED event is received. If METER_ON is not received within 1 hour this session will be dropped.

RESTful mapping:

```
      HTTP Method:   POST
      Resource:      /taxi/ehail-session/{medallionNumber}
      Payload:       TaxiSessionRequest
      Response:      TaxiSessionResponse
```

Method Name

updateSession

Updates session parameters while the session is active.

RESTful mapping:

```
      HTTP Method:   PUT
      Resource:      /taxi/session/{CheckInCode}
      Payload:       TaxiSessionRequest
      Response:      ResponseData
```

Method Name

updateSessionById

Updates session parameters while the session is active or pending.

RESTful mapping:

```
      HTTP Method:   PUT
      Resource:      /taxi/ session-by-id/{sessionId}
      Payload:       TaxiSessionRequest
      Response:      ResponseData
```

Method Name

dropSession

Deletes the active session.

RESTful mapping:

```
      HTTP Method:   DELETE
      Resource:      /taxi/session/{CheckInCode}
```

```
Response:      ResponseData
```

Method Name
dropSessionById

Deletes the active or a pending session.

RESTful mapping:

```
HTTP Method:   DELETE
Resource:      / taxi/session-by-id/ { sessionId}
Response:      ResponseData
```

Method Name	Supported in Way2i version
paySession	1.0

Processes the fare payment for the active session.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/session/pay/{checkInCode}
Payload:       PayTaxiRequest
Response:      PayTaxiResponse
```

Method Name	Supported in Way2i version
paySessionExt	1.0

Identical to paySession but supports splitting the payment between several payment instruments

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/session/payExt/{checkInCode}
Payload:       PayTaxiRequestExt
Response:      PayTaxiResponse
```

Method Name	Supported in Way2i version
paySessionBySessionId	1.15

Identical to paySession but uses the session ID instead of the check in code to identify the session.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/session/pay-by-id/{sessionId}
Payload:       PayTaxiRequest
Response:      PayTaxiResponse
```

Method Name	Supported in Way2i version
paySessionExtBySessionId	1.15

Identical to paySessionExt but uses the session ID instead of the check in code to identify the session.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /taxi/ session/payExt-by-id/{sessionId}
Payload:       PayTaxiRequestExt
Response:      PayTaxiResponse
```

Method Name	Supported in Way2i version
configurePayment	1.15

Initiates the two-step payment.

RESTful mapping:

```
HTTP Method:   PUT
Resource:      /taxi/session/prepare-completion/{sessionId}
Payload:       ConfigurePaymentRequest
Response:      ConfigurePaymentResponse
```

Method Name	Supported in Way2i version
payPreparedSession	1.15

Completes the two-step payment

RESTful mapping:

```
HTTP Method:   PUT
Resource:      /taxi//session/pay-prepared-with-token/{sessionId}
Payload:       PayPreparedTaxiSessionWithTokenRequest
Response:      PayTaxiResponse
```

5. The Way2i Payment Interface

The Way2i payment interface supports payment transactions and sending/receiving funds, when supported by the Way2i payment instrument tokens.

The following methods are included in the payment interface:

- pay
- payExt
- rollbackPayment
- getPaymentState
- topup
- registerAutoTopup
- getAutoTopupInfo
- getBalance
- getHistory

Common data types for Way2i Payment Interface:

PaymentInfo: a JSON object that contains the following fields

- state: state of the payment (XXX) (string)
- amount: amount of the payment (string)
- currency: the ISO 4217 currency code (string)
- timestamp: the timestamp of the transaction in the ISO 8601 format (string)
- paymentType: payment type of that payment transaction (PaymentType) paymentAttribu array of payment attributes (Attribute)

PaymentRequest:

- paymentInstrument: the token of the payment instrument (string)
- wallet: the token of the wallet of the payment instrument (string)
- pan: credit card number (string)
- expMonth: credit card expiration month (string)
- expYear: credit card expiration year (string)
- amount: the amount (string)
- currency: the ISO 4217 currency code in which the payment must be delivered to the payee (string)
- paymentType: one of the supported payment types. Currently supported values: PAY_WAY2I, PAY_VTS_NYC_TAXI
- payeeAttributes: list of Attribute elements (key/value pair) that depend on the payee
- SplitPaymentInstrumentpaymentInstrument: the token of the payment instrument (string)
- wallet: the token of the wallet of the payment instrument (string)
- pan: credit card number (string)
- expMonth: credit card expiration month (string)
- expYear: credit card expiration year (string)
- amount: the amount (string)

- currency: the ISO 4217 currency code in which the payment must be delivered to the payee (string)

PaymentRequestExt

- paymentInstruments: list of SplitPaymentInstrument
- paymentType: one of the supported payment types. Currently supported values: PAY_WAY2I, PAY_VTS_NYC_TAXI
- payeeAttributes: list of Attribute elements (key/value pair) that depend on the payee

PaymentResult: a JSON object that includes the following elements

- returnCode: return code
- extendedAuthCode: extended transaction authorization code that includes the following concatenated strings:
  - payment gateway auth code (6 characters)
  - first two digits of the PAN of the payment instrument (if available or 00 if the payment instrument does not have an ISO 7812 PAN
  - last four digits of the PAN (or identifier of the non-PAN payment instrument) NOT AVAILABLE IN v1.0 of Way2i API
- transactionID: 256 bit base64-encoded transactionID that uniquely identifies the payment transaction

PaymentResponse: a JSON object that includes the following elements

- paymentResult: a PaymentResult object
- paymentInfo a PaymentInfo object
- response a ResponseData object

PaymentStatusResponse:

- paymentInfo: information about the payment (PaymentInfo)
- responseData: common response data (ResponseData)

PaymentRollbackResponse:

- paymentResult: a PaymentResult object
- response: a ResponseData object

TopUpRequest

- wallet: ID of the subscriber's wallet (string)
- sourcePaymentInstrument: token of the payment instrument from which the money will be withdrawn (string)
- targetPaymentInstrument: token of the payment instrument (that must be a prepaid account) into which the money will be deposited (string)
- amount: amount (string)
- currency: the ISO 4217 currency code in which the transaction has to be processed (string). Default value if missing: "USD"

TopUpResponse

- paymentResult: a PaymentResult object
- response: a ResponseData object

AutoTopUp:

- amount: amount in which the automatic top up transaction has to be processed
- currency: the ISO 4217 currency code in which the transaction has to be processed (string). Default value if missing: "USD"
- disabled: indicates whether this request is in the enabled or disabled state. You can disable and re-enabled the auto top-up request (bool)
- sourcePaymentInstruments: an ordered array of payment instruments from which the top-up transaction will be processed. Processing will always start with the first payment instrument, and if the top-up transaction fails, then the Way2i auto top-up handler will try the second, and so on until the top-up transaction is processed successfully or the list has been exceeded (list of strings)
- threshold: the top-up threshold. The auto top-up will be processed if the amount in the prepaid account for which it is registered drops below this threshold.

AutoTopUpRequest:

- wallet: ID of the subscriber's wallet (string)
- targetPaymentInstrument: the token that represents the prepaid account for which the auto top-up is registered
- autoTopUp: the AutoTopUp object

AutoTopUpResponse:

- autoTopUp: AutoTopUp object
- response ResponseData object

BalanceResponse:

- amount: available balance
- currency: currency of the account
- responseData: a ResponseData object

HistoryResponse:

- payments: an array of PaymentInfo objects
- response an array of ResponseData objects

ServiceFee – used by the Way2i users to include service fee amounts and descriptions when paying for the rides

- code: service fee code (string)
- description: service fee description (string)
- amount: service fee amount (double)

PayPreparedTaxiSessionRequest

- pan: credit card number
- expMonth: credit card expiration mouth in the format MM
- expYear: credit card expiration year in the format yyyy

Method Name

pay

A generic payment method that authorizes and processes a payment transaction. The list of supported payment methods is returned by the getSupportedPaymentTypes operation for a specific wallet.

Currently, the following payment types are supported:

- PAY_VTS_NYC_TAXI: payment to NYC taxi drivers (or fleets) supported by VeriFone Transportation Systems (VTS).
- PAY_WAY2I: internal payment from one subscriber (wallet) to another subscriber (wallet)

RESTful mapping:

```
HTTP Method:   POST
Resource:      /operations/payment
Payload:       PaymentRequest
Response:      PaymentResultResponse
```

Method Name

payExt

This method identical to pay but allows split payment.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /operations/paymentExt
Payload:       PaymentRequestExt
Response:      PaymentResponse
```

Method Name

rollbackPayment

This method cancels the payment transaction identified by its transaction ID. The mechanism for rolling back credit card transactions depends on the specific payment method and the wallet. If the credit card transaction was rolled back before the batch processing time as defined by the payment gateway for a specific wallet and payment type, the transaction may be cancelled, and if it was roll back after being processed by the gateway, it will be charged back.

RESTful mapping:

```
HTTP Method:   DELETE
Resource:      /operations/payment/{transactionID}
Payload:       none
Response:      PaymentRollbackResponse
```

Method Name

getPaymentState

This method returns information about the status of the payment request

RESTful mapping:

```
HTTP Method:   GET
Resource:      /operations/payment/{transactionID}
Payload:       none
Response:      PaymentStatusResponse
```

Method Name

topup

This method tops up a prepaid account created for the subscriber (or customer) by Way2i.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /operations/topup
Payload:       TopUpRequest
Response:      TopUpResponse
```

Method Name

registerAutoTopup

This method creates a Way2i server based automatic top-up handler. As soon as the amount in the prepaid account that is passed to this method drops below a defined threshold, it will be automatically topped up.

RESTful mapping:

```
HTTP Method:   POST
Resource:      /operations/autotopup /{walletId}/{paymentInstrumentId}
Payload:       AutoTopUpRequest
Response:      ResponseData
```

Method Name

getAutoTopupInfo

Returns info about the previously registered (with registerAutoTopUp) automated top-up.

RESTful mapping:

```
HTTP Method:   GET
Resource:      /operations/autotopup/{walletID}/{paymentInstrumentID}
Payload:       none
Response:      AutoTopUpResponse
```

Method Name

getBalance

Returns the available balance on the payment instrument. Not all payment instruments support this operation. For example, credit cards do not support it.

RESTful mapping:

```
HTTP Method:   GET
Resource:      /operations/balance/{walletID}/{paymentInstrumentID}
Payload:       none
Response:      BalanceResponse
```

Method Name

getHistory

Returns the history of transactions on the payment instrument. Only transactions processed through Way2i are included in history, so if this method is called on a credit card token, transactions processed outside of Way2i will not be included in the history.

RESTful mapping:

```
HTTP Method:   GET
Resource:
               /operations/history/{walletID}/{paymentInstrumentID}/{st
art}
Payload:       none
Response:      HistoryResponse
```

Method Name

payPreparedSession

Completes the two-step payment

RESTful mapping:

```
HTTP Method:   POST
Resource:      /operations/pay-prepared/{sessionId}
Payload:       PayPreparedTaxiSessionRequest
Response:      PayTaxiResponse
```

Appendix 1 - Return Codes - Errors.

Error Code	Error Description
1	General error
2	PAN is not valid
3	Authentication has failed
4	Internal Way2i error
5	Invalid wallet
6	Wallet already exists
7	Credit card is already assigned
8	Confirmation of the enrolled credit card failed
9	Payment instrument is not found in the given wallet.
11	Invalid credit card expiration date. Expiration date should be formatted as MM/YY or MM/YYYY, and it should not be in the past
12	Invalid PAN. PAN should contain only digits, can't be empty and its length is determined by the card type: VISA – 13, 16 AMEX – 15 MC – 16; DINERS – 14, 16 DISCOVER – 16 JCB – 16
13	Invalid ZIP. ZIP should contain 5 digits.
14	Invalid cardholder name. Cardholder name can't be empty and its length should be below 26 characters.
15	Invalid CVV. CVV can't be empty and should contain only digits Length of CVV is determined by the card type: VISA – 3 AMEX – 4 MC – 3 DINERS – 3 DISCOVER – 3 JCB – 3
16	Invalid card type. The following card types are allowed: VISA, AMEX, MC, DINERS, DISCOVER, JCB
17	Reserved for future use
18	Reserved for future use
19	Reserved for future use
20	Reserved for future use
21	Payment general error
22	Unsupported payment type.
23	Invalid combination of a payment instrument and wallet
24	Invalid amount. Should be greater than 0 and can't be empty
25	Invalid currency code.
26	Reserved for future use

27	Reserved for future use
28	Unsupported type of payment instrument for a given operation. E.g. attempt to get balance of a credit card will return this error.
29	Invalid currency
30	Transaction not found
31	Unable to rollback the transaction
32	Internal error communicating with the VeriFone payment gateway
33	Card has been declined
34	Card verification has failed
35	Internal error communicating with the VeriFone payment gateway
36	Error processing the transaction
37	Internal payment error
38	Reserved for future use
39	Reserved for future use
40	Reserved for future use
41	Reserved for future use
42	Reserved for future use
43	Taxi session was not found
44	Ride info is not found
45	Internal Way2i error
46	Taxi payment has failed
47	Invalid value for share should be > 0 and < 1. Sum of all shares in the request should be 1
48	Invalid medallion number