cairo.API

- Informacje podstawowe
- Dropshipping
- Dokumentacja

Informacje podstawowe

cairo.API

System cairo.ERP udostępnia klientom funkcjonalności API w oparciu o komunikację JSON poprzez metodę HTTP `POST`. W odróżnieniu od klasycznej architektury REST, dostęp do funkcji odbywa się poprzez wywoływanie nazwanych operacji, przekazywanych w ciele żądania. Wszystkie dane — zarówno wejściowe, jak i odpowiedzi — przekazywane są w formacie JSON. API umożliwia przesyłanie parametrów (np. identyfikatora sesji) w treści żądania, co eliminuje konieczność ich przekazywania w URL-u lub nagłówkach.

> Komunikacja XML nie jest już wspierana.

Komunikacja odbywa się z wykorzystaniem jednego punktu końcowego:

POST https://api.example.com/ws/<nazwa metody>

Przykładowe pobranie wersji systemu:

```
POST /ws/getVersion HTTP/1.1
Host: api.example.com
Content-Type: application/json; charset=utf-8
SOAPAction: "getVersion"

{
  "getVersion": {}
}
```

Odpowiedź:

```
{
  "getVersionResponse": {
    "ownum": "123",
    "verFull": "5.645.15127 2024-09-27",
    "version": "dbfd2 5.645.15127 2024-09-27 JSON"
  }
}
```

Struktura zapytań

Każde zapytanie powinno mieć jednolitą strukturę:

```
{
 "<nazwaOperacji>": {
  <parametryWejściowe>
 }
}
```

Np.:

```
{
 "getProductsInfo": {
  "sessionId": "abc123",
  "productList": {
   "product": [{
      "id": "0006VI"
    }, {
      "reference": "144 666"
   }]
  }
 }
}
```

Struktura odpowiedzi

Poprawna odpowiedź:

```
{
 "<nazwa_operacji>Response": {
  <dane_wyjściowe>
 }
}
```

Np.:

```
{
 "getVersionResponse": {
  "ownum": "123",
  "verFull": "5.645.15127 2024-09-27",
```

```
    "version": "dbfd2 5.645.15127 2024-09-27 JSON"
  }
}
```


Błędy

W przypadku wystąpienia błędu serwer zwraca odpowiedź HTTP 500 z obiektem `error` w formacie JSON:


```
{
  "error": {
    "code": "ERR_SESSION",
    "msg": "Nieprawidłowy identyfikator sesji"
  }
}
```


Protokół komunikacyjny

Dostęp do API odbywa się poprzez połączenie HTTP(S) do serwera. Wszystkie żądania muszą być wysyłane z nagłówkiem `Content-Type: application/json; charset=utf-8`. Komunikacja jest bezstanowa — sesje są identyfikowane przez parametry przesyłane w ciele zapytań.

Dostęp do API

Aby uzyskać dostęp, potrzebujesz znać host, z którym będziesz się łączył oraz uzyskać dane dostępowe do API. W tym celu skontaktuj się ze swoim opiekunem.

Wersja językowa

Domyślnie wszystkie komunikaty zwracane są w języku systemu cairo.ERP. Jeśli chcesz, aby odpowiedź przychodziła w konkretnym języku, wyślij właściwy kod języka w polu `languageId` podczas logowania metodą `doLogin`.

Zasady pracy z sesjami

Dostęp do metod udostępnianych przez API wymaga przejścia procesu logowania. Logowanie realizowane jest poprzez wywołanie metody `doLogin`, w której należy podać login i hasło. Metoda `doLogin` zwraca unikalny identyfikator sesji (`sessionId`), który należy dołączać do każdego kolejnego wywołania API jako parametr w ciele żądania.

Przykład: logowanie i użycie sesji

Zapytanie logujące:

```
POST /ws/doLogin HTTP/1.1
Host: api.example.com
Content-Type: application/json
Accept: application/json

{
 "doLogin": {
  "userLogin": "test",
  "userPassword": "289dff07669d7a23de0ef88d2f7129e7"
 }
}
```

Odpowiedź:

```
{
 "doLoginResponse": {
  "sessionId": "eP3cFozcI3pnyq9wO3Fa7vWg0H7CthI0029736_D"
 }
}
```

Kolejne zapytanie z użyciem sesji:

```
{
 "getProductsInfo": {
  "sessionId": "eP3cFozcI3pnyq9wO3Fa7vWg0H7CthI0029736_D",
  "productList": {
   "product": [
    { "id": "0006VI" },
    { "reference": "144 666" }
   ]
  }
 }
}
```

Cykl życia sesji

- Sesja pozostaje aktywna przez **10 minut od ostatniej aktywności**.
- Maksymalny czas życia sesji wynosi **3 godziny od momentu zalogowania**, niezależnie od aktywności.

- Po wygaśnięciu sesji serwer zwraca błąd z kodem `ERR_SESSION`. W takiej sytuacji należy ponownie wywołać metodę `doLogin`, aby uzyskać nowy identyfikator sesji.
- Webservice nie wymaga jawnego wylogowania.

Limity i ograniczenia

Korzystanie z API objęte jest następującymi limitami i zasadami:

- **Limit jednoczesnych sesji**: domyślnie jedna aplikacja może posiadać maksymalnie **30 aktywnych** sesji. Próba utworzenia kolejnej sesji po przekroczeniu limitu zakończy się błędem.
- **Czas życia sesji**:
  - 10 minut bezczynności (brak żądań),
  - 3 godziny od momentu zalogowania.
- **Limit połączeń jednoczesnych**: każda instalacja systemu posiada określoną liczbę dostępnych jednoczesnych połączeń do API. Wartość tego limitu zależy od konfiguracji środowiska i może być ustalana indywidualnie.
- Przekroczenie dostępnych połączeń może skutkować odrzuceniem kolejnych żądań do czasu zwolnienia zasobów.

Przykład użycia w języku PHP

Poniższy przykład pokazuje, jak w prosty sposób zintegrować się z cairo.API przy użyciu języka PHP. Funkcja `ws_call()` stanowi główny interfejs do komunikacji z WebService, automatycznie obsługując proces logowania, zarządzanie sesją oraz automatyczne ponowne logowanie w przypadku wygaśnięcia sesji (błąd `ERR_SESSION`).

Dzięki temu można wywoływać dowolne metody API bez konieczności ręcznego zarządzania cyklem życia sesji – wystarczy jedno wywołanie funkcji `ws_call()` z nazwą metody oraz parametrami.

```
<?php


function ws_call($method, $params = [])
{
    static $sessionId = null;
    static $host = 'http://127.0.0.1:7888/';
    static $credentials = [
        'userLogin'    => 'test',
        'userPassword' => '289dff07669d7a23de0ef88d2f7129e7', // MD5 z hasła
    ];

    // Wewnętrzna funkcja wykonująca właściwe żądanie HTTP
```

```
$call = function($method, $params) use ($host) {
    $request = [$method => $params];
    $requestStr = json_encode($request);
    $ch = curl_init();
    curl_setopt_array($ch, [
        CURLOPT_URL            => $host,
        CURLOPT_POST           => true,
        CURLOPT_POSTFIELDS     => $requestStr,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_HEADER         => true,
        CURLOPT_HTTPHEADER     => [
            'Content-Type: application/json',
            'Accept: application/json',
        ]
    ]);

    $responseStr = curl_exec($ch);
    $error       = curl_error($ch);
    $headersSize = curl_getinfo($ch, CURLINFO_HEADER_SIZE);
    curl_close($ch);

    if ($error) {
        echo "CURL ERROR: $error\n";
        return null;
    }

    $responseBody = substr($responseStr, $headersSize);
    $response = json_decode($responseBody, true);

    echo "REQUEST:\n$requestStr\n";
    echo "RESPONSE:\n$responseBody\n";

    return $response;
};

// Logowanie przy braku aktywnej sesji
if ($sessionId === null && $method !== 'doLogin') {
    $loginResult = ws_call('doLogin', $credentials);
    if (!isset($loginResult['doLoginResponse']['sessionId'])) {
        echo "Autoryzacja nie powiodła się.\n";
```

```
            return null;
        }
        $sessionId = $loginResult['doLoginResponse']['sessionId'];
    }

    // Dołączenie sesji do parametrów
    if ($method !== 'doLogin') {
        $params['sessionId'] = $sessionId;
    }

    // Wywołanie metody
    $result = $call($method, $params);

    // Obsługa wygasłej sesji i ponowienie logowania
    if (isset($result['error']['code']) && $result['error']['code'] === 'ERR_SESSION') {
        echo "Sesja wygasła, ponawiam logowanie...\n";
        $sessionId = null;
        return ws_call($method, $params); // ponowne wywołanie po re-loginie
    }

    return $result;
}

// ============================
// Przykład użycia metody getProductsInfo
$productList = [
    'product' => [
        ['id' => '0006VI'],
        ['reference' => '144 666']
    ]
];

$response = ws_call('getProductsInfo', ['productList' => $productList]);

if ($response) {
    print_r($response);
} else {
    echo "Wywołanie nie powiodło się.\n";
}
```

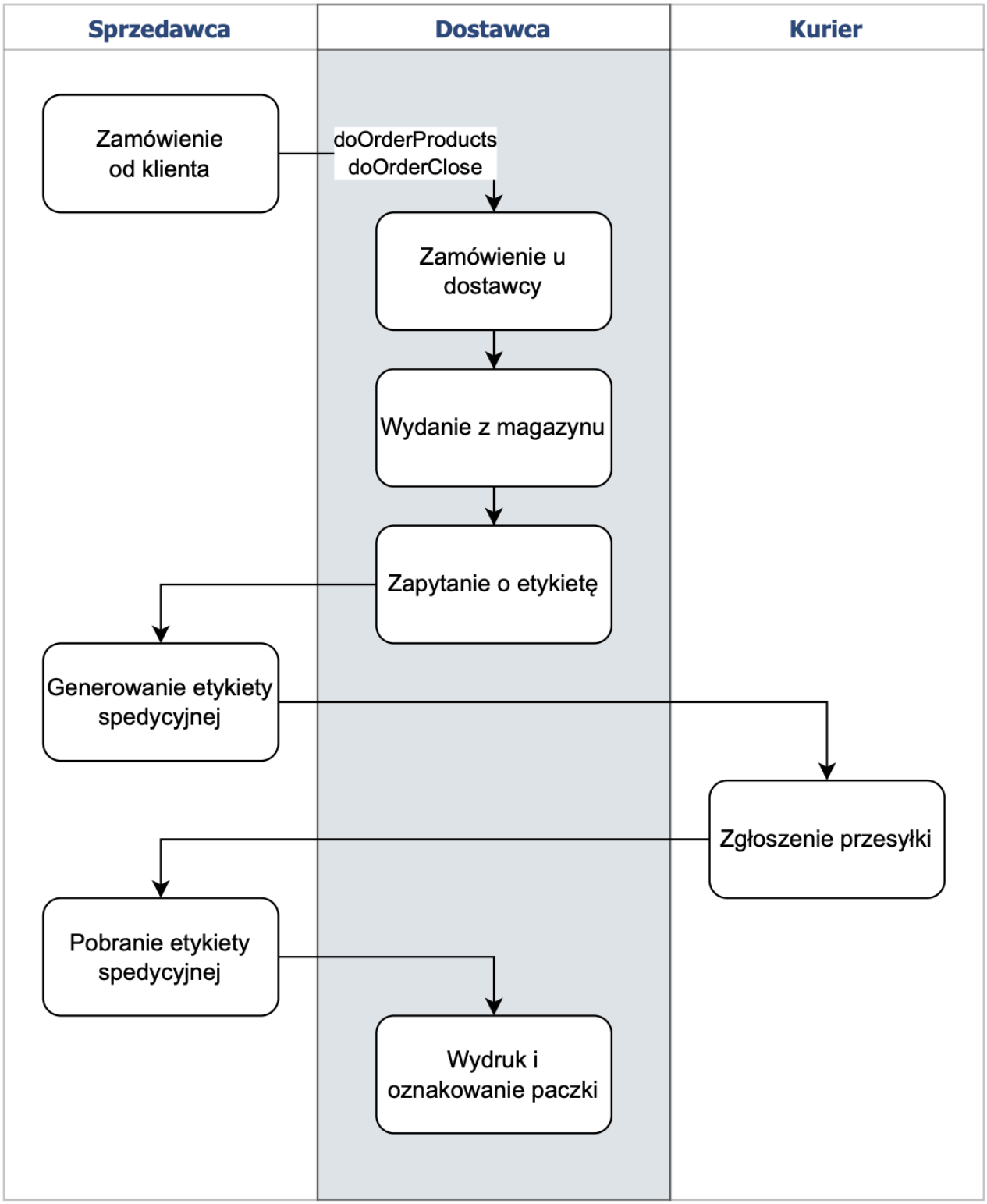
Dropshipping

Dropshipping to model logistyczny, w którym sprzedawca przyjmuje zamówienia, ale nie przechowuje produktów ani ich nie wysyła bezpośrednio. Zamiast tego przekazuje zamówienia do dostawcy, który przygotowuje przesyłkę i wysyła ją do końcowego odbiorcy. W zależności od wybranego scenariusza tego procesu za zgłoszenie przesyłki do firmy kurierskiej może odpowiadać sprzedawca (wówczas dostawca komunikuje się ze sprzedawcą, aby pobrać etykietę na paczę – scenariusz A) lub bezpośrednio dostawca (scenariusz B).

Scenariusz A

W tym układzie sprzedawca musi udostępnić możliwość pobierania etykiet na paczki przygotowane przez dostawcę. W zapytaniu dostawca może przekazać następujące informacje o paczce: nr zamówienia, wymiary, masa, zawartość. W odpowiedzi sprzedawca musi przekazać etykietę w pliku PDF, która umieszczona zostanie na paczce. W zależności od uwarunkowań technicznych mogą występować ograniczenia techniczne odnośnie maksymalnego wymiaru etykiety. W sprawie szczegółów należy kontaktować się bezpośrednio z dostawcą.

> Uwaga! Ta forma procesu wymaga każdorazowo dostosowania systemu dostawcy do rozwiązań informatycznych sprzedawcy, więc wdrożenie go może być bardziej czasochłonne.



Przykład zapytania **doOrderProducts**:

```
{ "doOrderProducts": {
   "sessionId": "*****************************",
```

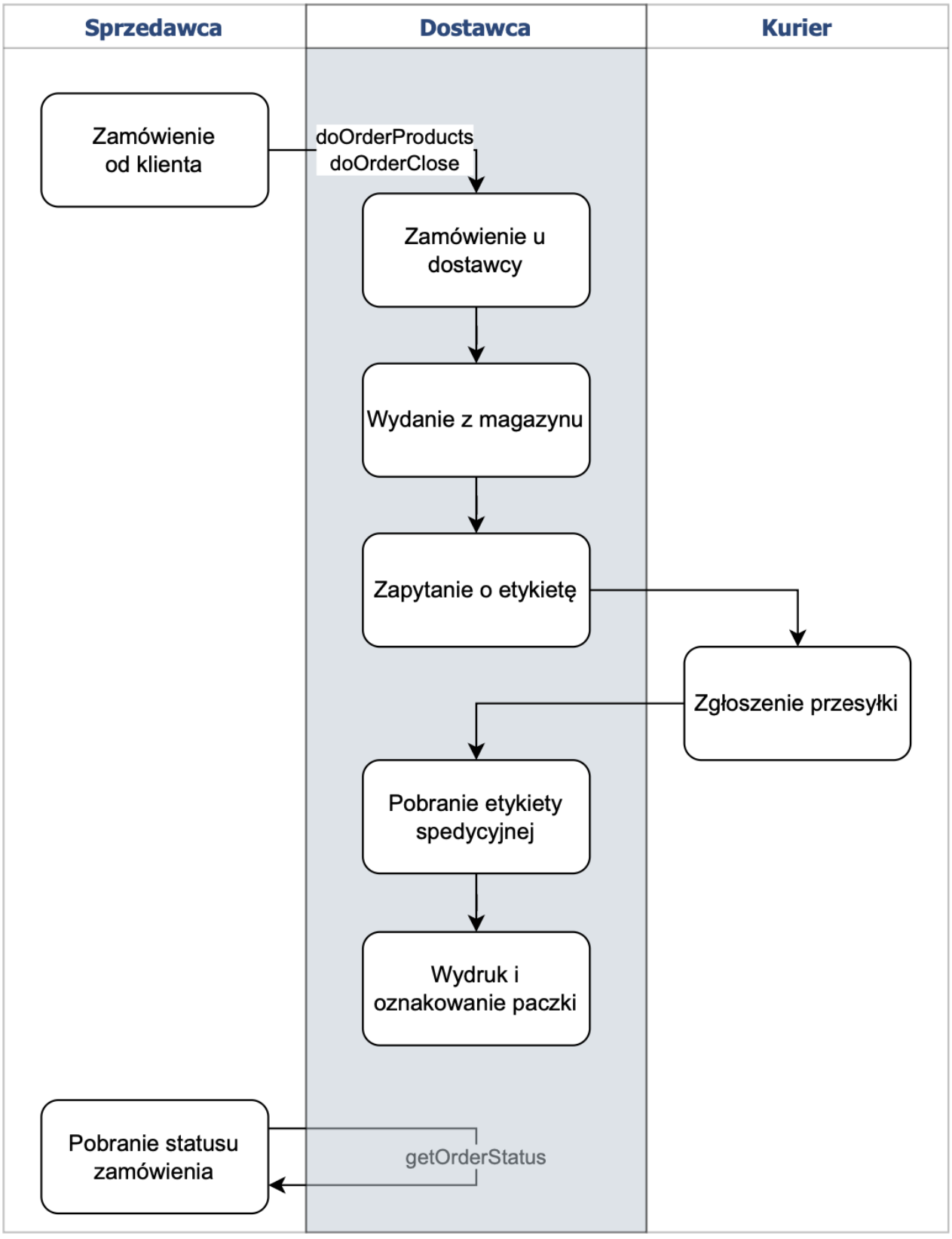
```
    "forceNewOrder": 1,
    "onlyFoundItems": 1,
    "newOrderInfo": {
       "externalId":"**********",
       "routeId":"******"
    },
    "productOrderList": {
       "productOrder": [{
          "tecidd":"350",
          "tecnum":"ADV184326",
          "quantity": 2
       }]
    }
}}
```

Scenariusz B

W tym układzie za wysyłkę paczki do odbiorcy odpowiedzialny jest dostawca. Dane odbiorcy muszą być przekazane wraz z zamówieniem w metodzie doOrderProducts. Dodatkowo w zależności od ustaleń pomiędzy sprzedawcą a dostawcą, wysyłka może być zgłaszana do kuriera danymi dostawcy (wówczas musi to być niezależnie rozliczone ze sprzedawcą) lub sprzedawcy (dane muszą być wprowadzone do systemu dostawcy na etapie konfiguracji połączenia).

W każdym układzie sprzedawca zamawiając towar powinien wskazać trasę – czyli formę dostawy do odbiorcy – wraz z zamówieniem (metoda doOrderProducts).

> Uwaga! Zakładając, iż korzystamy z kurierów już zintegrowanych w systemie cairo.WMS wdrożenie komunikacji w tej formie nie wymaga modyfikacji systemu a jedynie jego odpowiedniej konfiguracji.



Lista metod sugerowanych do wykorzystania w Dropshipping:

- doLogin

- getProductsInfo
- doOrderProducts
- doOrderClose
- getOrderStatus
- getMyInvoices
- getMyRoutes

Przykład zapytania **doOrderProducts** dla wysyłki kurierem:

```
{ "doOrderProducts": {
   "sessionId": "*****************************",
   "forceNewOrder": 1,
   "onlyFoundItems": 1,
   "newOrderInfo": {
      "deliveryAddress": {
         "name":"**************",
         "street":"******************",
         "postcode":"*******",
         "city":"********************",
         "country":"**",
         "phone":"*****************",
         "email": "**********************"
      },
      "externalId":"***********",
      "routeId": "******"
   },
   "productOrderList": {
      "productOrder": [{
         "tecidd":"350",
         "tecnum":"ADV184326",
         "quantity": 2
      }]
   }
}}
```

Przykład zapytania **doOrderProducts** dla wysyłki do punktu odbioru:

```
{"doOrderProducts": {
   "sessionId": "*****************************",
   "forceNewOrder": 1,
   "newOrderInfo": {
```

```
      "pickupPointAddress": {
         "name": "***************",
         "street":"********************",
         "postcode": "**************",
         "city": "****************",
         "country": "*****",
         "phone": "********************",
         "email": "******************************",
         "pickupPointId": "**********************"
      },
      "externalId":"208755449",
      "routeId":"******"
   },
   "productOrderList": {
      "productOrder": [{
         "reference": "VK22 DENSO",
         "quantity": 4
      }]
   }
}}
```

Dokumentacja

Dokumentacja dostępna jest na stronie:

https://api-ws.cairo.pl