



Universal PUSHER ORLEN Paczka

1 TABLE OF CONTENTS

1	Table of contents.....	1
2	Change history.....	2
3	Introduction.....	3
4	Scope of the document	3
5	Architecture and method of operation	4
5.1	Implementation process.....	4
5.2	Calls to ORLEN Paczka API	4
5.3	Behavior in case of client system unavailability.....	4
6	Authorization and client-side requirements	5
6.1	Access credentials for ORLEN Paczka API (SOAP).....	5
6.2	Types of client endpoint authorization	5
6.2.1	Basic.....	5
6.2.2	OAuth2	6
6.2.3	OAuth2StaticToken.....	6
6.3	Requirements for the client endpoint.....	6
7	Implementation.....	6
7.1	WebhookRegister	7
7.1.1	Request.....	7
7.1.2	Response	8
7.1.3	Example registration errors.....	9
7.2	WebhookStatus	9
7.2.1	Request.....	9
7.2.2	Response	10
7.3	WebhookUnregister	11
7.3.1	Request.....	11
7.3.2	Response	11
8	Returned data format (notification payload).....	12
9	Deployment and testing scenario	13

9.1	Preparation.....	13
9.2	Registration test – Basic	13
9.3	Registration test – OAuth2	13
9.4	Registration test – OAuth2StaticToken	14
9.5	Status and deregistration test.....	14
10	Recommendations and best practices on the client side.....	14
11	Support.....	15
12	Example of an endpoint in C#	15

2 CHANGE HISTORY

Date	Version	Changes
11.2025	1.0	First release

3 INTRODUCTION

Universal Pusher, internally called **Pack Event Notifier**, is a *webhook* based mechanism designed to deliver real-time package status updates to the client. Instead of periodically querying the **ORLEN Paczka API (SOAP)** for statuses, the ORLEN Paczka system automatically sends a notification each time a package status changes.

Business benefits:

- immediate receipt of status change information (the *push* is performed as soon as a new status appears),
- reduction of network traffic on the API side,
- simplified client-side integration (no need to build polling schedules for status-tracking methods).

To use **Universal Pusher**, you need to:

1. Provide your own **REST endpoint** (URL) that will receive notifications,
2. Register this **endpoint** in ORLEN Paczka system using the *WebhookRegister* method.

Supported types of client-side endpoint authorization:

- **Basic** – login and password,
- **OAuth2** – obtaining an access token from the client's dedicated endpoint,
- **OAuth2StaticToken** – a static *Bearer* token provided by the client.

4 SCOPE OF THE DOCUMENT

This document is intended for development teams of ORLEN Paczka clients who will be implementing **Universal Pusher** integration in their systems. It serves as a supplement to the **ORLEN Paczka API (SOAP)** documentation (available at <https://www.orklenpaczka.pl/integracja-z-orklen-paczka/> in the API Documentation section).

It describes:

- business and technical context of **Universal Pusher**,
- methods of client-side authorization,
- detailed usage of the following methods:
 - *WebhookRegister*,
 - *WebhookStatus*,
 - *WebhookUnregister*,
- recommended implementation and testing scenario,
- data format used in notifications.

5 ARCHITECTURE AND METHOD OF OPERATION

5.1 IMPLEMENTATION PROCESS

1. The client has an active agreement with **ORLEN Paczka** and uses **ORLEN Paczka API (SOAP)**.
2. Client creates a **RESTful HTTP(S) endpoint** in their system that accepts **POST** requests with content in **JSON** format.
3. Client registers this endpoint by calling the *WebhookRegister* method in **ORLEN Paczka API (SOAP)**.
4. After successful registration, **ORLEN Paczka** system:
 - sends a **test notification** for a package with a number, e.g., **99999999999999**, to confirm the configuration is correct,
 - then, for each package handled under the given **PartnerID**, sends a notification whenever the status changes.
5. Client can at any time:
 - check current pusher configuration using *WebhookStatus* method,
 - unregister the endpoint using the *WebhookUnregister* method (methods described in the following sections).

5.2 CALLS TO ORLEN PACZKA API

Pusher methods are available in public **ORLEN Paczka API (SOAP)**.

- **Production environment:**
<https://api.orlenpaczka.pl/WebServicePwRProd/WebServicePwR.asmx?wsdl>
- **Test environment:**
<https://api-test.orlenpaczka.pl/WebServicePwR/WebServicePwR.asmx?WSDL>

Authorization to **ORLEN Paczka API (SOAP)** is performed – just like in other **ORLEN Paczka API (SOAP)** methods—by including the *<PartnerID>* and *<PartnerKey>* fields in the request.

5.3 BEHAVIOR IN CASE OF CLIENT SYSTEM UNAVAILABILITY

If **ORLEN Paczka** system does not receive an HTTP 200 response from the client's endpoint:

- notification is considered **not accepted**,

- system retries sending **every 5 minutes** until successful (until **endpoint** returns **HTTP 200**) or until client unregisters **pusher**,
- if the **pusher** is unregistered using *WebhookUnregister*, **the queue of pending notifications is cleared**.

6 AUTHORIZATION AND CLIENT-SIDE REQUIREMENTS

6.1 ACCESS CREDENTIALS FOR ORLEN PACZKA API (SOAP)

ORLEN Paczka API (SOAP) login credentials:

- *<PartnerID>* – client login provided by **ORLEN Paczka**,
- *<PartnerKey>* – client password provided by **ORLEN Paczka**.

Production environment credentials are provided by the sales representative.

6.2 TYPES OF CLIENT ENDPOINT AUTHORIZATION

In parameters of the *WebhookRegister* method, client specifies the authorization method for their endpoint using the *<AuthorizationType>* field:

- *Basic*
- *OAuth2*
- *OAuth2StaticToken*

Each type requires a different set of fields:

Field	Basic (AuthorizationType=Basic)	OAuth2 (AuthorizationType=OAuth2)	OAuth2StaticToken (AuthorizationType=OAuth2StaticToken)
<i>UserName</i>	user login (required)	ClientId (required)	null (not used)
<i>Password</i>	password (required)	password/ClientSecret (required)	token (required)
<i>LoginUrl</i>	null (not used)	URL to obtain the token (required)	null (not used)
<i>NotificationUrl</i>	webhook address (required)	webhook address (required)	webhook address (required)

6.2.1 Basic

- **ORLEN Paczka** sends a POST notification to the *NotificationUrl* with the header:
 - *Authorization: Basic <base64(UserName:Password)>*.
- On the client side, the **endpoint** should handle standard **Basic authorization**.

6.2.2 OAuth2

- Before sending a notification, **ORLEN Paczka** system:
 1. calls client's *LoginUrl*, passing the *UserName* (**ClientId**) and *Password* (**ClientSecret**) provided in *WebhookRegister* method request,
 2. receives an **access token**,
 3. uses it in the header:
 - *Authorization: Bearer <access_token>*.

6.2.3 OAuth2StaticToken

- Client provides a static token (e.g., *eyJhbGciOi...*) in the *Password* field.
- **ORLEN Paczka** system does not fetch the token dynamically – it always uses the value provided during registration:
 - *Authorization: Bearer <Password>*.

6.3 REQUIREMENTS FOR THE CLIENT ENDPOINT

Client endpoint (**NotificationUrl**) should:

- accept **HTTP POST** requests with content in **JSON** format (UTF-8),
- return **HTTP 200** upon successful processing of the notification (regardless of internal logic, e.g., database write/logging),
- return a code other than **HTTP 200** in case of an error – **ORLEN Paczka** system will treat this as not accepted and will retry sending,
- be accessible via **HTTPS** (recommended),
- have a response timeout that allows returning a message in an optimal time (e.g., **up to 10 seconds**).

7 IMPLEMENTATION

WebhookRegister, *WebhookStatus*, *WebhookUnregister* methods are available in **ORLEN Paczka API (SOAP)**.

Production environment:

<https://api.orldenpaczka.pl/WebServicePwRProd/WebServicePwR.asmx?wsdl>

Test environment:

<https://api-test.orldenpaczka.pl/WebServicePwR/WebServicePwR.asmx?WSDL>

7.1 WEBHOOKREGISTER

WebhookRegister method is used to register a new client endpoint in **ORLEN Paczka** system. This is the first step required to start receiving notifications about package statuses. After successful registration, a **push notification** is sent for a package with the number **9999999999999999**.

Example:

```
"PackCode":9999999999999999,"PackCodePrev":null,"Status":"200","Updated":"2000-01-01T00:00:00","Operator":"ORLEN-PACZKA"
```

7.1.1 Request

Request parameters:

<PartnerID>	Customer's id given by ORLEN Paczka	(REQUIRED) 10 char
<PartnerKey>	Customer's key given by ORLEN Paczka	(REQUIRED) 10 char
<AuthorizationType>	Authorization types – Allowed values: <ul style="list-style-type: none">• Basic• OAuth2• OAuth2StaticToken	(REQUIRED) 30 char
<UserName>	<ul style="list-style-type: none">• for Basic – user login,• for OAuth2 – ClientId,• for OAuth2StaticToken – not used (null / empty).	30 char
<Password>	<ul style="list-style-type: none">• for Basic – user password,• for OAuth2 – password / ClientSecret,• for OAuth2StaticToken – token (e.g., Bearer) used directly in the Authorization header.	70 char
<LoginUrl>	<ul style="list-style-type: none">• for OAuth2 – URL to obtain the access token,• for Basic and OAuth2StaticToken – not used (null / puste).	100 char
<NotificationUrl>	Client webhook URL to which notifications about package status changes will be sent.	(REQUIRED) 100 char

Required fields for each *<AuthorizationType>* authorization type:

AuthorizationType	Basic	OAuth2	OAuth2StaticToken
UserName	UserName (required)	ClientId (required)	null
Password	Password (required)	Password (required)	Token (required)
LoginUrl	null	Login URL (required)	null
NotificationUrl	Client notification URL (required)	Client notification URL (required)	Client notification URL (required)

Example with minimum required fields – Basic authorization:

```
<PartnerID>1234567890</PartnerID>
<PartnerKey>abcdefghijk</PartnerKey>
<AuthorizationType>Basic</AuthorizationType>
<UserName>login</UserName>
<Password>password</Password>
<LoginUrl></LoginUrl>
<NotificationUrl>https://your-system.example.com/pack-events/notification</NotificationUrl>
```

7.1.2 Response

Returned values:

<Err>	Error number (`0` indicates no error)
<ErrDes>	Error description (empty if no error)
<Data>	Section with information about the registered webhook

<Data> structure	
<PartnerID>	Client identifier given by ORLEN Paczka
<Status>	Registration status(`Active`, `Unregistered`, etc.)
<Annotation>	Registration description (e.g. `Webhook registered and activated`)
<AuthorizationType>	Authorization type
<UserName>	Client endpoint login / ClientId
<Password>	Client endpoint password or token
<LoginUrl>	Login URL (for OAuth2)
<NotificationUrl>	Client webhook address

Example:

```
<Err>0</Err>
<ErrDes />
<Data>
<PartnerID>1234567890</PartnerID>
<Status>Active</Status>
<Annotation>Webhook registered and activated</Annotation>
<AuthorizationType>Basic</AuthorizationType>
<UserName>login</UserName>
<Password>password</Password>
<NotificationUrl>http://domain.com/PackEventNotifierBasicExample/notification</NotificationUrl>
</Data>
```


7.1.3 Example registration errors

Common issues detected during testing:

1. Invalid login credentials for client endpoint

- For example, incorrect *UserName* / *Password* (Basic) or an invalid token (*OAuth2StaticToken*).
- **Effect:** client system rejects requests, e.g., **HTTP 401 Unauthorized**.
- In pusher registration logs, an entry appears indicating that registration ended with the status ``incorrect`` and with a message such as:
request failed, received HTTP code Unauthorized (Unauthorized).

2. Correct authorization data, but incorrect URL

- *NotificationUrl* indicate to a non-existent **endpoint**.
- **Effect:** client system returns, for example, **HTTP 404 Not Found**.
- Example message:
request failed, received HTTP code NotFound (Not Found).

3. Missing required fields depending on AuthorizationType

- Missing *UserName* or *Password* for *Basic*,
- Missing *LoginUrl* for *OAuth2*,
- Missing *Password* (token) for *OAuth2StaticToken*,
- Invalid *AuthorizationType* value (other than *Basic*, *OAuth2*, *OAuth2StaticToken*).

In each of the above cases, **Err** will be different from **0**, and error details will appear in **ErrDes**.

7.2 WEBHOOKSTATUS

WebhookStatus method is used to check current configuration and status of a registered endpoint. It can be used both:

- during implementation (to verify that registration was successful),
- during maintenance (monitoring – e.g., in periodic integration checks).

7.2.1 Request

Request parameters:

<PartnerID>	Customer's id given by ORLEN Paczka	(REQUIRED) 10 char
<PartnerKey>	Customer's key given by ORLEN Paczka	(REQUIRED) 10 char

Example with minimum required fields:

```
<PartnerID>1234567890</PartnerID>
<PartnerKey>abcdefghijk</PartnerKey>
```

7.2.2 Response

Returned values:

<Err>	Error number
<ErrDes>	Error description
<Data>	<Data> structure returning informations

<Data> structure	
<PartnerID>	Client identifier given by ORLEN Paczka
<Status>	Pusher status (<i>Active</i> or <i>Unregistered</i>)
<Annotation>	Description of current status (e.g. <i>Webhook registered and activated</i>)
<AuthorizationType>	Authorization type
<UserName>	Consistent with current configuration
<Password>	Consistent with current configuration
<LoginUrl>	Consistent with current configuration
<NotificationUrl>	Consistent with current configuration

Example (pusher active):

```
<Err>0</Err>
<ErrDes />
<Data>
  <PartnerID>1234567890</PartnerID>
  <Status>Active</Status>
  <Annotation>Webhook registered and activated</Annotation>
  <AuthorizationType>Basic</AuthorizationType>
  <UserName>login</UserName>
  <Password>password</Password>
  <NotificationUrl>http://domain.com/PackEventNotifierBasicExample/notification</NotificationUrl>
</Data>
```

Example (pusher unregistered):

```
<Err>0</Err>
<ErrDes />
<Data>
  <PartnerID>1234567890</PartnerID>
  <Status>Unregistered</Status>
  <Annotation>Webhook unregistered</Annotation>
  <AuthorizationType>Basic</AuthorizationType>
  <UserName>login</UserName>
  <Password>password</Password>
  <NotificationUrl>http://domain.com/PackEventNotifierBasicExample/notification</NotificationUrl>
</Data>
```

7.3 WEBHOOKUNREGISTER

WebhookUnregister method is used to **disable** Universal Pusher notifications for a given **PartnerID**. After unregistration:

- **ORLEN Paczka** system stops sending new notifications,
- queue of pending notifications (which have not yet been successfully delivered) is cleared.

7.3.1 Request

Request parameters:

<PartnerID>	Customer's id given by ORLEN Paczka	(REQUIRED) 10 char
<PartnerKey>	Customer's key given by ORLEN Paczka	(REQUIRED) 10 char

Example with minimum required fields:

```
<PartnerID>1234567890</PartnerID>
<PartnerKey>abcdefghijk</PartnerKey>
```

7.3.2 Response

Returned values:

<Err>	Error number
<ErrDes>	Error description
<Data>	<Data> structure returning informations

<Data> struktura	
<PartnerID>	Client identifier given by ORLEN Paczka
<Status>	Pusher status
<Annotation>	Description of current status
<AuthorizationType>	Authorization type
<UserName>	Configuration that has been disabled
<Password>	Configuration that has been disabled
<LoginUrl>	Configuration that has been disabled
<NotificationUrl>	Configuration that has been disabled

Example:

```
<Err>0</Err>
<ErrDes />
<Data>
  <PartnerID>1234567890</PartnerID>
  <Status>Unregistered</Status>
  <Annotation>Webhook unregistered</Annotation>
  <AuthorizationType>Basic</AuthorizationType>
```

```
<UserName>login</UserName>
<Password>password</Password>
<NotificationUrl>http://domain.com/PackEventNotifierBasicExample/notification</NotificationUrl>
</Data>
```

8 RETURNED DATA FORMAT (NOTIFICATION PAYLOAD)

After each package status change, **ORLEN Paczka** system sends a **POST** request to **NotificationUrl** with a **JSON** body:

```
{
  "PackCode": PACKAGE_NUMBER,
  "PackCodePrev": ORIGINAL_PACKAGE_NUMBER,
  "Status": STATUS+ATTRIBUTE,
  "Updated": DATE_TIME,
  "Operator": OPERATOR
}
```

Field descriptions:

- **PackCode:**
Package number (EAN-13), a 13-digit number.
- **PackCodePrev:**
Original package number (EAN-13), a 13-digit number, or null if there is no association.
- **Status:**
Package status number (e.g., 200, 210, etc.) – according to **ORLEN Paczka** status list.
ATTRIBUTE:
 - no attribute – for standard packages.
Example. „Status”: **200**
 - **_POWROT** or **_2_POWROT** - operational return attributes.
Example. „Status”: **100_POWROT** or **100_2_POWROT**
 - **_ZWROT** - consumer return attribute
Example. „Status”: **100_ZWROT**
- **Updated:**
Date and time of the status update in **ISO 8601** format, e.g.,
2025-06-26T01:02:03.225694.
Time is returned in the **UTC+1 timezone (Polish local time)**.
- **Operator**
Operator name: "ORLEN-PACZKA".

Full event example:

```
{
  "PackCode": 1234567890123,
  "PackCodePrev": null,
  "Status": "200",
  "Updated": "2025-06-26T01:02:03.225694",
```

```
"Operator": "ORLEN-PACZKA"  
}
```

Complete list of statuses with descriptions is available in **ORLEN Paczka API (SOAP)** documentation in section **7. PARCEL STATUSES**.

Documentation is available at <https://www.orklenpaczka.pl/integracja-z-orklen-paczka/> in **API Documentation** section.

9 DEPLOYMENT AND TESTING SCENARIO

The following scenario is a summary of an actual implementation and can be applied to any integration.

9.1 PREPARATION

1. Receive from your sales representative:
 - PartnerID and PartnerKey for test environment
2. On your side, prepare:
 - NotificationUrl endpoint (*REST, HTTPS, POST JSON*),
 - handling of the chosen authorization method (*Basic / OAuth2 / OAuth2StaticToken*),
 - logging of all incoming notifications (payload, headers, response codes).

9.2 REGISTRATION TEST – BASIC

1. Call *WebhookRegister* method in test environment with:
 - *AuthorizationType = Basic*,
 - filled-in fields: *UserName, Password, NotificationUrl*.
2. Check response:
 - *Err = 0*,
 - *Status = Active*,
 - correct *NotificationUrl*.
3. Verify your system logs:
 - test notification for package **99999999999999** should appear.
4. Register a test package using your *PartnerID* (e.g., using the shipping methods in ORLEN Paczka API (SOAP)) and verify that you received status **200** for your package.

9.3 REGISTRATION TEST – OAUTH2

1. Set up on your side a *LoginUrl* endpoint that generates an access token.
2. Call *WebhookRegister* with:
 - *AuthorizationType = OAuth2*,

- *UserName* = *ClientId*,
 - *Password* = *ClientSecret*,
 - *LoginUrl* = endpoint URL for obtaining token,
 - *NotificationUrl* = webhook URL.
3. Verify response and test notification.
 4. Test both valid and invalid data (incorrect *LoginUrl*, incorrect *NotificationUrl*, invalid *UserName*) to observe the corresponding errors.

9.4 REGISTRATION TEST – OAUTH2STATICTOKEN

1. Prepare a static token that will be used in *Authorization: Bearer* header.
2. Call *WebhookRegister* with:
 - *AuthorizationType* = *OAuth2StaticToken*,
 - *Password* = *token*,
 - *NotificationUrl* = *webhook URL*,
 - *UserName* and *LoginUrl* – empty (*null*).
3. Verify reception of test and status notifications.

9.5 STATUS AND DEREGISTRATION TEST

1. After successful registration, call *WebhookStatus* and ensure that:
 - *Status* = *Active*,
 - *AuthorizationType*, *NotificationUrl* and other fields match configuration.
2. Next, call *WebhookUnregister*.
3. Call *WebhookStatus* again:
 - Status should be *Unregistered*.
4. Create a test package and ensure that you **no longer receive notifications** – this confirms that the pusher has been successfully disabled.

10 RECOMMENDATIONS AND BEST PRACTICES ON THE CLIENT SIDE

Idempotency

Endpoint implementation should be **idempotent** – if the same event is sent more than once (e.g., due to a retry), the client system must not create duplicates or perform the same business action multiple times.

Logging and monitoring

It is recommended to log:

- date/time the notification was received,
- full **JSON** payload,
- headers (e.g., *Authorization*, *CorrelationId* if used),
- client system's response code.

Security

- endpoint should operate over **HTTPS**,
- login credentials/tokens should be stored securely (e.g., in a secret manager).

Error handling

In case of internal errors (e.g., database unavailability), the **endpoint** should return a code other than **HTTP 200** to allow the **ORLEN Paczka** system to retry delivering the event. It is recommended to use codes from the **HTTP 400** family.

11 SUPPORT

For questions regarding:

- Universal Pusher configuration,
- selection of authorization type,
- details about package statuses,

please contact your ORLEN Paczka sales representative or the technical support team (integracje@orlenpaczka.pl), providing:

- client number (**PartnerID**),
- environment (test / production),
- a brief description of the issue,
- an example package (PackCode) and the time range when the issue occurred.

12 EXAMPLE OF AN ENDPOINT IN C#

It is available in the file „PublicUniversalPusherExample.zip”.



ORLEN Paczka Pack Event Notifier Basic Authorization

Projekt `PackEventNotifierBasicExample` to przykładowa implementacja endpointu na który ORLEN Paczka może przysyłać dane autoryzując się określoną nazwą użytkownika i hasłem.

Test lokalny

1. Otwórz w Postmanie [OrlenPaczkaPackEventNotifierExample.postman_collection.json](#)
2. Przejdź do `OrlenPaczkaPackEventNotifierExample > Basic Example`
3. Jeżeli trzeba podmień adres w requście na `http://domena.com/PackEventNotifierBasicExample/notification`
4. Zdarzenie zostanie wyświetlone na panelu poniżej
5. W celu symulacji stanu serwera użyj przełącznika 'Akceptuj notyfikacje'

☒ Akceptuj notyfikacje

Ostatnie odświeżenie:
28.11.2025, 13:38:48

```
2025-11-28 13:37:59Z OK {"PackCode":9999999999999999,"PackCodePrev":null,"Status":"200","Updated":"2000-01-01T00:00:00","Operator":"ORLEN-PACZKA"}
```

The “Akceptuj notyfikacje” checkbox allows testing the blocking of **push** messages for the registered **endpoint**. When unchecked, sent requests will not be received, and messages with the description **REJECT** will appear on the list (replacing the **OK** message shown in the screenshot).