SPRINTER Futárszolgálat Kft.
H-1097 Budapest, Táblás utca 39.
SPRINTER IT Support
E-mail: pudointerface@lapker.hu

PUDO Partner Interface specification

Prepared by:

SPRINTER Futárszolgálat Kft.

Budapest, 2021.

SPRINTER Futárszolgálat Kft.
H-1097 Budapest, Táblás utca 39.
SPRINTER IT Support
E-mail: pudointerface@lapker.hu

Document follow-up

Date	Modified by	Changes	Version
23.02.2017	Kovács, Richárd	Creation	1.0
28.02.2017	Szűcs, Krisztián	Finalization	1.1
21.06.2017	Kovács, Richárd	Token field	1.2
27.10.2017	Kovács, Richárd	Return of document	1.3
22.03.2018	Szűcs, Krisztián	Tracking	1.4
13.07.2018	Szűcs Krisztián	Parcel values	1.5
29.07.2019	Szűcs Krisztián	Printing labels, A/6	1.6
16.11.2020	Tóth, Dávid	ZPL	1.7
01.02.2021	Zagyi, Péter	ThirdPersonDelivery	1.8

SPRINTER Futárszolgálat Kft.
H-1097 Budapest, Táblás utca 39.
SPRINTER IT Support
E-mail: pudointerface@lapker.hu

Table of contents

1. Introduction

The interface document is aimed at specifying and presenting with examples how to register a parcel for Sprinter services (Pick Pack Pont and Home delivery) from an external application, furthermore the document generator function is also presented. The document discusses in detail the technical implementation and business rules of accessing the function. Demo codes (.Net, C# and PHP) are enclosed in the document as well. The demo codes are built up with taking account of rules of data structures creation.

2. Communication

The communication is performed through an HTTP protocol with messages described in the HTTP 1.1 specification and with UTF-8 encoding. The client starts a transaction by sending a HTTP request to which an HTTP response is given. WCF service is used during the communication.

2.1. General information

The parameter type means an XSD type (*XML Schema Definition Language,* http://www.w3.org/TR/xmlschema-0/).

Explanation of a few data type:

int
- Its values: 4-byte signed number (+2,147,483,647 through 2,147,483,648)
- Represented by: a string of digits (e.g. „14400”)

boolean
- Its values: true/false
- Represented by: „true” or „false”, „1” or „0”

dateTime
- Its values: valid dates are AD 1 to 9999
- Represented by: „CCYY-MM-DDThh:mm:ss”, where ’CC’ denotes the century, ’YY’ the year, ’MM’ the month, ’DD’ the day, ’hh’ the hour, ’mm’ the minute, ’ss’ the second, filled with zeros from the left. ’-’, ’T’ and ’:’ are mandatory hyphens. (e.g.. 2003-05-05T16:41:57)

If an error occurs during the method call, the form of the response given to it depends on the level at which the error occurred:

- HTTP error message is the response to HTTP-level errors (e.g. Not Authorized)
- SOAP Fault message is the response to SOAP-level errors (e.g. in the XML representation a date parameter is incorrect)
- DocumentGenerationResult is the response to PUDO application-level errors (e.g. parcel does not exist).

3. Parcel registration

3.1. Parcel registration in general

The Service call to be presented below describes parcel registrations performed by a partner.

Sending a mixed order is possible during the parcel registration, which means you can register Pick Pack Pont (henceforth abbreviated as PPP) and Home delivery (henceforth abbreviated as HD) parcels within a Service call, it is not required to start a new Service call at each delivery type.

Parcels sent within **one** service call are registered in **one** shipment **per delivery type**! It is important, because – after a successful registration - the label and document generator functions will send different documents (PPP/HD) back in the response. Parcels sent from the same drop-off point and / or coming to the same destination can be placed in the same shipment.

Maximum of 1000 parcels can be registered in the system in one call. Due to security reasons more parcels than that cannot to be processed and PSR_TOO_MANY_PARCEL_IN_REQUEST error message will be sent back. It is not possible to perform partial parcel registration. In case of **Home Delivery** parcels it is possible to register sub-parcels within a parcel data (ParcelCount field – these parcels' data is the same, since all of them go to the same receipient), maximum number of sub-parcels is 100. More parcels than that cannot be processed and PSR_TOO_MANY_PARCEL_IN_REQUEST error message will be set on parcel level. Other parcels in the request can be registered. In case **of Pick Pack Point** parcels ParcelCount field is not to be used, since only single parcels can be processed. Maximum of 1000 parcels can be registered per request. Based on the above mentioned information more parcels cannot be registered.

The maximum length of barcodes is 25 characters per parcels.

Every partner has a unique identifier (partner code). This way it is possible to ensure, that parcel registration is connected to that specific partner.

UNIQUE IDENTIFIER (PREFIX) IS DIFFERENT FOR EACH PARTNER. THE PREFIX HAS TO BE REQUESTED FROM SPRINTER IT SUPPORT (pudointerface@lapker.hu)!

This code will be visible on parcel label too.

FURTHER SECURITY FIELD, A SO-CALLED TOKEN (10 DIGIT OF NON-CAPITAL AND NUMERIC CHARACTERS) IS ALSO NEEDED TO BE REQUESTED FROM SPRINTER IT SUPPORT! THIS IS A PARTNER SPECIFIC UNIQUE IDENTIFIER, CAN BE MODIFIED BY SPRINTER IT SUPPORT ONLY.

Sending this value is mandatory. If a token is wrongly sent or has not sent upon parcel registration, then parcel registration became invalid. In this case **PSR_INVALID_PARTNER_TOKEN** error message will be sent back by PUDO for all parcels can be found in concerned request.

We receive a response named **RegisterParcelContainerResponse** concerning the success of the registration attempt (in which the **ErrorCode** field informs whether the parcel registration was

successful or not), and gets back a list named **ParcelResult,** in which you can find parcels' data for each parcel.

If the parcel registration was successful, PSR_OK response returns in the **ErrorCode** field. Further error codes will be discussed later in the document.

3.2. Pick Pack Pont

There are 4 different cases at Pick Pack Pont (henceforth abbreviated as PPP) parcel registration.

Simple parameterization can be used to define the required sub-service type in the program code, when we do the parcel registration. These settings will be specified later in the document.

The dispatch point and the destination can be specified with fields **`PickupLocationId`** and **DestinationLocationID.** These LocationIDs identify PPP shops, and shop IDs have to be given in the service call. These data are available e.g. in the shop locator (iframe to embed): http://online.sprinter.hu/terkep/#/

3.2.1. Normal – Parcel dispatched from depot to PPP shop

Dispatching a normal parcel means, that the dispatch location is the partner's default pickup depot, whereas the destination location is a PPP shop. In this case only the **`DestinationLocationId`** has to be defined.

3.2.2. POS2POS – Parcel dispatched from a PPP to another PPP shop

Dispatching a POS2POS parcel means, that the parcel is sent from a PPP to another PPP shop. In this case both LocationIDs must be given, **`PickupLocationId`** field defines the dispatch / pickup location and **`DestinationLocationID`** field defines the destination, where the parcel has to be handed over to the customer.

IF YOU REGISTER A POS2POS PARCEL WITH SAME PICKUPLOCATIONID AND DESTINATIONLOCATIONID, SO SHOPIDS ARE THE SAME IN BOTH FIELDS, THEN "DIRECT SERVICETYPE" IS EXECUTED. IN THIS CASE THE "DIRECT" SETTINGS MUST BE APPLIED.

3.2.3. Direct – Parcel with the same pickup and destination location

Dispatching a Direct parcel means, that the parcel is dispatched from a PPP and delivered to customer in the same PPP, so the sender leaves the parcel on a certain PPP shop, it remains physically in the shop until handing it over to the customer. In this case only `DestinationLocationID` must be defined (filled **`PickupLocationId`** is not required).

3.2.4. Return – Parcel to be returned from a PPP shop

Dispatching a Direct parcel means, that the parcel has to be returned to the sender, so the customer dispatches the parcel from a PPP and - after the return transport - the partner can take it over in their supply depot. In this case only **`DestinationLocationID`** must be defined.

3.3. Home delivery

There are 1 ServiceType at Home delivery service (henceforth abbreviated as HD), but the delivery method (DeliveryType) can be different at this service type.

Delivery types are the following:

- **OnlyDelivery:** Delivery, when the partner sends the parcel (the courier transports it from the partner's site or the partner brings it to the depot) and it will be delivered to the customer's unique address.
- **OnlyReturns: "**Customer return" means that the parcel is picked up at the customer's unique address and returned to the partner.
- **ThirdPersonDelivery:** The parcel is picked up at a third person's unique address and delivered to the customer's unique address. The marketing team has to approve this type of delivery for the partner to use it.

The unique address is built up with the following data: Country code, Postal code, City, Street, House number.

Simple parameterization can be used to define the required sub-service type upon parcel registration. These settings will be specified later in the document.

3.3.1. International home delivery - FanCourier

Currently HD parcels can be shipped only to Romania with FanCourier courier service. In this case not only "RO" has to be given in CountryCode, but also SupplimentJSONData has to be filled up correctly with the supplier specific data. Further details can be found in the source code.

In this case the NewBarcode field is always filled up upon registration -parcel by parcel - with a barcode given by FanCourier (e.g. 20467001200260001F0500). The AWB number can be found in the ShipmentID field, which number serves as a consignment note ID, this one is shown in tracking information.

FanCourier generates own shipment label, which has to be placed on the parcel. The error message returned by FanCourier can be found in the ErrorMessageInternational field in the response. Further information about label generation will be presented later in the document.

3.4. Replacing the barcode

AFTER SUCCESSFUL PARCEL REGISTRATION THE NEWBARCODE FIELD IS ALSO RETURNED, IN WHICH WE MAY FIND A BARCODE DIFFERENT FROM THE ONE PREVIOUSLY GIVEN BY US. IF CONERNED BARCODE HAS ALREADY BEEN REGISTERED IN PUDO, THEN NEWBARCODE FIELD CONTAINS A NEW AND UNIQUE BARCODE GENERATED BY PUDO. THIS ID HAS TO BE STORED, BECAUSE YOU CAN REQUEST FOR THE LABEL BASED ON THE NEW BARCODE, AND THAT BARCODE CAN BE SEARCHED IN TRACKING INFORMATION.

NewBarCode can be different from OriginalBarCode in the following cases:

- The caller resends a barcode, which has already been successfully registered in PUDO.
- The sent barcode already exists in PUDO. (Previously another partner sent and registered successfully the same barcode. In order to avoid the barcode collision a unique BarCodePrefix is assigned to each partner, which prefix has to be built in the barcode.)

- The **Response** is not returned in time (**TimeOut**) due to internet connection problem, but the parcel has already been registered in the system. In order to have a successful **Response** the same barcode is sent again. If we sent a barcode and an error message returns due to **TimeOut,** then we suggest launching a Request for the label based on that barcode. If the label is not generated, then that barcode is not registered / does not exist in PUDO system.
- If multiple parcel shipment is registered (ParcelCount > 1), then NewBarCode field contains the different barcodes parcel by parcel (Barcode, Barcode-2, Barcode-3…). (Registration of a multiple parcel shipment is possible only at Home delivery service.)
- The foreign logistics sub-contractor generates new barcode at international transport. (e.g. FanCourier -> 2347600120021001F1200)

3.5. Document return service

3.5.1. Pick Pack Point

In such a case, when we wish to get a contract /document/note signed by the recipient back. Return of document parcel registration offers our contracted partners a possibility to cover this need. For having the above mentioned service previous discussion with our Sales department - having a contractorial background - is necessary. Please note that parcel registrations without having right will be failed. Upon returning a document it is necessary to put an envelope in the parcel to be shipped. Please make sure that the return label is sticked onto the envelope. After a successful delivery, the shopkeeper will put the return document in the envelope (to be sent back to the original sender).

In case of Document return service True value should be set - during Normal/POS2POS/Direct parcel registration - in the ReturnofDocument field. Upon data transmission a Return type parcel will be atomatically generated in the system, at which the barcode is the original barcde + an „R” digit. This value will be sent back in the response too. In case of call-based label generation this barcode (e.g. OriginalBarcode + R = TESTBARCODER) is needed to be sent in the request, afterwards based on the above mentioned process, the physical label should be placed in the parcel.

Please note: the above mentioned restrictions are not true for Home delivery service.

3.5.2. Home Delivery

For using Document return service one HD parcel (way bill) is to be registered. The important difference is that document return is provided for multiple parcels as well.

When a parcel is registered with ServiceType = HomeDeliver and the value of ReturnOfDocument=true, then we can attach to the concerned parcel a document to be returned.

3.6. Description of types and fields

3.6.1. Most important fields

Paying attention to the following fields is very important, because an incorrect setting or a skipped, but required setting (which can be handled by the system with default value) may cause unintended modification in the life path of the parcel. **Please get in touch with your sales manager concerning those fields, because the options depend on your contract and can have different remuneration.**

3.6.1.1. Service Type

This parameter defines which service type is used; the separated types can have different shipment method and fees. There are 4 different categories within the Pick Pack Pont service: Normal, Pos2Pos, Direct, and Return. It can be specified the home delivery service with HomeDeliver. Detailed description is available here.

3.6.1.2. Transit Time

This field has to be filled only at home delivery service, but not required at PPP service. This value defines the delivery time of the parcel. Possible values can be found here.

3.6.1.3. ParcelCount

This field defines at home delivery service how many parcels have to be delivered together. It can happen due to the parcel size or weight, that more parcels are shipped physically, but they have to be delivered together for just the same addressee. In this case a given parcel quantity is created with the following barcode format: Barcode, Barcode-2, Barcode-3. These numbered barcodes return in the response in the NewBarcode field and separated labels are generated for each parcel.

3.6.1.4. ReturnOfDocument

True value is to be inserted in this field in case of Parcels to be registered to Pick Pack Point with document return service. The field indicates that the document attached to the parcel is to be signed by the addressee before parcel handover.

3.6.2. Functions

RegisterParcelContainer

Return object type: `return RegisterParcelContainerResponse`
Parameters: `RegisterParcelContainerRequest request`
Mandatory field: **Yes**

Type: Function
Description:

This is the general parcel registration, for the value **RegisterParcelContainerRequest** we get **RegisterParcelContainerResponse** data. The original barcodes and/or new barcodes (if they were generated) of the registered parcels are returned in the response, whereas the ErrorCode informs whether the registration was successful or not.

3.6.3. Types

3.6.3.1. Request

RegisterParcelContainerRequest

Type: Class
Description:

This class performs the Request required for parcel registration. It needs partner ID, list of parcels and a supply mode field, the lattest one will be disctontinued in the future.

Please note that longer values than the given (Type) string of characters will be cut off automatically!

Fields:

Type	Field name	Remark	Mandatory
string (10)	**PartnerCode**	Partner (sender) code	**Yes**
string	**PartnerAddress**	**This field will be discontinued.** Partner address	No
List<Parcel>	**ParcelContainer**	Bunch of parcel data, which contains the parcel's list to be registered. It depends on the setting of the Parcel item, which service type	**Yes**

		will be registered.	
DateTime	**ArriveDate**	Expected delivery time. It is not shown on any surface.	No
Supplier	**Supplier**	**This field will be discontinued.** ENUM for differentiating the supplier type. It is set based on PPP and HD supplier types stored in the Master data. Currently the field must be filled per Partner.	**Yes**
string(10)	**Token**	A field verifiying the partner's package sending. 10 character long, it contains non-capital letters and numbers.	Yes
bool	**OnlyLabelPrinting**	Data transfer for the sole purpose of label printing	No
bool	**FinalizeLabelPrinting**	Finalizing label printing, transfering the package register data	No
string	**SupplimentJSONData**	Other data	No

Parcel

Type: Class

Description:

This class executes the parcel registration; it depends on the parametrization which parcel type is registered.

Field:

Type	Field name	Remark	Mandatory
dateTime	**AccountingDeadlineDate**	Payment deadline	**No**
string (28)	**AutorizationCode**	Order code, which is the partner's own parcel code or invoice number (if an invoice can be found in the box).	Yes
string (25)	**BarCode**	Parcel barcode	**YES**
string (64)	**ContactName**	Contact name	NO
string (256)	**CustomerAddress**	Customer's address (street)	**YES**
string (25)	**CustomerCity**	Customer's address (city)	**YES**
string (10)	**CustomerCode**	Customer code, if it exists	No
string (2)	**CustomerCountryCode**	Customer's country code	NO
string (64)	**CustomerEmail**	Customer's e-mail address	**YES**
string (256)	**CustomerName**	Customer's name	**YES**
string (64)	**CustomerPhone**	Customer's mobile phone number for sending SMS notification, format: +36201234567	**YES**
string (8)	**CustomerPostalCode**	Customer's postal code	**YES**

string (40)	**CustomerRegion**	Customer's region/county	NO
string (40)	**CustomerStreetNumber**	Customer's house number, and further address details (floor, door)	**YES**
CustomerType	**CustomerType**	Customer type (always B2C)	NO
decimal(9,2)	**DeliveryPrice**	**This field will be discontinued.** Previously it contained the transport fee; currently the system calculates the value of the field.	NO
string(3)	**DeliveryPriceCurrency**	Currency of the transport fee	NO
ParcelDeliveryType	**DeliveryType**	It defines the delivery type.	In case of HD
string (100)	**Description**	Remarks about the package and/or message for the courier	NO
string (10)	**DestinationLocationId**	The ID of destination PPP, where the parcel has to be delivered. If the destination is the central depot of Lapker, then 2000 has to be used as destination code. Always ShopCode has to be given.	**YES** In case of PPP
string(28)	**DirectPackageBarCode**	**This field will be discontinued.** Barcode of Direct parcel type	NO
string (64)	**InvoiceNumber**	**This field will be discontinued.** Invoice number placed in the parcel, AuthorizationCode takes its place.	NO
boolean	**IsPartnerInvoiced**	This field sets whether the invoice has to be made out for the Partner or not.	NO
decimal(9,2)	**PackagePrice**	Value of the parcel (That value is charged in case of compensation). Maximum value: 13 000 000 HUF. Only integer numbers can be given. Decimals will be rounded by the system according to matachematic rules of rounding.	YES
string(3)	**PackagePriceCurrency**	Currency of parcel price, default HUF	YES
decimal	**PackageSizeX**	Parcel size	In case of HD
decimal	**PackageSizeY**	Parcel size	In case of HD
decimal	**PackageSizeZ**	Parcel size	In case of HD

PackageType	**PackageType**	Parcel type according to the size	In case of PPP
decimal	**PackageVolume**	Volume of the parcel (m^3)	In case of HD
decimal	**PackageWeight**	Weight of the parcel in kg, maximum weight is 20kg	In case of HD
string (64)	**ParcelContentDescription1**	Description of the parcel content	NO
string (64)	**ParcelContentDescription2**	Product name	NO
int	**ParcelCount**	The quantity of the parcels, which have to be delivered together in case of HD. The parcel number within a Parcel object can be given in this field. E.g.: ParcelCount = 100 means, that 100 pieces of parcels will be registered.	In case of HD
ParcelCreationStatus	**ParcelCreationStatus**	The actual status of parcel creation, the request has to be sent always with “Create” status.	YES
string (64)	**PickupCustomerName**	Sender name	NO
string (10)	**PickupLocationId**	Dispatch location, ShopCode has to be given.	In case of POS2POS
string (64)	**PickupNotificationEmailAddress**	Sender’s email address	NO
string (32)	**PickupNotificationPhone**	Sender’s phone number	NO
decimal(9,2)	**PriceAtDelivery**	Cash on delivery, price to be paid at the parcel handover. Maximum amount: 700 000 HUF. Only integer numbers can be given. Decimals will be rounded by the system according to matachematic rules of rounding.	YES
string (3)	**PriceAtDeliveryCurrency**	COD currency, default HUF	YES
string (28)	**ReferenceBarCode**	Reference barcode, concerned values are displayed in the accounting files. In case of document return this will be a reference data for the return parcel in the system.	NO
dateTime	**RegisterDate**	Date of the dispatch	NO
boolean	**ReturnOfDocument**	Claim for returning a document. In case of Pick Pack Point parcels the ReferenceBarcode field is to be filled.	NO
ParcelServiceType	**ServiceType**	Delivery type. It can be decided based on	YES

		the given value at parcel registration which service type has to be applied at the parcel.	
string	**SupplimentJSONData**	Other, specific data can be sent in this field.	NO
boolean	**Tracking**	**This field will be discontinued.** Claim for Tracking information	NO
int	**TransitTime**	Delivery time. Possible values are: 0 – Before 9 o'clock 1 – forenoon between 9-12 o'clock 2 - 1 workday 3 - 2 workdays 4 - 3 workdays 5 - 5 workdays	In case of HD
ThirdPersonDelivery	**ThirdPersonDelivery**	Senders data in case of third person delivery type.	NO

Supplier

Type: Enumerator

Description:

Supply method, which defines the party who transports the parcel to the logistic starting point. This enumerator will be discontinued in the near future, but currently it must be filled with Partner value.

Fields:

Name	Remark
Partner	Partner brings the parcel
Lapker	Lapker brings the parcel
Customer	Sprinter brings the parcel

CustomerType

Type: Enumerator

Description:

It is used for differentiating customer type, B2C must be given.

Fields:

Name	Remark
B2B	Business-to-Business type
B2C	Business-to-Consumer type
C2C	Consumer-to-Consumer type

ParcelDeliveryType

Type: Enumerator

Description:

It defines the delivery mode. The **ParcelServiceType** can be used only at home delivery service.

Fields:

Name	Remark
ThirdPersonDelivery	Third person delivery
DeliveryAndReturns	Delivery and return
OnlyDelivery	Only delivery
OnlyReturns	Only return
None	Default value, the system automatically sets as OnlyDelivery

PackageType

Type: Enumerator

Description:

Classification based on the parcel size, it can be used only at PPP service.

Fields:

Name	Size	Remark
Small	X: 20cm, Y: 30cm, Z: 10cm, Volume: 0.006000m^3	Small
Medium	X: 30cm, Y: 30cm, Z: 20cm, Volume: 0.018000m^3	Medium
Large	X: 50cm, Y: 50cm, Z: 50cm, Volume: 0.125000m^3	Large
Special	X: 60cm, Y: 60cm, Z: 60cm, Volume: 0.216000m^3	Special
None		Size is not given

ParcelCreationStatus

Type: Enumerator

Description:

This enum defines the processing method of the parcel data sent.

Fields:

Name	Remark
Create	Creation of a parcel

ParcelServiceType

Type: Enumerator

Description:

This enum defines the service type. Separated settlements will be created for each type of services.

Fields:

Name	Remark
Normal	Normal PPP parcel, dispatch from depot
Direct	The dispatch point and the

	destination is the same
Pos2Pos	Delivery from a PPP to another PPP
HomeDeliver	Home delivery, courier performs the delivery
Return	Return parcels sent from a PPP

ThirdPersonDelivery

Type: Osztály

Description:

It contains the sender data in case of third person delivery type. The mandatory fields are mandatory in case the delivery type is third person delivery.

Fields:

Type	Field name	Remark	Mandatory
string	**Name**	Sender name	YES
string	**CountryCode**	Sender country code	YES
string	**PostalCode**	Sender postal code	YES
string	**City**	Sender city	YES
string	**Street**	Sender street	YES
string	**StreetNumber**	Sender street number	YES
string	**Floor**	Sender floor	NO
string	**RoomNumber**	Sender room number	NO
string	**EmailAddress**	Sender email address	NO
string	**PhoneNumber**	Sender phone number	NO

3.6.3.2. Response

RegisterParcelContainerResponse

Type: Class

Description:

This class performs the Response required for parcel registration. The response gives information whether the registration was successful or not, and in case of the original barcode changed the new one is returned in the response.

Fields:

Type	Filed name	Remark
string	**ErrorCode**	Error code
List<ParcelResult>	**ParcelResults**	Parcel registration result,

		information

ParcelResult

Type: Class

Description:

The response gives information whether the registration was successful or not. It contains the original barcode and sometimes the shipment ID, whereas in case of the original barcode changed the new one is also returned in the response.

Fields:

Type	Field name	Remark
string	**ErrorCode**	Error code
string	**OriginalBarCode**	Original barcode, which was sent at the registration
string	**NewBarCode**	New barcode
string	**OriginalDirectPackageBarCode**	Not used anymore
string	**NewDirectPackageBarCode**	New barcode of a physical parcel, if it had to be re-generated. Not used anymore
string	**ShipmentID**	Shipment ID connected to the parcel, which is generated by the system. In case of PPP service, 1 ID is generated per ServiceType. In case of HD service each Parcel has a separated ID.
string	**DirectionNumber**	Route number of the delivery address. In case of HD: postal code -> route number, in case of PPP route from depot to PPP
string	**DirectionNumberSender**	Route number of pickup address (Only at HD service and return)
string	**ErrorMessageInternational**	Response from international call
boolean	**IsDelayedDelivery**	Delayed delivery. Not used anymore.

3.6.3.3. Error codes

ErrorCode

Type: String

Description:

Possible error codes and their explanations.

Fields:

Shipment level

Name	Remark

PSR_OK	Parcel or shipment registration was successful
PSR_FAILED	Parcel or shipment registration was unsuccessful due to unknown error
PSR_PARTNERCODE_MISSING	Partner ID is missing
PSR_INVALID_PARTNER	Partner ID is not found or set incorrectly
PSR_PARTNER_MISSING_COURIER_PREMISE	Missing the premise of the courier, please call the Customer Service
PSR_PARTNER_NOT_FOUND	Partner code is not found
PSR_EMPTY_SHIPMENT	Empty parcel list
PSR_SHIPMENT_NOTENUMBER_ALREADY_USED	Shipment ID has already been used.
PSR_SHIPMENT_NOTENUMBER_MISSING	Missing shipment ID
PSR_SHIPMENT_NOTENUMBER_NOT_FOUND	Shipment ID does not exist

Parcel level

Name	Remark
PSR_OK	Parcel or shipment registration was successful
PSR_FAILED	Parcel or shipment registration was unsuccessful due to unknown error
PSR_INVALID_PACKAGE	Parcel to be modified is not found
PSR_INVALID_SHOPID	Invalid shop ID
PSR_INVALID_PARTNER_SHOP_COUNTRY	The partner can not send a parcel to the given shop, because the destination country is not set for the partner
PSR_PACKAGE_BARCODE_MISSING	Parcel barcode is missing
PSR_PACKAGE_BARCODE_INVALID	Parcel barcode is invalid
PSR_SHIPMENTITEM_CANNOTMODIFY_ALREADY_PICKEDUP	Shipment has already been picked up, it cannot be modified anymore
PSR_PACKAGE_INVALID_CURRENCY	Invalid currency
PSR_INVALID_TRANSIT_TIME	Invalid delivery time
PSR_INVALID_PAYMENT_METHOD	Invalid payment method
PSR_INVALID_SUPPLIMENT_DATA	Suppliment data are invalid
PSR_INVALID_SUPPLIMENT_DATA_DELIVERY_PRICE	Suppliment data are invalid, delivery price
PSR_SHIPMENT_CUSTOMERNAME_MISSING	Customer name is missing
PSR_SHIPMENT_CUSTOMERPHONE_MISSING	Customer phone number is missing
PSR_SHIPMENT_CUSTOMERPHONE_INVALID	Invalid customer phone, incorrect country code or format, only +36 and +40 country code is accepted Regex rule: ^((0036|[+]36|36|06)[0-9]{8,9}|(0040|[+]40|40|07)7[0-9]{8})$
PSR_SHIPMENT_CUSTOMERADDRESS_MISSING	Customer address is missing
PSR_INVALID_POSTALCODE	The given postal code is not assigned to route number
PSR_FAILED_POSTALCODE_MISMATCH	Error, when the postal code / city does not exist or city does not match with postal code
PSR_FAILED_THIRD_PERSON_DENIED	Registering third-party parcel is not permitted for the partner
PSR_FAILED_THIRD_PERSON_MISSING_DATA	Missing data at third-party parcel registration, all data must be given
PSR_FAILED_THIRD_PERSON_INVALID_POSTALCODE	The pickup location has invalid postal code at third-party parcel registration
PSR_FAILED_THIRD_PERSON_POSTALCODE_MISMATCH	City does not match with postal code at third-party parcel registration

PSR_INVALID_DELIVERY_PARTNER	Missing or invalid delivery partner
PSR_EXTERNAL_SYSTEM_NOT_AVAILABLE	System is not available, please try it later
PSR_EXTERNAL_SYSTEM_UNKNOW_ERROR	Unknown error occurred during the call, further information can be found in InternationalErrorMessage field.
PSR_EXTERNAL_SYSTEM_NOT_EXCEPTED_RESULT	Unexpected result returned during the call, please contact with our Customer Service
PSR_SHIPMENT_CUSTOMERCOUNTRY_MISSING	Customer country is missing
PSR_SHIPMENT_CUSTOMERPOSTALCODE_OR_CUSTOMERREGION_REQUIRED	Customer postal code or region is required
PSR_INVALID_SERVICETYPE_DIRECT_NOT_POS2POS	Pickup and destination location is the same at Pos2Pos ServiceType, please resend it as a Direct parcel!
PSR_INVALID_PARTNER_TOKEN	Invalid token. Non-filled, or filled incorrectly.
PSR_SHOP_NOT_ENABLED_RETURN_OF_DOCUMENT	The choosen POS is not permitted for document return handling. The shop is not allowed to hand over such parcel.
PSR_PARTNER_NOT_ENABLED_RETURN_OF_DOCUMENT	Our partner is not permitted for using document return service. Concerned partner is not allowed to register such parcel.
PSR_RETURN_OF_DOCUMENT_NEEDED_REFERENCE_BARCODE	RefereneBarcode was not given for a parcel with document return service.
PSR_RETURN_OF_DOCUMENT_MISSING_ORIGINAL_PACKAGE	Parcel with return of document option cannot be registered, since the reference barcode was not registered / does not exist in the system.
PSR_TOO_MANY_PARCEL_IN_REQUEST	You have tried to register too many parcels, the maximum value is 1000.
PSR_PACKAGE_AUTHCODE_MISSING	Authorization code is not given.
PSR_PACKAGE_USED_BARCODE	The parcel barcode is or were already used. New barcode is necessary.
PSR_INVALID_PRICE_CANNOT_BE_NEGATIVE	The price cannot be negative number.
PSR_PACKAGE_INVALID_PRICE	Either the parcel value or the COD amount is invalid. Maximal values: Parcel value: 13 000 000 HUF COD: 700 000 HUF
PSR_OK_PACKAGE_LABEL_PRINTING	Succesful data transfer for label printing (with OnlyLabelPrinting parameter) In this case, the package data isn't handled, the system only stores data for the label printing.
PSR_PACKAGE_LABEL_PRINTING_NEEDED	They tried to finalize label printing (FinalizeLabelPrinting) on a package, of which we didn't recieve data for. (OnlyLabelPrinting)

4. Parcel label creation

4.1. About the label creation in general

This service call returns a document, only if we got back a **PSR_OK** response at the parcel registration, so the parcel dispatch was successful and the barcode exists in PUDO system, otherwise the document creation will return with error message. For example we will receive error messages **WAR_PARCEL_NOT_EXISTS** or **WAR_INVALID_BARCODE**, if the barcode do not exist or is not registered in PUDO. If we sent a valid barcode, the returned object contains the RES_OK response and the name of the document appears in the DocumentName. The document name is generated from the document settings.

4.2. Printing label without package registration

In a normal case, printing a label (GetDocument service) requires that the package's registration has been finalized (RegisterParcelContainer), in cases like this, the package shows up in the logistical system, and it's data isn't modificable anymore.

An other option would be that if we only need to print the label (since the packages are under preparation, and the shipment will only be ready by evening), that they mark the package data during the calling of RegisterParcelContainer with the **OnlyLabelPrinting**=true setting. If the data transfer is successful, then we give back the PSR_OK_PACKAGE_LABEL_PRINTING answer. In this case, we register it's data, but it doesn't transfer in the logistical system yet, however with the GetDocument call we can print label for them. Until finalization, the parcel's data are modificable. When they finish preparing the packages, then we have to call the RegisterParcelContainer again, with the **FinalizeLabelPrinting**=true parameter, and with those barcodes from the Parcel list which they plan to send. With this step, the data transfer to the logistical system, and they are no longer modificable.

Process steps:

1. Asking for the label
    a) RegisterParcelContainer call with **OnlyLabelPrinting**=true parameter
    b) GetDocument call to ask for the label
2. Finalizing the data transfer
    a) RegisterParcelContainer call with **FinalizeLabelPrinting**=true parameter

4.3. Document types

4.3.1.Label/waybill

This document type returns the labels generated based on the parcel barcodes. The Size attribute defines how many labels will be printed on 1 page. If the Size attribute is empty, the default size (2x2) will be used.

DocumentName: **DT_PackageLabel**

4.3.1.1 Size 2x2 – A4 paper split into 4 segments

This document is generated after setting **Size = DS_2x2**, which is the default value for Size attribute. The mentioned setting will split an A4 paper into 4 segments, generate and place the labels segment by segment.

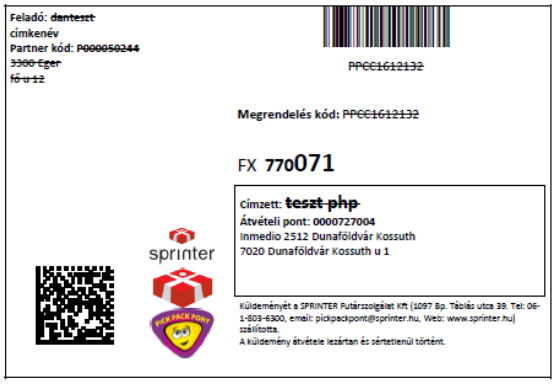
DocumentName: **DT_PackageLabel_DS_2x2**

You can see 2 labels placed in P_0 and P_1 positions on the picture, which were generated for home-delivery parcels.

The Service will decide automatically which label type (home delivery or PickPackPont) has to be printed, that is not an adjustable parameter.

The following data have to be placed on the generated labels:

- Sender's data (name, postal code, city, street, phone number, partner code)
- Comment for the courier (If there was not any comment, then the field is empty)
- Parcel data (barcode, quantity, weight, value of goods)
- Delivery time
- Order and invoice number
- Addressee's data (name, postal code, city, street, phone number)
- Amount to be collected at registration and delivery
- COD amount
- 2D Data Matrix code
- Sprinter and PPP logo



- Delivery type [HD or FX], route number

4.3.1.2. A/6 size (For thermal- and tape-printers)

This document is available after the setting **Size=DS_A6.** This document is identical with the 2x2, with the only exception being that it's generated in A6 size, so only 1 label gets printed by page. This format is suitable for thermal- and tape-printers.

For succesful printing, the following settings must be done in the printing settings:

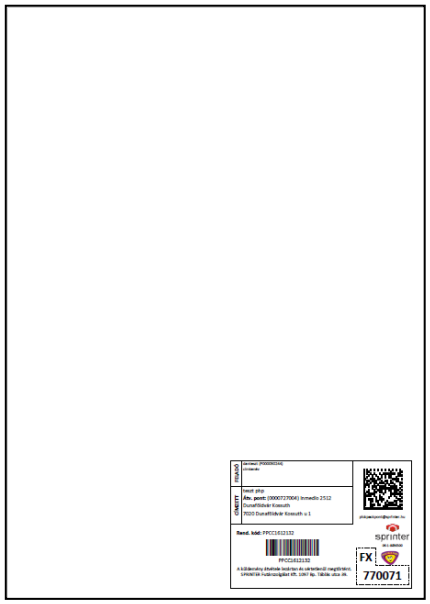
It has to be printed in A6 size, landscape layout, and the print image has to be checked.

DocumentName: **DT PackageLabel DS A6**

4.2.1.3. Size 2x4 – A4 paper split into 8 segments

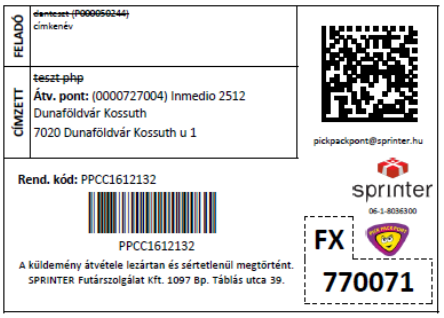
This document is generated after setting **Size = DS_2x4**. The mentioned setting will split an A4 paper into 8 segments, generate and place the labels segment by segment.

DocumentName: **DT_PackageLabel_DS_2x4**



You can see a label placed in P_7 position on the picture.

The same data have to be printed on the labels, as on the 2x2 version. This setting differs in page layout and label size from the 2x2 version, and only the sender's name, partner code are visible at the sender's data.



4.3.1.4. International FANCOURIER label

This document is only created if the parcel is sent to Romania; it is generated by FanCourier courier service.

A parcel ID or waybill ID known by FanCourier system is needed to generate this document.

DocumentName: **International_FANCOURIER_<<AWB_Number>>**

4.3.2 Delivery Note

The parcels to be handed over are listed in this document.

DocumentName: **DT_DeliveryNote**

sprinter **Csomagátvételi elismervény - Fix pontos szállítás**

Dátum 2016. december 14.

	Átadó		Átvevő
	P000050244	Átvevő neve:	Sprinter Futárszolgálat Kft.
Átadó neve:	danteszt	Átvevő címe:	Táblás utca 39
Átadó címe:	fő u 12	Település:	1097 Budapest

Csomag vonalkód	Címzett	Fuvardíj összeg	Utánvét összeg	Szállítás típusa
PPCC1612132	teszt php			Fix pontos szállítás
Összesen				**1 db Fix pontos szállítás**

Átadó + bélyegző

Átvevő

The document layout is the same at both delivery model (home delivery or fixed drop-off point shipping). The only difference is that the transport fee and COD amount is not presented on the fix drop-off point document.

4.3.2.1. Fixed drop-off point shipping – PPP

This type of delivery note is created in case of fix drop-off point shipping. The document is printed on one page on size A4.

The following data are placed on the document:

- **Consignor's data** (Partner's data)
- **Recipient's data**
- **Parcel barcode(s)**
- **Addressee(s)**
- **Shipment type(s)**
- Date of creation

Furthermore, the document has to be signed by both parties (consignor and recipient) and stamped by the recipient.

DocumentName: **DT_DeliveryNote_PPP**

4.3.2.2. Home delivery shipping – HD

This type of delivery note is created in case of home delivery shipping. The document is printed on one page on size A4.

The following data are placed on the document:

- Partner's data
  - **Partner code**
  - **Partner's name**
  - **Partner's address**
- Recipient's data
  - **Recipient's name**
  - **Recipient's address**
  - **City**
- **Parcel barcode(s)**
- **Addressee(s)**
- **Transport fee(s)** (optionally)
- **COD amount(s)** (depends on the currency)
- **Shipment type(s)**
- Summary of **transport fees**, **COD amounts** and **shipment types**

- Aggregated amount of transport fees to be paid on the spot
- Date of creation

Furthermore, the document has to be signed by both parties (partner and recipient) and stamped by the recipient.

DocumentName: **DT_DeliveryNote_Courier**

4.3.3. Proof of receipt

This document contains the proof of receipt regarding cash handover for both parties (courier and consignor). Two copies of the receipt are printed on an A4 size paper; there is a separator line in the middle of the page to mark off the documents from each other. 1 copy of the receipt is created for the courier, the other one for the consignor.

The following data are placed on the document:

- Order code
- Amount of cash received
- City, date

Furthermore, the identity card numbers, the consignor's and recipient's name (with print written and hand-written form) have to be written on the document.

DocumentName: **DT_ReceiptReport**

SPRINTER Futárszolgálat Kft.
H-1097 Budapest, Táblás utca 39.
SPRINTER IT Support
E-mail: pudointerface@lapker.hu

Megrendelés kód PPCC16120716 **Sprinter példánya**

SPRINTER CSOMAG ÉS LOGISZTIKA KÉSZPÉNZ ÁTVÉTELI ELISMERVÉNY

Alulírott ..... Szem.ig.sz.: ..... ezúton elismerem, hogy a mai napon a danteszt –től átvettem 8788 Ft -ot a szállítási díjra*. A számla két hét múlva postai úton érkezik a Feladó nevére és címére kiállítva.

Átvevő aláírása	Átadó aláírása
Átvevő aláírása nyomtatottan	Átadó aláírása nyomtatottan

Eger, 2016. december 14.

*A szállítási díj csak a házhoz szállításos szolgáltatási díjat foglalja magában. Ha fix pontra történt a megrendelés, akkor annak a házhoz szállítási díját nem fedezi.

Megrendelés kód PPCC16120716 **Feladó példánya**

SPRINTER CSOMAG ÉS LOGISZTIKA KÉSZPÉNZ ÁTVÉTELI ELISMERVÉNY

Alulírott ..... Szem.ig.sz.: ..... ezúton elismerem, hogy a mai napon a danteszt –től átvettem 8788 Ft -ot a szállítási díjra*. A számla két hét múlva postai úton érkezik a Feladó nevére és címére kiállítva.

Átvevő aláírása	Átadó aláírása
Átvevő aláírása nyomtatottan	Átadó aláírása nyomtatottan

Eger, 2016. december 14.

*A szállítási díj csak a házhoz szállításos szolgáltatási díjat foglalja magában. Ha fix pontra történt a megrendelés, akkor annak a házhoz szállítási díját nem fedezi.

4.4. Presentation of types and fields

4.4.1.Functions

GetDocument

Return object type `return MassDocumentResponse`
Parameters: `MassDocumentRequest request`
Mandatory field: **yes**

Type: Function
Description:

This is a normal report generating method, which expect a **MassDocumentRequest** as value and returns a **MassDocumentResponse** type data.

4.4.2. Types

4.4.2.1. Request

MassDocumentRequest

Type: Category
Description:
This is a normal report generating method, which expect a **MassDocumentRequest** as value and returns a **MassDocumentResponse** type data.
Fields:

Type	Field Name	Remark	Mandatory
List<BarcodeData>	**Barcodes**	Barcode list	**Yes**
List<DocumentSetting>	**DocumentSettings**	Document Settings	**Yes**
string	**SuplimentJSONData**		No

BarcodeData

Type: Category
Description:

This is defining the barcode and the type of the barcode.

Fields:

Type	Field Name	Remark	Mandatory
string	**Barcode**	Barcode	**Yes**
BarcodeType	**Type**	Barcode type	**Yes**

string	**SuplimentJSONData**		**No**

BarcodeType

Type: Enumerator

Description:

This is an enum to define the type of the Barcode.

Fields:

Name	Remark
BT_PackageBarcode	Parcel Barcode
BT_ShipmentBarcode	Shipment Barcode

DocumentSetting

Type: Category

Description:

The necessary documents can be parameterized.

Fields:

Type	Field name	Remark	Mandatory
DocumentFormat	**Format**	can be DF_PDF or DF_ZPL	**yes**
boolean	**IsPositioned**	true = label false = A4 sized page	no
LabelDocumentPosition	**Position**	From P_0 To P_7	no
LabelDocumentSize	**Size**	2x2 or 2x4	no
string	**SupplimentJSONData**		no
DocumentType	**Type**	Label/Delivery note/Proof of receipt or each of them	no

DocumentFormat

Type: Enumerator

Description:

This is the type is to describe the format of document given back as response.

Fields:

Name	Remark
DF_PDF	Sign of the PDF format
DF_ZPL	Sign of the ZPL format

LabelDocumentPosition

Type: Enumerator

Description:

In case of label printing process the starting position of the first label should be specified. Order of the positions must be handled from up to down and from left to right.

Fields:

Name	Remark	
	2x2	2x4
P_0	1	1
P_1	2	2
P_2	3	3
P_3	4	4
P_4	This position not defined	5
P_5		6
P_6		7
P_7		8

LabelDocumentSize

Type: Enumerator

Description

This is the size of the label based on the fact that how many label can be visible on A4 size of paper. 2X2 = 4 quantities 2X4 = 8 quantities

Fields:

Name	Remark
DS_2x2	At label printing the A4 size paper is split to 2x2 segments
DS_2x4	At label printing the A4 size paper is split to 2x4 segments

DocumentType

Type: Enumerator

Description:

This is the list of document types.

Fields:

Name	Remark
DT_All	Sign all types, except if DocumentFormat is DF_ZPL, new request will have to be made with DF_PDF to get all types
DT_PackageLabel	Package label, DF_ZPL only returns this type of document
DT_DeliveryNote	Delivery Note
DT_ReceiptReport	Receipt Report

4.4.2.2. Response

MassDocumentResponse

Type: Category

Description:

This is a response message from the server that will be received back from GetDocument() method.

Fields:

Type	Field Name	Remark
List<DocumentGenerationResult>	**DocumentGenerationResults**	Document printed Status based on the barcodes.
List<DocumentData>	**Documents**	Document belongs to the barcode.
DocumentResults	**Result**	Result response of the document generation success
string	**SuplimentJSONData**	

DocumentResults

Type: Enumerator

Description:

This field provides information about the fact that document has been generated successfully or not. For example the requested barcode does not exist.

Fields:

Name	Remark
RES_OK	Document has been generated successfully.
RES_DATA_MISSING	Document has not been generated due to missing data.
RES_GENERATE_ERROR	Document generation error

DocumentGenerationResult

Type Category

Description:

Related to the given barcode this field provides information about the fact that document has been generated successfully or not. In case of failure the Results filed can be informative.

Fields:

Type	Field name	Remark
string	DocumentName	Name of the generated document based on value of DocumentSettings

		field.
ResultCode	**Result**	Result information of the generated document
string	ResultMessage	
string	SuplimentJSONData	

ResultCode

Type: Enumerator

Description:

This field provides further information about the generated document. It has been generated successfully or some kind of problem occurred. For example the required barcode does not exist or invalid barcode has been used.

Fields:

Name	Remark:
RES_OK	Document has been generated successfully.
ERR_NOT_IMPLEMENTED_DOCUMENT	Not implemented document
WAR_INVALID_BARCODE	invalid barcode
WAR_PARCEL_NOT_EXISTS	parcel not exists
WAR_PARCEL_NOT_PRESENTED_IN_ANY_REPORT	parcel not presented in any report
ERR_DOCUMENT_GENERATION_FAILED	document generation failed
WAR_SHIPMENT_NOT_EXISTS	shipment not exists

DocumentData

Type: Enumerator

Description:

In case, if the value of the DocumentGenerationResult.Result is **RES_OK**, then this contains the document itself in file named Document, or the ZPL code in the string named DocumentString.

Fields:

Type	Field name	Remark
base64Binary	**Document**	This byte block should contain the generated document in case of PDF
string	**DocumentString**	This string should contain the generated ZPL code in case of ZPL
string	**DocumentName**	Name of the generated document based on value of DocumentSettings field.

boolean	**IsPositioned**	Label or not
string	**SupplimentJSONData**	
DocumentType	**Type**	Type of the generated document.

4.4.2.3. Error codes

The whole document creation process possible responses can be found at MassDocumentResponse in DocumentResults

The specific document creation responses being listed at DocumentGenerationResult in ResultCode description

5. Source code demo

The code contains the all basic types of placing orders with examples. This should be replaced with current data.

Order creation access path for interface test:
https://tsx.lapker.hu/PudoTest/PartnerPudoService?singlewsdl

Document creation access path interface test:
https://tsx.lapker.hu/PudoTest/PudoDocumentService?singlewsdl

6.3. C# code

.NET source for the Interface creation: can be found in a folder named **csharp.zip** next to the partner interface specification. csharp/ParcelRegistrationPUDOClient.sln

The sln is going to load the code example written below after opening the Visual Studio 2015.

6.3.1. Order creation

Within your VisualStudio project, please add the access path of PUDODocumentService Interface to the references with using Add Service Reference.

A new Service References folder is created in Solution Explorer after adding the service successfully. You can start to implement and use the given category by the Service References.

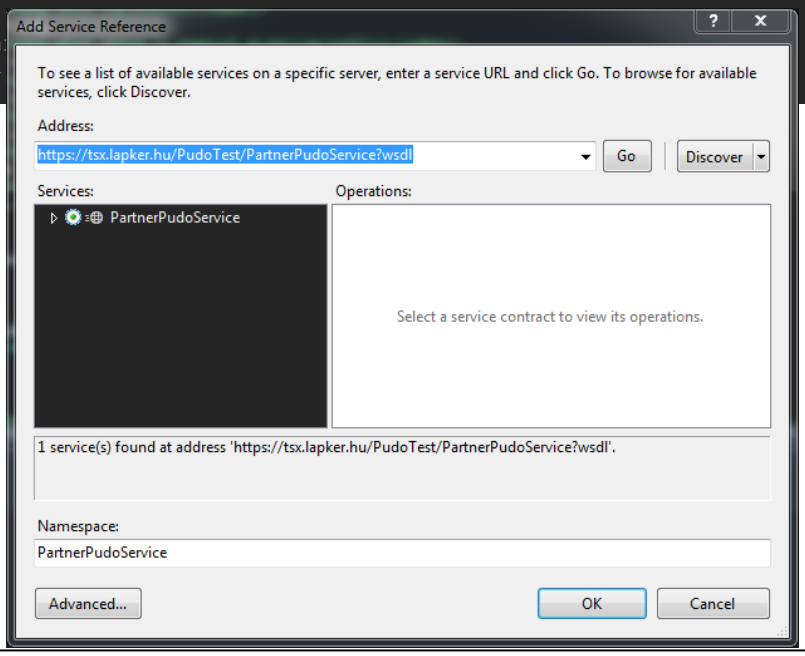
WITHIN THE ORDER CREATION PROCESS THERE ARE SOME MANDATORY FIELDS. THESE CANNOT BE IGNORED, SINCE IF THESE ARE MISSING, THE ORDER CREATION WILL NOT BE SUCCEED. DUE TO THE FACT TO PROVIDE THE PREFIX UNIQUE BARCODE IS MANDATORY, THIS DOCUMENT WILL CONTAIN RELATED EXAMPLES.

UNIQUE IDENTIFIER (PREFIX) IS DIFFERENT FOR EACH PARTNER. THE PREFIX HAS TO BE REQUIRED FROM SPRINTER IT SUPPORT (pudointerface@lapker.hu)!

The source code demo has been generated based on a sample and these will be partly visible in this document. The program's execution logic uses the same logic too. In order to be able to make the explanation of the code more unified, communication between different proceedings, it is not segmented.

First step is to build the parcels wish to be registered up.

```
//berögzíten
List<Parcel>
```



The parameters will be described separately for home delivery and for PPP. Kindly pay attention to these fields, because different registrations are not implemented with different objects.

Due to different cases, there is no possibility to sign all special, but important fields as mandatory field.

Once we have created the required quantity of parcel with correct data we have to build a request and send it to PUDO

```
RegisterParcelContainerRequest _registerParcelContainerRequest = new
RegisterParcelContainerRequest()
{
     PartnerCode = "0000001000",              //Partnerazonosító
     ParcelContainer = _parcelContainer,      //csomagadatok listája
     Supplier = Supplier.Partner,             //Ez a sor kivezetésre fog kerülni, de
                                              jelenleg kötelező a kitöltése
     //ArriveDate = new DateTime(2017,1,1,18,0) //Beszállítás várható ideje, információs
                                              jellegű, opcionális
};

// kliens kapcsolat létrehozása és a korábban beállított csomagok elküldése/berögzítése
PartnerPudoClient _partnerPudoClient = new PartnerPudoClient();
// csomag rögzítés sikerességéről válasz üzenet
RegisterParcelContainerResponse _registerParcelContainerResponse =
          _partnerPudoClient.RegisterParcelContainer(_registerParcelContainerRequest);
```

In the **RegisterParcelContainerResponse** response message we receive the notification about the successful order creation.

In **registerParcelContainerResponse.ErrorCode** response will receive the parcel success notification. **registerParcelContainerResponse.ParcelResults.ErrorCode** informs us which was not successful or which was failed.

6.3.1.1. Error handling and data processing

This chapter will describe how can identify the given ErrorCode field value

If the value **PSR_OK** , then the order creation was successful or partially successful, but the shipment has been registered.
If the error message not **PSR_OK,** the order creation was failed. For the sake of an example parcel data have not been transferred, then the following message will be visible in the **ErrorCode** PSR_EMPTY_SHIPMENT

The possible error messages have been listed and described in chapter Error codes.
The following chapter describes steps in case of order creation success.

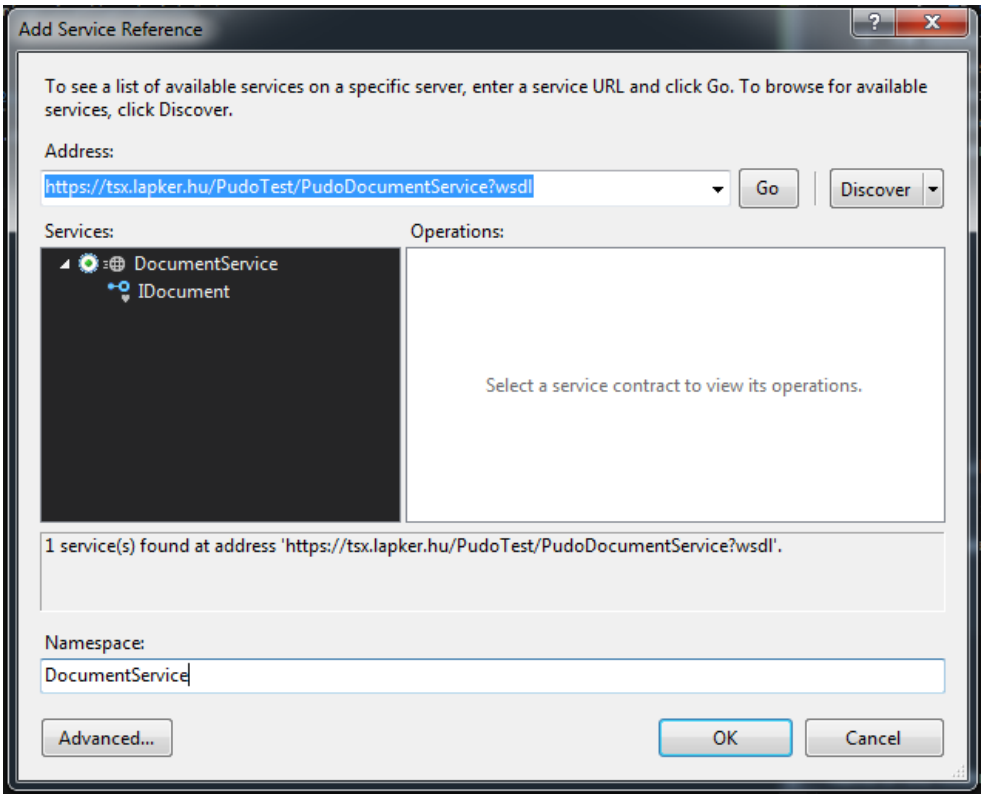
```
// Amennyiben a Response ErrorCode értéke "PSR_OK", úgy van értelme tovább
vizsgálni a Response-t
if(_registerParcelContainerResponse.ErrorCode == "PSR_OK")
{
    //Ebbe a listába tudjuk kigyűjteni a sikeresen berögzített csomagok
    vonalkódját, így később ezekre a vonalkódokra tudjuk lekérdezni a
    csomagcímkéket.
    List<BarcodeData> _barcodes = new List<BarcodeData>();
    // Haladjunk végig a ParceResults mezőn, ebben találjuk meg csomagonkénti
    lebontásban a csomagregisztrációhoz tartozó válasz adatokat.
    foreach(ParcelResult results in
    _registerParcelContainerResponse.ParcelResults)
    {
        Console.WriteLine($"OriginalBarcode : {results.OriginalBarCode}");
        Console.WriteLine($"NewBarcode : {results.NewBarCode}");
        Console.WriteLine($"ErrodCode : {results.ErrorCode}");
        Console.WriteLine($"ShipmentID : {results.ShipmentID}");
        //Nemzetközi csomagregisztráció esetén
        if (!string.IsNullOrWhiteSpace(result.ErrorMessageInternational))
        {
            Console.WriteLine($"ErrorMessageInternational
                :{result.ErrorMessageInternational}");
        }
        Console.WriteLine();
        //Ha PSR_OK az adott csomag ErrorCode értéke, abban az esetben
        jegyezzük meg a vonalkódot.
        if (result.ErrorCode == "PSR_OK")
        {
            _barcodes.Add(new BarcodeData()
            {
                Barcode = result.NewBarCode,
                Type = BarcodeType.BT_PackageBarcode
            });
        }
    }
}
```

This was a complete parcel registration and from **Response** we can get more information about its success.

During sample code creation we paid attention to interwork parcel registration-related and barcode-based document generator-related code parts.

6.3.2. Label creation

Within your VisualStudio project please add the access path of PUDODocumentService Interface to the references with using Add Service Reference.



A new Service References folder is created in Solution Explorer after adding the service successfully. Next step is to create a list of the barcodes.

```
//vonalkódokat tartalmazó lista
List<BarcodeData> _barcodes = new List<BarcodeData>();
_barcodes.Add(new BarcodeData()
{
    Barcode = "23476001200021001F1200",       //kötelező megadni a vonalkódot
    Type = BarcodeType.BT_PackageBarcode      //Kötelező beállítani a Type-ot
});

                              //Bármelyik két mező hiánya, exception-t okozhat
_barcodes.Add(new BarcodeData()
{
    Barcode = "PPCC1612131",
    Type = BarcodeType.BT_PackageBarcode
});
```

To fill the Barcode and Type field is a must.
Settings related to the documents that would like to print out needs to be added.

```
List<DocumentSetting> _documentSettings = new List<DocumentSetting>();
_documentSettings.Add(new DocumentSetting()
{
    Format = DocumentFormat.DF_PDF,
    IsPositioned = false,
    Position = LabelDocumentPosition.P_0,
    Size = LabelDocumentSize.DS_2x2,
    Type = DocumentType.DT_All
});

_documentSettings.Add(new DocumentSetting()
{
    Format = DocumentFormat.DF_PDF,
    //IsPositioned = false,
    //Position = LabelDocumentPosition.P_0,
    //Size = LabelDocumentSize.DS_2x2,
    //Type = DocumentType.DT_All
});
```

AT LEAST 1 DOCUMENTTSETTING() NEEDS TO BE ADDED TO THE DOCUMENTSETTINGS LIST.

Identify the differences and also the parities between two document settings.

`To add = DocumentFormat.DF_PDF is a must`, otherwise we receive ***exception***.

To add value for the other fields are not mandatory since the Service handle them as **default** automatically.

Size	=	2x2 or on page A4 4quanties label can be printed
Position	=	0 is start position, from top left corner
Type	=	each document type (labels, delivery notes and proof of receipts)

```
//adjuk meg a request-nek a korábban beállított adatokat (vonalkód, dokumentum
beállítások)
MassDocumentRequest _massDocumentRequest = new MassDocumentRequest();
_massDocumentRequest.Barcodes = _barcodes;
_massDocumentRequest.DocumentSettings = _documentSettings;

//hozzuk létre a klienst
DocumentClient _documentClient = new DocumentClient();
//Majd a GetDocument fv-nek adjuk át a request-et
MassDocumentResponse _massDocumentResponse =
              _documentClient.GetDocument(_request);
```

For **GetDocument**() method we receive response with document and Result code for the related barcodes.

6.3.2.1. Error handling and data processing

In case of `_response.Result == DocumentResults.RES_OK` the service call was successful, so the data in `_response.Documents` object can be processed.

```
switch (_massDocumentResponse.Result)
{
    //Amennyiben a Result RES_OK, a dokumentumgenerálás sikeresen lefutott
    // és kaptunk vissza nyomtatható dokumentumot
    case DocumentResults.RES_OK:
        {
            //a response sikeres volt, fel lehet dolgozni a legenerált dokumentumokat.
            //Példa kód lentebb
            …
            break;
        }
    case DocumentResults.RES_GENERATE_ERROR:
        {
            Console.Write("Hiányzó adatok miatt nem lehet előállítani
                dokumentumokat");
            break;
        }
    case DocumentResults.RES_DATA_VALIDATION_ERROR:
        {
            Console.Write("Nem sikerült a beküldött adatok validálása");
            break;
        }
    case DocumentResults.RES_DATA_MISSING:
        {
            Console.Write("Dokumentum előállítás közben hiba történt");
            break;
        }
    default:
        {
            Console.Write(_response.Result);
            break;
        }
}
```

In case of **RES_OK** the documents can be processed as follows:

```
//Válasz üzenetek a dokumentumokról
foreach (DocumentGenerationResult _documentGenerationResult in
_maddDocumentResponse.DocumentGenerationResults)
{
    Console.WriteLine(_documentGenerationResult.DocumentName);
    Console.WriteLine(_documentGenerationResult.Result);
    if (!String.IsNullOrWhiteSpace(_documentGenerationResult.ResultMessage))
    {
        Console.WriteLine(_documentGenerationResult.ResultMessage);
    }
    Console.WriteLine();
}
int _index=0;  //Ez a számláló segít abban, hogy az azonos nevű dokumentumok ne írják egymást felül
FileStream lemezre írásakor
//Egyszerűsített dokumentum lemezre írása
foreach (DocumentData _document in _response.Documents)
{
    using (FileStream savefile = new
    FileStream($"C:/{_document.DocumentName}{_index++}.pdf",
    FileMode.OpenOrCreate))
    {
            savefile.Write(_document.Document, 0, _document.Document.Count());
            savefile.Flush();
    }
}
```

The couple of rows above can be useful for processing the returned data.

Based on the instruction above the C# source code can be prepared, which gets back the printable labels, delivery notes and proof of receipts in case of **Valid** barcodes.

6.4. PHP code

The sample contains the previously placed orders and related labels based on the barcodes.

6.4.1. Order creation

PHP source for the interface: can be found in a folder named **php.zip** next to the PUDO partner interface specification. php/**PartnerPudoService.php**

Test code for the interface:
php/**PudoService.php**

Download the php category description from the above mentioned link and add it to the php source code, then prepare the header data needed for a successful Service call. (Describes SSL encryption)

```
include("PartnerPudoService.php");
//Ez az opció  a https-en keresztüli TLS kapcsolat titkosítási módját állítja be. Ennek hiányában nem lehet
behívni a serverre.
$soap_options = array(
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
        'stream_context'=>
                stream_context_create(
                        array(    'ssl' => array(       'protocol_version' => 'tls1',
                                                        'ciphers' => 'DES-CBC3-SHA:RC4-SHA:RC4-MD5'
                                                )
                        )
                ),
        'cache_wsdl'=> WSDL_CACHE_NONE
);
$wsdl = " https://tsx.lapker.hu/PudoTest/PartnerPudoService?wsdl ";
```

At this point please create the **request** and fill it with appropriate data.

First step: Prefix needs to be entered then the list of the parcels should be defined.

```
//Egyedi vonalkód Prefix, amit kaptunk
$BarCodePrefix  ='ASD'.date('YmdHis');          // => ASD20170101010101
//Csomagok gyűjtő listája
$ParcelContainer = array();
```

After the parcels list creation with correct details a Request need to be prepared and sent to PUDO

```
//    Hozzuk létre a Request
 $RegisterParcelContainerRequest = new RegisterParcelContainerRequest();
 $RegisterParcelContainerRequest->PartnerCode ="0000001000";
 $RegisterParcelContainerRequest->ParcelContainer = $ParcelContainer;
//Itt adhatjuk meg a tervezett csomagfelvételének dátumát
 //$RegisterParcelContainerRequest->ArrivedDate = date("Y.m.d H:i:s");  //DEFAULT NULL

//    Hozzuk létre a Klienst
$PartnerPudoServiceClient  = new  PartnerPudoService($wsdl, $soap_options);

//    Küldjük el a Request-et és a Respons választ ezután tudjuk feldolgozni
$RegisterParcelContainerResponse  =
        $PartnerPudoServiceClient->RegisterParcelContainer($RegisterParcelContainerRequest);
```

In RegisterParcelContainerResponse response will receive the parcel creation success notification: **$RegisterParcelContainerResponse->RegisterParcelContainerResult->ErrorCode** field shows it parcel by parcel **$RegisterParcelContainerResponse->RegisterParcelContainerResult->ParcelResults->ErrorCode** and give us further information about which is succeed or which is failed.

6.4.1.1. Error handling and data processing

This chapter will describe how can identify the given ErrorCode field value.

If the value **PSR_OK** , then the order creation was successful or partially successful, then the shipment has been registered.
If the error message **not PSR_OK,** the order creation was failed. In case if the parcel data have not been transferred, then in the **ErrorCode** the following message will be visible `PSR_EMPTY_SHIPMENT`

The optional error messages have been listed and described in chapter Error codes.
The following chapter describes steps in case of order creation
success

```
//Szállítmány szinten történő hibakódok feldolgozása
switch ($RegisterParcelContainerResponse->RegisterParcelContainerResult->ErrorCode) {
        case 'PSR_OK':{
                //Sikeres csomagregisztráció esetén ebben az ágban lehet feldolgozni a válasz üzenetben kapott
                adatokat.
                //Érdemes a csomag címkékhez tartozó adatokat itt feldolgozni illetve ebben a részben
                érdemesebb a csomag címke lekérést is elvégezni, amit szintén a későbbiekben be is mutatunk.
                break;
        }
        default:{
                echo $RegisterParcelContainerResponse->RegisterParcelContainerResult->ErrorCode;
                break;
        }
}
```

In case of **PSR_OK** ; the parcel level data can be proceed in the below mentioned way:

```
//Sikeresen berögzített csomag vonalkódok kigyűjtéséhez szükséges tömb
$Barcodes = array();

//Nézzük végig csomagszinten a válaszüzenetben szereplő adatokat
 $ParcelResults = $RegisterParcelContainerResponse->RegisterParcelContainerResult->ParcelResults;
foreach ($ParcelResults->ParcelResult as $Parcel) {
        echo ("Original Barcode : $Parcel->OriginalBarCode <br>");
        echo ("New Barcode : $Parcel->NewBarCode <br>");
        echo ("ErrodCode : $Parcel->ErrorCode <br>");
        echo ("ShipmentID : $Parcel->ShipmentID <br>");
        //Amennyiben nemzetközi csomagot regisztráltunk, az alábbi változóban lesz érték.
        if (!is_null($Parcel->ErrorMessageInternational) || $Parcel->ErrorMessageInternational!=""){
                echo ("ErrorMessageInternational : $Parcel->ErrorMessageInternational <br>");
        }
        //Ha csomaghoz tartozó ErrorCode PSR_OK, úgy sikeres volt a csomag regisztráció. Ennek a
        vonalkódját érdemes kigyűjteni a későbbi csomag címkelekérdezéséhez szükségünk lehet rá
        if($Parcel->ErrorCode=='PSR_OK'){
                $Barcodes[] = new BarcodeData($Parcel->NewBarCode);
        }
}
```

This was a complete parcel registration and data from Response shows that it was successful.

6.4.2. Label creation

PHP source for the interface:
php/**PudoDocumentService.php**

Test code for the Interface:
php/**PudoService.php**

Download the php category description from the above mentioned link and add it to the php source code, then prepare the header data needed for a successful Service call. (Describes SSL encryption)

```
include("PudoDocumentService.php");
//Ez az opció, a https-en keresztüli TLS kapcsolat titkosítási módját állítja be. Ennek hiányában nem lehet
behívni a serverre.
$soap_options = array(
        'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
        'stream_context'=>
                stream_context_create(
                        array(  'ssl' => array(
                                        'protocol_version' => 'tls1',
                                        'ciphers' => 'DES-CBC3-SHA:RC4-SHA:RC4-MD5'
                                )
                        )
                ),
        'cache_wsdl'=> WSDL_CACHE_NONE
);
$wsdl = " https://tsx.lapker.hu/PudoTest/PudoDocumentService?wsdl ";
```

At this point please create the request and fill it with appropriate data.

Data required:

- Barcode list

```
// vonalkód lista létrehozása, Példányosításkor kötelező vonalkódot megadni!
$Barcodes[] = new BarcodeData("PPCC1612131");        //PPP
$Barcodes[] = new BarcodeData("PPCC1612132");        //PPP
```

It is an array, in which all elements are filled with the instance of the BarcodeData class. At the beginning the barcode has to be defined from the constructor, this is a mandatory parameter, because the php interpreter stops running without that setting.

In contrast with C# code, the php constructor sets the Type field automatically.

Please note, that in case of damaged or changed constructor in PudoDocumentService.php file you need to set **$this->Type = "BT_PackageBarcode";** row in the constructor.

- List of document settings

```
//documentSettings
$documentSetting = new DocumentSetting();
$documentSetting->IsPositioned = false;                          // Etikett? true | false
$documentSetting->Position     = LabelDocumentPosition::$P_0;    // pozíció 0-7 2x4 esetén  0-3 2x2 esetén
$documentSetting->Size         = LabelDocumentSize::$DS_2x2;     // címke méret, A4-es lap felosztása
$documentSetting->Type         = DocumentType::$DT_All;          // generálandó dokumentum típusok
                                 //          DT_All
                                 //          DT_PackageLabel  => csomag címke
                                 //          DT_DeliveryNote     => csomag átvételi elismervény
                                 //          DT_ReceiptReport => kézpénz átvételi elismervény

 //az elkészült dokumentum beállításokat fűzzük hozzá a $DocumentSettings listához
$DocumentSettings[] = $documentSetting;
```

Setting the fields is not mandatory, in case of sending empty field or **null** value the Service call is executed with **default** values, so error message will not return.

Defining `Format = DocumentFormat.DF_PDF` is mandatory, but the PudoDocumentService.php constructor executes that setting automatically.

```
$MassDocumentRequest = new MassDocumentRequest();
//kikell tölteni a Request-et
//vonalkódokat tartalmazó tömb
$MassDocumentRequest ->Barcodes = $Barcodes;
//dokumentum beállításokat tartalmazó tömb
$MassDocumentRequest ->DocumentSettings = $DocumentSettings;

$DocumentClient = new DocumentService($wsdl, $soap_options);
$MassDocumentResponse = $DocumentClient ->GetDocument($MassDocumentRequest);
```

The **GetDocument**() method of the created instance can be called; the response will contain the documents related to the barcodes (**Documents**), the created messages during document creation process (**DocumentGenerationResults**) and the **Result** codes.

The **Result** code is useful at error handling, if the process is not performed due to any reason, then do not try to reach the Document object, because it is not exist at all.

6.4.2.1. Error handling and data processing

Based on the above mentioned Result code easy to define, whether we get processable data back or not.

```
switch($MassDocumentResponse ->GetDocumentResult->Result){
        case "RES_OK": {
                //a response sikeres volt, fel lehet dolgozni a legenerált dokumentumokat.
                //Példa kód lentebb
                ...
                break;
        }
        case "RES_DATA_MISSING": {
                echo "Hiányzó adatok miatt nem lehet előállítani dokumentumokat";
                break;
        }
        case "RES_GENERATE_ERROR": {
                echo "Dokumentum előállítás közben hiba történt";
                break;
        }
        default : {
                echo $MassDocumentResponse ->GetDocumentResult->Result;
                break;
        }
}
```

In case of **RES_OK** the documents can be processed as follows:

```
//Válasz üzenetek a dokumentumokról
foreach($MassDocumentResponse ->GetDocumentResult->DocumentGenerationResults-
>DocumentGenerationResult as $DocumentGenerationResult) {
        print("DocumentName: " . $DocumentGenerationResult->DocumentName . "<br>");
        print("Result: " . $DocumentGenerationResult->Result . "<br>");
        print("ResultMessage: " . $DocumentGenerationResult->ResultMessage . "<br>");
        print("SupplimentJSONData: " . $DocumentGenerationResult->SupplimentJSONData . "<br>");
        print("<br>");
}
//Egyszerűsített dokumentum lemezre írás
foreach($MassDocumentResponse ->GetDocumentResult->Documents->DocumentData as $docResponse){
        $PrintDoc = fopen($docResponse->DocumentName.".pdf","w");
        fwrite($PrintDoc, $docResponse->Document);
        fclose($PrintDoc);
        echo $docResponse->DocumentName.".pdf fájl írása megtörtént<br>";
}
```

The couple of rows above can be useful for processing the returned data.

Based on the instruction above the php source code can be prepared, which gets back the printable labels, delivery notes and proof of receipts in case of **Valid** barcodes.

IF **$MassDocumentResponse ->GetDocumentResponse** is not created, then there can be a problem with the soapClient call.

7. Parcel follow up (Tracking)

Based on our partner's decision we can send tracking information about the status changes occured during the parcels' life cycle. For this it is a necessity to have a WEB API (REST) taker side (https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design implemented on our contracted parter's side to where we can send out the data with HTTP POST call.

It is highly recommended to place the API in the following data structure: https://websitename.com/api/tracking

It is also recommended to set up the website with https certification. This way we can send the tracking data through a secured channel. Please note, that without having https certification we cannot and do not assume any responsibility for any data may get in posession of a third party.

Information regarding the parcels' status change will be sent out approximately in every 10 minutes. Status changes occured in the last 10 minutes for any parcel connected to the partner's premises will be included in the JSON object, for both Pick Pack Point and Home delivery.

If there was a server outage or we got any failed response back (not success), we would try to resend the data. It is a possibilty, that we send out new status change(s) as well. This is to be handled on or partner's side. It is possible to check with the tracking ID it, if the given status change has successfully logged or not. The logic order of the status changes is given by the EvenCreatedDate.

7.3. Request

ContentType for the http request: „application/json; charset=utf-8", the data will be sent out with http POST call. The format of the JSON file and the detailed description of further data can be found below. All necessary data and statuses will be sent out so our partner has the opporunity to decide in their circle of comepetence which ones are to be logged/stored, with which logic they wish to process them.

UnifiedParcelTrackingInfo

Type: Class

Description:

This class takes in all the tracking information.

Fields:

Type	Field name	Remark	Mandatory
string	**PartnerCode**	Partner (Sender) identifier	**Yes**
List<TrackingInfo>	**TrackingInfoList**	List of tracking data	**Yes**

TrackingInfo

Type: Class

Description:

The TrackingInfoList includes all information concerning parcel status changes. It is indicated in the „Service" column, if the given field should be filled only for Pick Pack Point (PPP), only for home delivery (HD) or both (PPP+HD)

Main fields

The following fields include all the basic information and data for processing the parcel tracking information.

Type	Field name	Remark	Mandatory	Service
int	**TrackingID**	Unique tracking identifier. With which it is possible to check, if we have already sent this data or not.	**Yes**	**PPP+HD**
string	**ParcelBarcode**	Parcel identifier. This is the parcel's identifier for logistic	**Yes**	**PPP+HD**
DateTime	**EventCreatedTime**	Time of the status change	**Yes**	**PPP+HD**
string	**DeliveryMode**	Delivery mode, its possible values: PICKPACKPOINT or HOMEDELIVERY	**Yes**	**PPP+HD**

Statuses

Basically we diffenrentiate two type of statuses: Partner and Customer tracking status.

The **Partner tracking status** provides detailed information on the parcel's actual state. This type of tracking provides us with detailed information on the parcel's state. It has a Partner main status (**PartnerMainSatatus**) and Partner sub-status (**PartnerSubStatus**), the latter provides further and more detailed information. (For instance: main status: „Under processing", sub-status can be: „Parcel arrived (Warehouse processing)" or „Under delivery"). These statuses can be found on our Partner Portal under the Partner tracking information. https://portal.pickpackpont.hu/#/tracking

The **Customer Tracking status** (**CustomerStatus**) includes all information relevant for the customer, recipient. (For instance „Arrived to Pick Pack Point. Pick Pack Point name: xxxx, Pick Pack Point address:" Customer can find these information on https://www.pickpackpont.hu/ or on https://www.sprinter.hu/ in the Parcel search menu.

All possible statuses can be found in a file named **TrackingStatus.xlsx**. In case of Customer tracking: where NULL value can be found, the latest status will be sent out. The **Reason of refusal (RefuseReason)** is for home derlivery parcels only. It is used when the recipient has decided not to take over the parcel. This field includes the cause of the refusal.

Each status information has 2 fields. A **Name** field, which is the unique name of the status. IT logic (mapping) can be built up on them. (For instance: WAITING_FOR_PACKAGE, DELIVERED). And a **DescriptionList** field. This includes the wording of the statuses - which is easy to understand for the end users - in both Hungarian and English. Please note, that the descriptions may change in the future!

Type	Field name	Remark	Mandatory	Service
string	**PartnerMainStatusName**	Name of partner main status	**Yes**	**PPP+HD**
List<Translation>	**PartnerMainStatusDescriptionList**	Description of partner main status	**Yes**	**PPP+HD**
string	**PartnerSubStatusName**	Name of partner sub-status	**No**	**PPP+HD**
List<Translation>	**PartnerSubStatusDescriptionList**	Discription of partner sub-status	**No**	**PPP+HD**
string	**CustomerStatusName**	Name of customer status	**Yes**	**PPP+HD**
List<Translation>	**CustomerStatusDescriptionList**	Description of customer status	**Yes**	**PPP+HD**
string	**RefuseReasonName**	Name of storno reason	**No**	**HD**
List<Translation>	**RefuseReasonDescriptionList**	Description of storno reason	**No**	**HD**

Other identifiers

In order to be able to ease parcel identification we send other identifiers known by our partner.

Type	Field name	Remark	Mandatory	Service
string	**HomeDeliveryNoteNumber**	Waybill identifier. In case of Home Delivery parcels it is an identifier for parcels to be shipped to the same address.	**No**	**HD**
string	**ShipmentID**	Shipment ID connected to the parcel, which is generated by the system. In case of PPP service, 1 ID is generated per ServiceType.	**No**	**PPP+HD**

		In case of HD service each Parcel has a separated ID.		
string	**OriginalBarCode**	Original barcode with which the parcel was registered	**No**	**PPP+HD**
string	**AutorizationCode**	Order code, which is the partner's own parcel code or invoice number (if an invoice can be found in the box).	**No**	**PPP+HD**
string	**InvoiceNumber**	**This field will be discontinued.** Invoice number placed in the parcel, AuthorizationCode takes its place.	**No**	**HD**

Other information

Type	Field name	Remark	Mandatory	Service
string	**CustomerName**	Customer name	**No**	**PPP+HD**
DateTime?	**ExpectedArrivalFrom**	Start of expected arrival	**No**	**PPP+HD**
DateTime?	**ExpectedArrivalTo**	End of expected arrival	**No**	**PPP+HD**
string	**DestinationLocationCode**	Identifier of destination Pick Pack Point	**No**	**PPP**
string	**DestinationLocationName**	Name of destination Pick Pack Point	**No**	**PPP**
string	**DestinationLocationAddress**	Address of destination Pick Pack Point	**No**	**PPP**
string	**InternationalDeliveryPartnerName**	International transportation company, (FanCourier)	**No**	**HD**
Decimal?	**MeasuredWeight**	Parcel's measured weight in kilogramm	**No**	**HD**

Translation

Type: Class

Description:

This includes the description of the statuses for the end users in both Hungarian and English.

Fields:

Type	Field name	Remark	Mandatory
string	**LanguageName**	`Name of language, HU or EN.`	**Yes**
string	**TranslatedText**	`Translated description`	**Yes**

7.4. Response

The response is a standard HttpResponse, with the appropriatlely filled HttpStatusCode. In case of successful registration: with **HttpStatusCode 200 OK** and content '**success**'. Sending back success response is mandatory so we are aware of the saving was successfully completed.

7.4.1. Error codes

The following HttpStatusCodes can be occured in the response. Further details can be found on the following link: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Status code	Status name	Content	Remark
200	OK	success	Successful registration
404	Not found	No HTTP resource was found	The given URL does not exist
405	Method Not Allowed	Request method must be POST!	The request method was not POST.
415	Unsupported Media Type	Content type must be: application/json; charset=utf-8. Request content type was:	The content of the request is not in the expected format
400	Bad Request	The request is invalid.	The received object is not appropriate, that is why it cannot be processed. We received the object in different format, then we expected.
500	Internal Server Error	Error occured! ErrorMessage:	Unexpected error has occurred

7.5. Source code demo

Demo codes (C# and PHP) for getting tracking information on server side are enclosed in the document

7.5.1. C# code

.Net source for the WebAPI: can be found in the folder named csharp.zip

csharp/Tracking/PUDOParcelTracking.sln

The sln is going to load the code example written below after opening the Visual Studio 2015.

An ASP.NET Web API 2.0 project can be found in the demo. The project can be run under IIS or IIS express, the necessary NuGet Packages should be Restored, if they would not be automatically downloaded by the Visual Studio.

The Controller for the API and the Post call receiver method can be found in the Controllers\TrackingController.cs file. It expects a UnifiedParcelTrackinginfo object in Parameter, which is to be built up according to the above mentioned data structure and can be found under the folder named Models.

According to the demo code it is necessary to check, if the request is appropriate so we can get the adequate respoonse back.

The following information should always be sent back in the response:

```
return Request.CreateResponse(HttpStatusCode.OK, "success");
```

7.5.2. PHP code

PHP source for the WebAPI: can be found int he folder named php.zip

php/Tracking/TrackingController.php

The demo code expects an HTTP post request, performs check on the request to see if it is approriate.

If it proved to be POST request with `'application/json; charset=utf-8'` ContentType, then the data sent by our partner is to be handled as JSON.

```
//Receive the RAW post data.

$request = trim(file_get_contents("php://input"));


//Attempt to decode the incoming RAW post data from JSON.
$contentObjectFromJson = json_decode($request);
```

The JsonMapper class (https://github.com/cweiske/jsonmapper) is used as follows: to transfrom the JSON obejct to `UnifiedParcelTrackingInfo` type and we also validate that the mandtory fields are filled, the data format is appropriate. If the validation fails, and error message will be sent back.

Afterwards the JSON object or the `UnifiedParcelTrackingInfo type` object can be processed at any time.

The reponse is to be always sent back with http status 200 and „success” message.

```
//At the end return success with HTTP status 200
echo "success";
```

7.5.3. Testing

The demo codes can be tested any kind of application that is capable of calling REST API, for instance Soap UI (https://www.soapui.org/). For the testing: it is important to send all the data with POST method and the Header ContentType shold be „application/json”!

Demo JSON data: one Pick Pack Point and one HD parcel (in order to save some place it was copied, however the formatted version can be seen with web tools. (https://jsonformatter.org/):

{"PartnerCode":"0000001000","TrackingInfoList":[{"TrackingID":1,"ParcelBarcode":"tracking_test_PPP","HomeDeliveryNoteNumber":null,"ShipmentID":"SP201802270000000012","OriginalBarCode":"tracking_test_PPP","AutorizationCode":"auth_tracking_test_PPP","InvoiceNumber":null,"EventCreatedTime":"2018-03-27T09:06:03.5249597+02:00","DeliveryMode":"PICKPACKPOINT","PartnerMainStatusName":"WAITING_FOR_PACKAGE","PartnerMainStatusDescriptionList":[{"LanguageName":"HU","TranslatedText":"A csomag beérkezésére várunk"},{"LanguageName":"EN","TranslatedText":"Waiting for the parcel to arrive"}],"PartnerSubStatusName":null,"PartnerSubStatusDescriptionList":null,"CustomerStatusName":"WAITING_FOR_PACKAGE_SENDER","CustomerStatusDescriptionList":[{"LanguageName":"HU","TranslatedText":"A csomagot még nem adta át a Feladó szállításra. További információért kérjük keresse a Feladót (webáruházat)."},{"LanguageName":"EN","TranslatedText":"The sender haven't handed over the parcel yet. For more information, please call the sender (webshop)."}],"RefuseReasonName":null,"RefuseReasonDescriptionList":null,"CustomerName":"Teszt Elek","ExpectedArrivalFrom":"2018-03-28T09:06:03.5259473+02:00","ExpectedArrivalTo":"2018-03-29T09:06:03.5259473+02:00","DestinationLocationCode":"0000101005","DestinationLocationName":"Teszt Shop","DestinationLocationAddress":"1164, Budapest, Szabadföldi út 1.","InternationalDeliveryPartnerName":null,"MeasuredWeight":null},{"TrackingID":2,"ParcelBarcode":"tracking_test_HD","HomeDeliveryNoteNumber":"tracking_test_HD","ShipmentID":"SP201802270000000013","OriginalBarCode":"tracking_test_HD","AutorizationCode":"auth_tracking_test_HD","InvoiceNumber":null,"EventCreatedTime":"2018-03-27T09:06:03.5269488+02:00","DeliveryMode":"HOMEDELIVERY","PartnerMainStatusName":"WAITING_FOR_PACKAGE","PartnerMainStatusDescriptionList":[{"LanguageName":"HU","TranslatedText":"A csomag beérkezésére várunk"},{"LanguageName":"EN","TranslatedText":"Waiting for the parcel to arrive"}],"PartnerSubStatusName":null,"PartnerSubStatusDescriptionList":null,"CustomerStatusName

":"SPRINTER_WAITING_FOR_PACKAGE_SENDER","CustomerStatusDescriptionList":[{"LanguageName":"HU","TranslatedText":"A csomagot még nem adta át a Feladó szállításra. További információért kérjük keresse a Feladót (webáruházat)."},{"LanguageName":"EN","TranslatedText":"The sender haven't handed over the parcel yet. For more information, please call the sender (webshop)."}],"RefuseReasonName":null,"RefuseReasonDescriptionList":null,"CustomerName":"Teszt Elek","ExpectedArrivalFrom":"2018-03-28T09:06:03.5269488+02:00","ExpectedArrivalTo":"2018-03-29T09:06:03.5269488+02:00","DestinationLocationCode":null,"DestinationLocationName":null,"DestinationLocationAddress":null,"InternationalDeliveryPartnerName":null,"MeasuredWeight":3.14}]}