cairo.API

- Basic information
- Dropshipping
- Documentation

Basic information

cairo.API

The cairo.ERP system provides its clients with API functionality based on JSON communication over the HTTP `POST` method. Unlike classical REST architecture, access to functions is done by invoking named operations passed in the request body. All data — both input and response — is transferred in JSON format.

The API allows parameters (such as session identifiers) to be sent in the request body, eliminating the need to pass them in the URL or headers.

> XML communication is no longer supported

All communication is done through a single endpoint:

POST https://api.example.com/ws/<method name>

Example: retrieving system version:

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


{
 "getVersion": {}
}
```

Response:

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

```
}
```

Request structure

Every request should follow a consistent structure:

```
{
 "<operationName>": {
  <inputParameters>
 }
}
```

Example:

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

Response structure

A valid response:

```
{
 "<operationName>Response": {
  <outputData>
 }
}
```

Example:

```
{
 "getVersionResponse": {
```

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


Error handling

In case of an error, the server returns an HTTP 500 response containing an `error` object in JSON format:


```
{
  "error": {
    "code": "ERR_SESSION",
    "msg": "Invalid session identifier"
  }
}
```


Communication protocol

Access to the API is provided via an HTTP(S) connection to the server. All requests must include the header: `Content-Type: application/json; charset=utf-8`. Communication is stateless — sessions are identified via parameters passed in the request body.

API access

To access the API, you need to know the target host and obtain API credentials. Please contact your account manager to obtain access data.

Language version

By default, all messages returned from the API are in the language configured in the cairo.ERP system. To receive responses in a specific language, include the appropriate `languageId` during the `doLogin` method call.

Session handling

Accessing API methods requires a successful login. Authentication is performed via the `doLogin` method, which requires a login and an MD5-encoded password. The `doLogin` method returns a unique session key (`sessionId`), which must be included in every subsequent API request.

Example: login and session usage

Login request:

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

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

Response:

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

Subsequent request using session:

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

Session lifecycle

- A session remains active for **10 minutes of inactivity**.
- The maximum lifetime of a session is **3 hours from login**, regardless of activity.

- After expiration, the server returns error code `ERR_SESSION`. In such cases, the doLogin method must be called again to obtain a new session key.
- Explicit logout is not required.

Limits and restrictions

The API is subject to the following limits and constraints:

- **Concurrent sessions**: By default, a single application may have up to **30 active sessions**. Attempts to create additional sessions beyond this limit will result in an error.
- **Session timeout**:
  - 10 minutes of inactivity,
  - 3 hours from login (absolute timeout).
- **Concurrent connections**: Each system installation has a limit on the number of concurrent API connections. The value of this limit depends on the system configuration and may be set individually.
- Exceeding available connections may result in requests being rejected until resources are released.

PHP example

The following example demonstrates how to easily integrate with cairo.API using PHP. The `ws_call()` function serves as the main interface for WebService communication. It automatically handles session management, login, and session renewal in the event of an `ERR_SESSION` error.

Thanks to this, you can call any API method without manually managing session state — just call `ws_call()` with the method name and parameters.

```
<?php


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


   // Internal request function
   $call = function($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;
};

// Login if no session yet
if ($sessionId === null && $method !== 'doLogin') {
    $loginResult = ws_call('doLogin', $credentials);
    if (!isset($loginResult['doLoginResponse']['sessionId'])) {
        echo "Authentication failed.\n";
        return null;
    }
```

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

    // Add sessionId to params
    if ($method !== 'doLogin') {
        $params['sessionId'] = $sessionId;
    }

    // Perform request
    $result = $call($method, $params);

    // Handle session expiration
    if (isset($result['error']['code']) && $result['error']['code'] === 'ERR_SESSION') {
        echo "Session expired, retrying login...\n";
        $sessionId = null;
        return ws_call($method, $params); // retry after re-login
    }

    return $result;
}

// ============================
// Example usage: getProductsInfo
$productList = [
    'product' => [
        ['id' => '0006Vl'],
        ['reference' => '144 666']
    ]
];

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

if ($response) {
    print_r($response);
} else {
    echo "Request failed.\n";
}
```

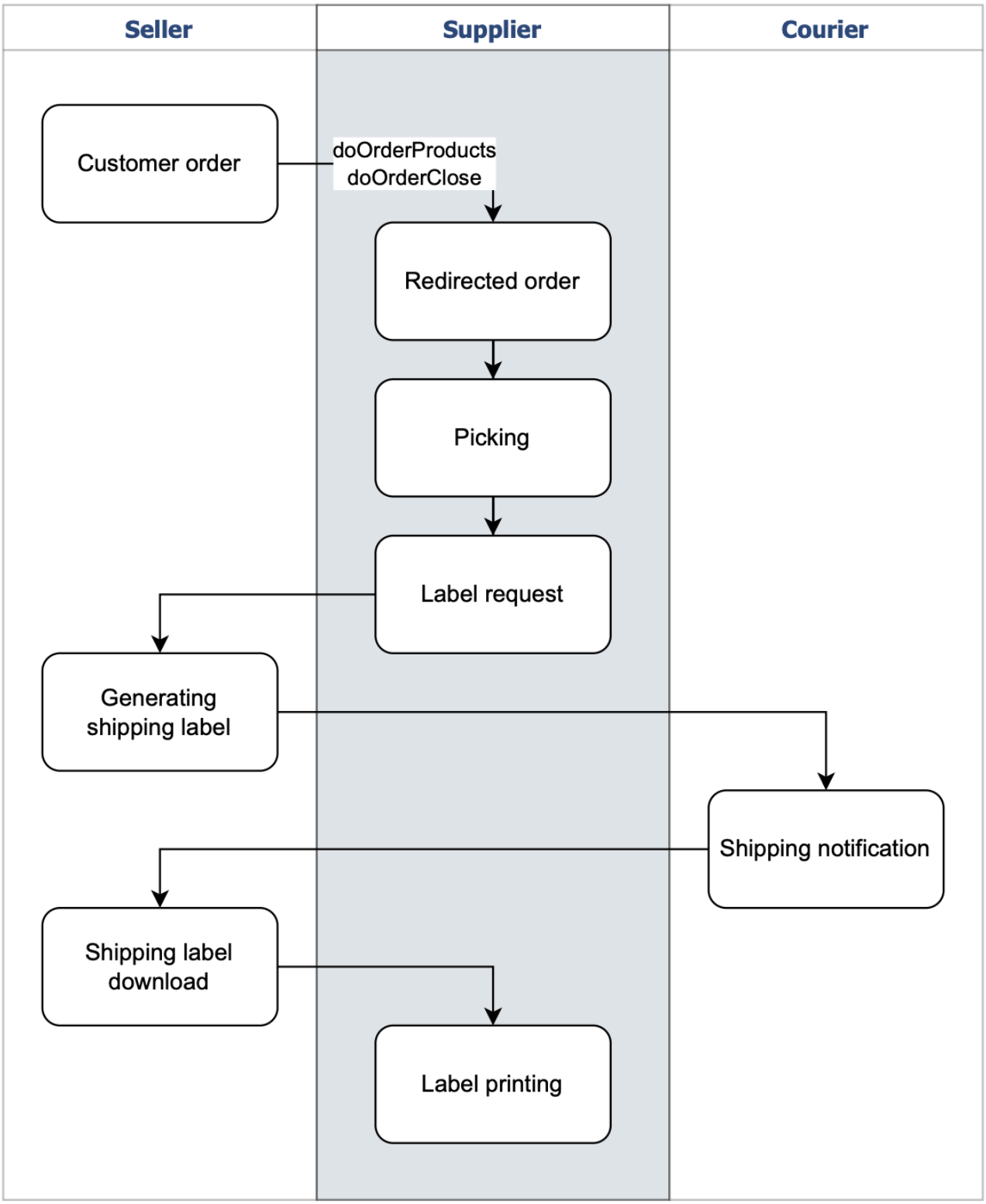
Dropshipping

Dropshipping is a logistics model in which the seller accepts orders but does not store the products or ship them directly. Instead, the orders are forwarded to the supplier, who prepares the shipment and sends it to the final recipient. Depending on the chosen scenario of this process, the seller may be responsible for reporting the shipment to the courier company (then the supplier communicates with the seller to download the label for the package - scenario A) or directly the supplier (scenario B).

Scenario A

In this arrangement, the vendor must provide the option of downloading labels for packages prepared by the supplier. In the query, the supplier can provide the following information about the package: order number, dimensions, weight, content. In response, the vendor must provide a label in a PDF file, which will be placed on the package. Depending on technical conditions, there may be technical restrictions regarding the maximum size of the label. For details, please contact the supplier directly.

> Note! This form of process requires each time the supplier's system to be adapted to the vendor's IT solutions, so its implementation may be more time-consuming.



Example of a **doOrderProducts** query:

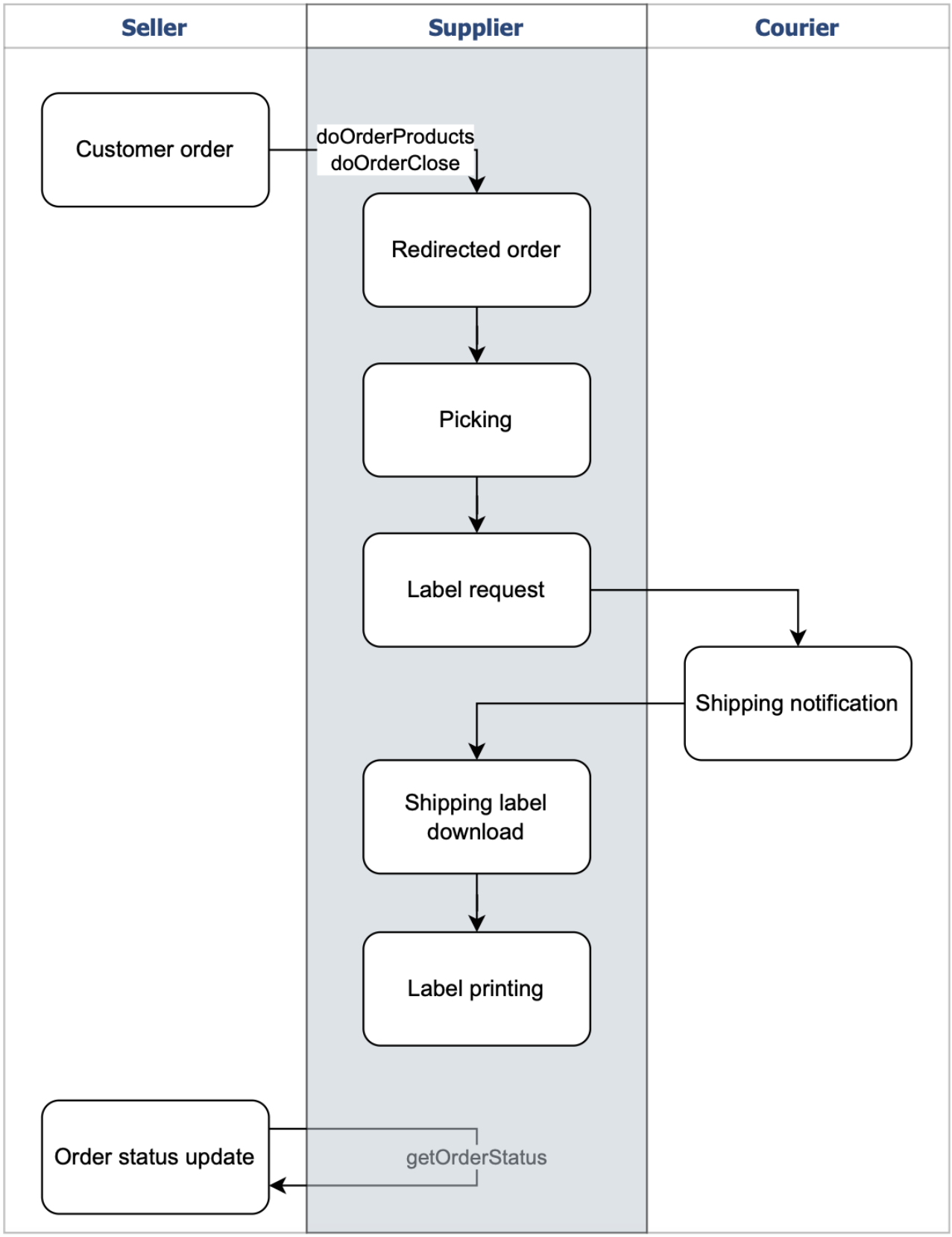
```
{ "doOrderProducts": {
   "sessionId": "*****************************",
```

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


Scenario B

In this arrangement, the supplier is responsible for sending the package to the recipient. The recipient's data must be transferred together with the order in the doOrderProducts method. Additionally, depending on the arrangements between the seller and the supplier, the shipment can be reported to the courier with the supplier's data (then it must be settled independently with the seller) or the seller (the data must be entered into the supplier's system at the connection configuration stage). In each arrangement, when ordering the goods, the seller should indicate the route - i.e. the form of delivery to the recipient - together with the order (doOrderProducts method).

> Note! Assuming that we are using couriers already integrated in the cairo.WMS system, implementing communication in this form does not require modification of the system, only its appropriate configuration.



List of methods suggested for use in Dropshipping:

- doLogin

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

Example of a **doOrderProducts** query for courier delivery:

```
{ "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
      }]
   }
}}
```

Example of a **doOrderProducts** query for shipping to a collection point:

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

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

Documentation

Documentation is available at:

https://api-ws.cairo.pl