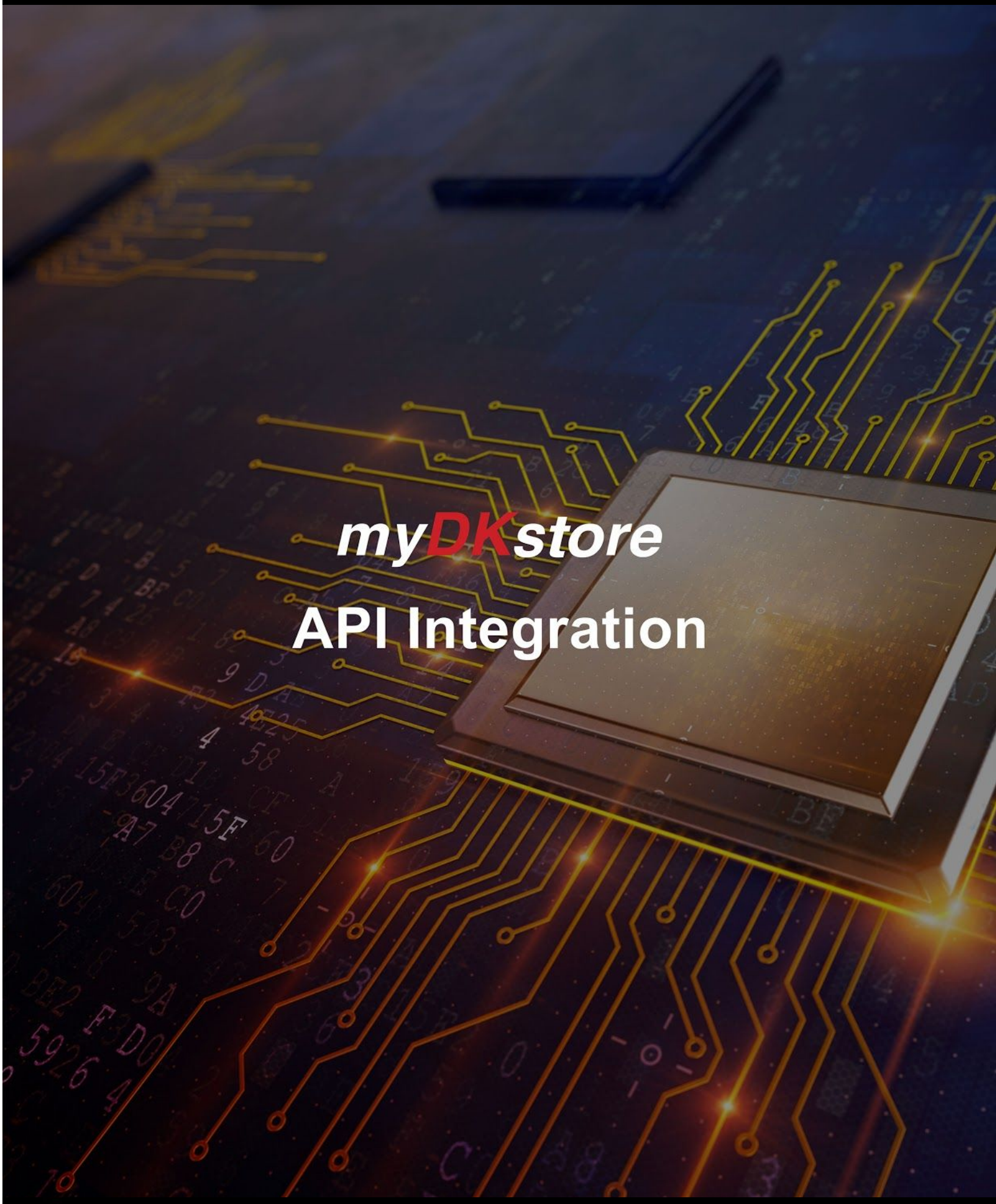


myDKstore API Integration

Contents

About Seller API

This topic explains the basics about integrating with Digi-Key seller API. It also outlines the key processes for:

- updating stock and prices for offers
- managing incoming and outgoing orders
- managing customer service using the messaging service

Automate all your Marketplace activity using Digi-Key API collection

The purpose of this guide is to present the APIs integration method for sellers. We will go through the main interactions between sellers systems and the Marketplace. In addition to this document you will have access to a technical documentation to build the APIs integration with your IT system. This integration method requires some technical resources.

What is an API?

An API (Application Programming Interface) is an interface provided by a computing program. It allows independent programs to interact with each other. In other words it enables sellers to seamlessly manage their Marketplace activity with their overall e-commerce activity.

Why use the API integration method?

Thanks to Digi-Key Seller APIs, sellers can automate their catalog, their orders and customer care management. It will increase their sales efficiency with no additional charge on the teams. It can also improve their overall quality of service by decreasing customer response time.

Do we have to automate everything?

APIs are made to ease Marketplace daily operations. Depending on your level of autonomy you can automate part or all of your activity on the Marketplace. For example, you can automate offers and orders but upload products manually with an Excel file.

Global view on APIs flows

Digi-Key Seller APIs allow you to automate 4 main parts of your activity on the Marketplace:

- Store information
- Catalog offers
- Orders management & shipping information
- Accounting management

You are free to integrate part or all the APIs related to these 4 topics.

API integration prerequisites

Choosing the APIs and the strategy of integration

Digi-Key provides different automation methods, you can use either one depending your IT system:

-If you are using Magento 1...

Digi-Key has developed a connector that will enable you to automate your products and offers very rapidly. This integration does not require any technical skills.

Note that the order management is not available yet in the connector. If you want to automate your orders management, you can combine this method with one of the following methods.

Ask for the Magento 1.9 Connector Documentation from DKE Marketplace.

-If your e-commerce solution is coded in PHP or Java...

You can access the Digi-Key SDK via the "Connectors and APIs" section of this documentation set. The SDK is a set of programming support tools that will help you integrate flows and Digi-Key APIs more easily and rapidly, whilst decreasing your margin of error in your regular flows sent out.

Retrieve and get information on how to use the Digi-Key PHP SDK or the Java SDK.

-If your e-commerce solution is not Magento or is not coded in PHP or Java...

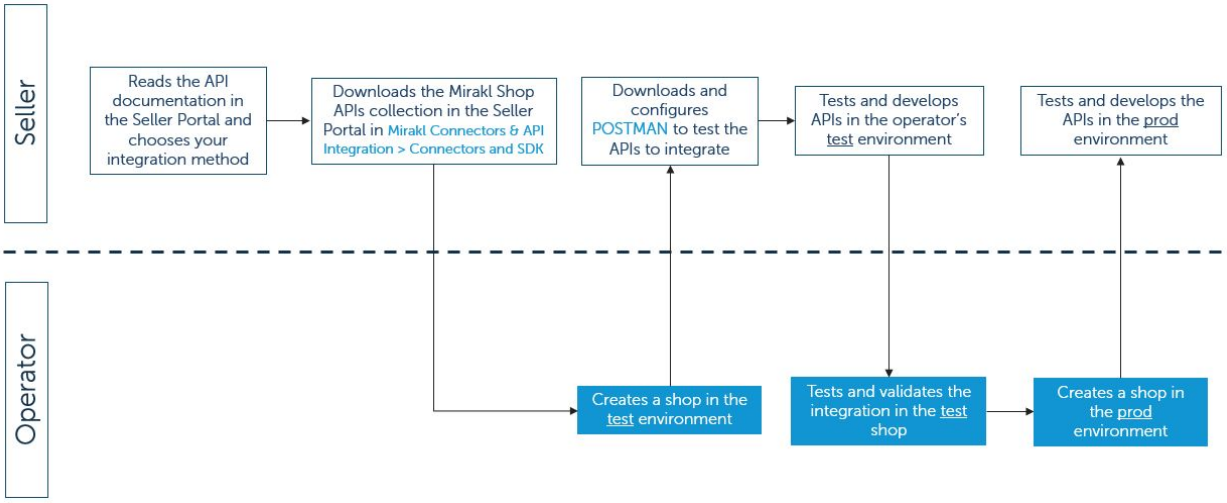
You will have to make all the developments to integrate the APIs yourself.

> The last two methods require development skills.
>
> If you do not have an IT department or if you do not want to use an external resource, you can opt for another integration method: manual, through file or FTP. Ask DKE Marketplace for advice.

Getting the technical documentation

If you are interested in integrating with Digi-Key APIs, you can access the API documentation and get the API collection. The documentation contains everything you need to know for authentication, response types and codes, accepted formats and the detailed input and output for every API.

Understanding the integration process



Getting your API key

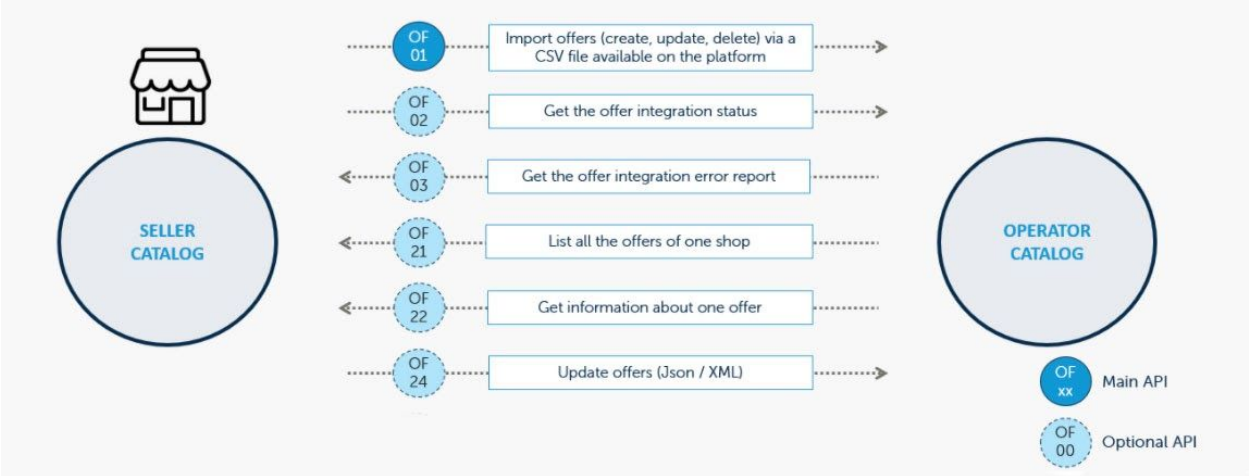
For more information, refer to: Generating your store API key.

Digi-Key has 2 environments: 1 test environment and 1 production environment.

As a seller, you will first integrate your APIs on the test environment. Then, once your integration is validated, you will repeat the integration on the production environment.

An API Key is specific to an environment. You will have to generate an API key for the test environment and then, one for the production environment.

Automate your store management



Seller account management API

The store management API A01 allows you to retrieve all seller account information.

GET A01 - Get shop information

URL	`/api/account`
DESCRIPTION	Get shop information Recommended: Once per day Maximum: Once per day
QUERY PARAMETERS	shop_id (*integer - int64*) [optional] - Use this parameter when your user has access to several shops. If not specified, the shop_id from your default shop will be used.
PRODUCES	application/json application/xml
HTTP RETURN CODES	200 - OK

Seller account management API

API	Description	Use case(s)
A01	Get shop information.	You want to display your account information (useful for full API integrated back office or mobile applications for example).

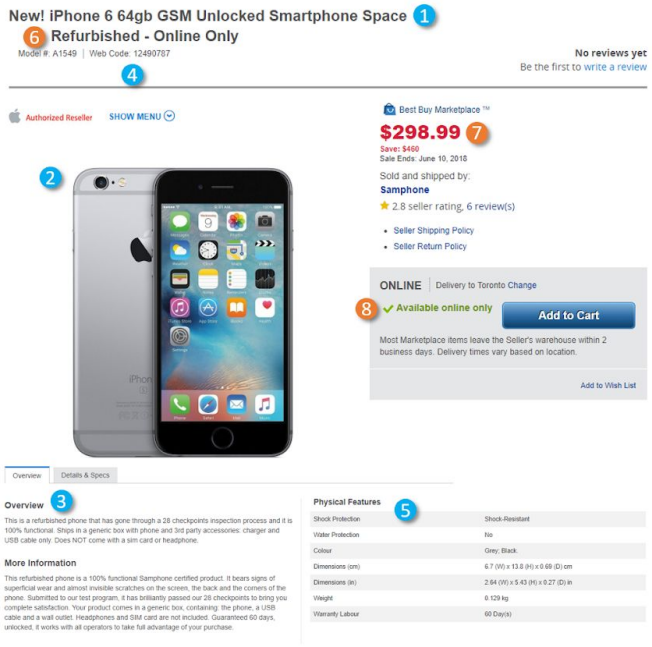
Store document management APIs

Documents API

API	Description	Use case(s)
S30	List shop's documents.	You want to get the list of all the documents uploaded by a shop.
S31	Download one or multiple shop's documents.	You want to download one or several documents uploaded by a shop to do the KYC process.
S32	Upload documents to attach to a shop.	You want to upload a document related to a shop.
S33	Delete a shop document.	You want to automate document deletion to comply with the General Data Protection Regulation

Automate your catalog management

Difference between "Product" and "Offer"



Product

Product characteristics cannot change from one seller to another:

1. Product Name
2. Image(s)
3. Description
4. Product_ID / EAN / UPCs / ISBN...
5. Size / Color / Dimensions

Offer

Offer characteristics can vary from one seller to another:

6. A condition (new, occasion, refurbished...)
7. A price
8. Stock quantity availability

Product management APIs

Product creation API

API	Description	Use case(s)
P41	Import products to the seller portal.	You want to import products to the seller portal.
P42	Get the import status for a product import.	You want to track the status of a product import after an API P41 call.
P51	Get information about product import statuses.	You want to integrate products from the seller portal to your information system using API only.
P44	Get the error report file for a product import.	You want to get the error report about a product import.
P45	Get the product integration report file for a product import.	You want to get the product integration error report.
P46	Get the transformed file for a product import.	You want to retrieve the transformed file from a product import.
P47	Get the transformation error report file for a product import.	You want to download a product import error report.

Catalog structure API

API	Description	Use case(s)
CM11	Export product data status from Source Product Data Sheets.	Exporting product data statuses will list out products that are live and products that are waiting on required information.
H11	List Catalog categories (parents and children).	Retrieve category codes.
PM11	Get product attribute configuration.	Retrieve list of attributes.
VL11	Get Digi-Key's value lists.	Retrieve list of codes.

Front-end API

API	Description
P11	List offers for each given product.

Offer and product upload

You can upload offers and products at the same time when using API OF01.

To do so, use "with_products=true" in the API request.

Offer management APIs

You can download the offer file from your back office: **My Inventory > Import from File > Download an Excel file template for offers**.

Offer management: APIs details

Offer management API

API	Description	Use case(s)
OF01	Import a file to add offers.	You want to create or update offers in the seller portal.
OF02	Get information and statistics about an offer import.	You want to track the status of an offer import an API OF01 call.
OF03	Get the error report file for an offer import.	You want to get an error report about an offer file import.
OF21	List offers of a shop.	You want to retrieve the inventory of the offers you created.
OF22	Get information about an offer.	You want to get information about an offer.
OF24	Create, update, or delete offers.	You want to create an offer or update an offer you already created (XML and JSON formats accepted).

About OF01 import

If you want to import your offers using API OF01, the CSV file (encoded in UTF-8 format) must comply with a specific format defined by Digi-Key.

For more information about the format for importing offers, refer to: Which on-boarding method should I use?

Here is a CSV template for OF01 import of offers.

"sku";"product-id";"product-id-type";"description";"internal-description";"price";"price-additional-info";"quantity";"min-quantity-alert";"state";"available-start-date";"available-end-date";"discount-price";"discount-start-date";"discount-end-date";"discount-ranges";"allow-quote-requests";"leadtime-to-ship";"min-order-quantity";"max-order-quantity";"package-quantity";"update-delete";"price-ranges";"returns-accepted"

Updating Offers

If you want to update offers, you must provide for each offer:

- the offer SKU
- the fields containing data you want to update

Offer file (API OF01) field description

Field	Description	Status	Accepted Value	Example
sku	The offer's unique identifier in the store	**Required**	Character string limited to 255 characters. The character '/' is forbidden	OFFER_SKU_1
product-id	Unique product identifier for a given product-id-type	**Required when creating an offer**	Generated by Digi-Key.	Article_123456791234
product-id-type	Type of product-id identifier. Refer to: Product reference identifiers.	**Required when creating an offer**	Value in uppercase: SKU (product SKU), ISBN, UPC, EAN, SHOP_SKU	SKU
description	Offer description	Recommended	Limited to 2000 characters	New product, original packaging
internal-description	The offer's description as shown in the back office	Optional	Limited to 2000 characters	-
price	The unit selling price	**Required when creating an offer**	Decimal number; a period is used to separate cents	45.50
price-additional-info	Information regarding the price of the offer. Will not display on the web.	Optional	Character string limited to 100 characters	"Environmental contribution (recycling fee) of 10 cents"
quantity	The quantity available in stock (maximum: one billion)	**Required**	Integer greater than or equal to 0	100

min-quantity-alert	The minimum stock level that triggers an email alert No alerts are sent if this field is not specified.	Optional	Integer greater than or equal to 0	10
state	The state code of the offer	**Required when creating an offer**	The accepted values are defined in the back office view	See available values in the back office: **Information > Offer Conditions**
available-start-date	The first day the offer becomes available. The offer has no start date if blank.	Optional	yyyy-MM-dd	2017-11-27
available-end-date	The last day the offer is available. The offer has no end date if blank.	Optional	yyyy-MM-dd	2017-03-16
logistic-class	The code for the logistics class. This logistic class overwrites the default logistic class assigned to the offer.	Optional	Values: FRS, S, M, L, XL	FRS
discount-start-date	The first day the discount becomes available. If blank, discount starts immediately.	Optional (Not an enabled feature at this time)	yyyy-MM-dd	2017-11-27
discount-end-date	The last day the discount is available. The	Optional (Not an enabled feature at this	yyyy-MM-dd	2017-12-27

	discount has no end date if blank.	time)		
discount-ranges	The discount quantity thresholds and prices to define discount ranges	Optional (Not an enabled feature at this time)	quantityThreshold1|price1, quantityThreshold2|price2	CSV format: 2|9.00,10|8.50
leadtime-to-ship	The calendar days it takes from the time the order is accepted to when the order is shipped.	Optional (If left blank, default value set to 5)	Integer greater than or equal to 0 and less than 60	5
min-order-quantity	The minimum quantity customers must put in their basket for one order	Optional	Integer between 1 and 1 000 000 000 or between 1 and max-order-quantity (if activated)	3
max-order-quantity	The maximum quantity customers must put in their cart for one order	Optional	Integer between 1 and 1 000 000 000	20
package-quantity	The increment number of product items for one order	Optional	Integer between 1 and 1 000 000 000 or between 1 and max-order-quantity (if activated)	5
update-delete	Used only in "Normal" import mode. **Update mode is used if blank.**	Optional	"", "update", "delete"	update
price-ranges	The ranges of the prices when the "Volume	Optional	quantityThreshold1|price1, quantityThreshol	CSV format: 1|12, 5|10

	Pricing" feature is activated		d2|price2	
returns-accepted	Returns accepted	**Required when creating an offer**	Values: TRUE, FALSE	TRUE

Offer and product upload

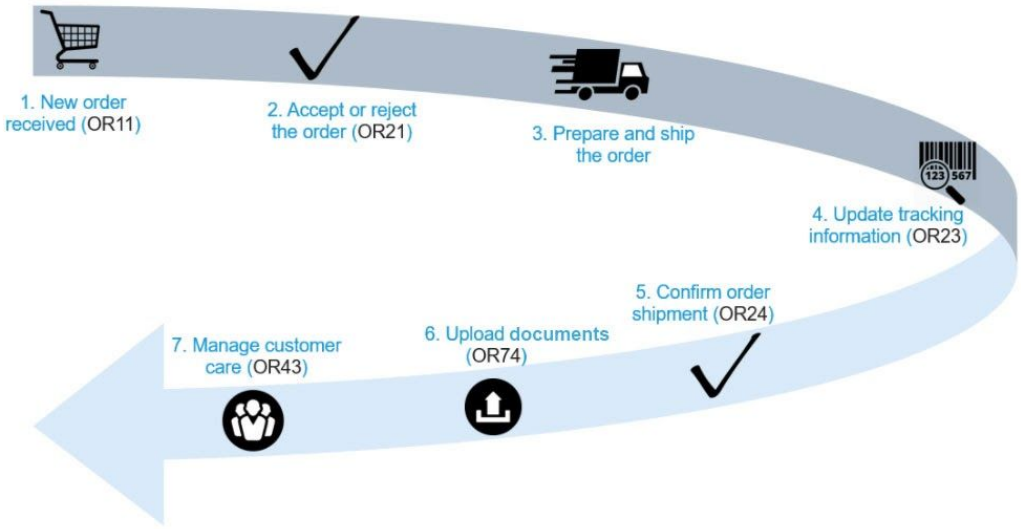
You can upload offers and products at the same time when using API OF01.

To do so, use "with_products=true" in the API request.

Automate your order management

The orders APIs allow to process the entire life cycle of Marketplace orders from receiving new orders to treating and shipping orders, and keeping track of the order's status.

The order cycle on a Marketplace



Order management APIs

If you do not want to automate your order management, you will still be able to track and process your orders from your Digi-Key seller back office.

Order management API

API	Description	Use case(s)
OR11	List orders.	You want to view orders to process (as a seller, useful to accept, track offers, and so on).
OR21	Accept or refuse order lines.	You want to accept or refuse order lines from an order.
OR23	Update carrier tracking information for a specific order.	You want to provide carrier information for each order of an order to process.
OR24	Validate the shipment of an order.	You want to confirm the shipment of an order.
OR28	Request refunds on order lines.	You want to create a refund on an order line.
OR29	Cancel an order not yet debited to the customer.	You want customers to be able to cancel an order for which the debit has not been confirmed yet.
OR31	Update the custom fields of an order and its order	You want to modify the value of a custom field to add information for an existing order line: serial number, number of

	lines.	packages, comment about the delivery, and so on.
OR51	Get the evaluation of an order.	You want to get an order review.
OR72	List order's documents.	You want customers to be able to view a list of documents associated with the order if previously provided: invoice, delivery slip, and so on.
OR73	Download one or multiple documents attached to one or multiple orders.	You want customers to be able to download a document associated with an order if previously provided: invoice, delivery slip, and so on.
OR74	Upload documents to attach to an order.	You want to upload documents associated with an order for your customers: invoice, return voucher, user guide, and so on.
OR75	List all the order taxes available on the platform.	You want to retrieve taxes from the platform.
OR76	Delete an order document.	You want to automate document deletion through API You want to automate document deletion to comply with the General Data Protection Regulation Payment Service Providers can delete documents from the platform when the KYC process is over

Shipping APIs

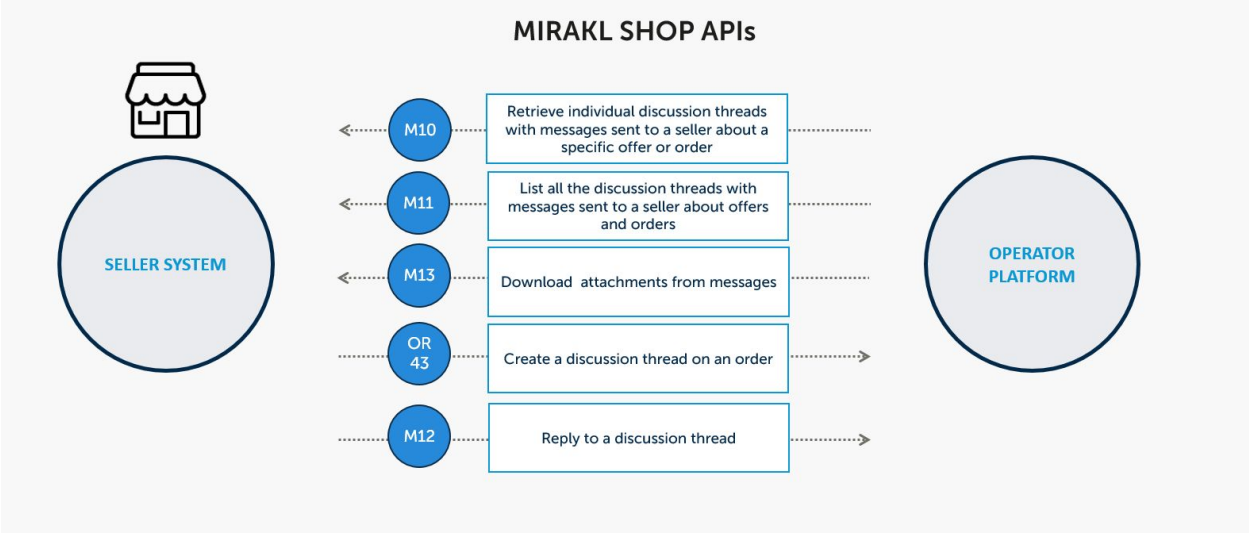
Use shipping APIs to list shipping zones and carriers.

Shipping API

API	Description	Use case(s)
SH11	List all active shipping zones.	You want to list all the active shipping zones in the seller portal.
SH21	List all carriers.	You want to get the list and codes of registered carriers.
SH31	List logistic classes.	You want to map logistic classes with offers at the offer level (API OF24).

Automate your after-sales service

Messaging APIs enable sellers to communicate with customers and answer order-related questions or questions about offers on the Marketplace. Messages can be sent to Digi-Key, the customer or both.



Messaging & customer care APIs

API	Description	Use case(s)
OR43	Create a thread on an order.	You want your customers to be able to create a discussion thread about an order.
M10	Retrieve one discussion thread.	You want your customers to be able to view the messages they sent or received in a discussion thread about a specific order or offer.
M11	List all discussion threads.	You want your customers to be able to view all the discussion threads with messages they sent about a specific offer or order.
M12	Reply to a discussion thread.	You want your customers to be able to reply to messages in discussion threads about an order or offer.
M13	Download an attachment.	You want your customers to be able to download attachments from messages about order or offers.

Automate your accounting

Use invoicing APIs to list and download Digi-Key's invoice (commission charges and monthly subscription).

You can retrieve all your invoices, credit notes and transaction history in your Digi-Key seller back office.

Invoicing API

API	Description	Use case(s)
IV01	List accounting documents.	You want to record turnover on commissions and subscriptions in your accounting.
IV02	Download an accounting document.	You want to download the invoice documents (.pdf) for your accounting records.

Generating and using your store API key

To use and test API flows (in Postman for example), you need your store API key. **Your store API key is strictly confidential.

Generating your store API key

If there is no store API key generated yet on your platform:

1. In the back office menu bar, in the top right corner, click your **user name**.
2. Click the **API key** tab.
3. Click **Generate a new key**.
4. Click **Validate.**

Getting your store API key

If there is already a store API key generated on your platform:

1. In the back office menu bar, in the top right corner, click your **user name**.
2. Click the **API Key** tab.
3. Click **Copy to clipboard**.

Digi-Key copies your API key in your clipboard.

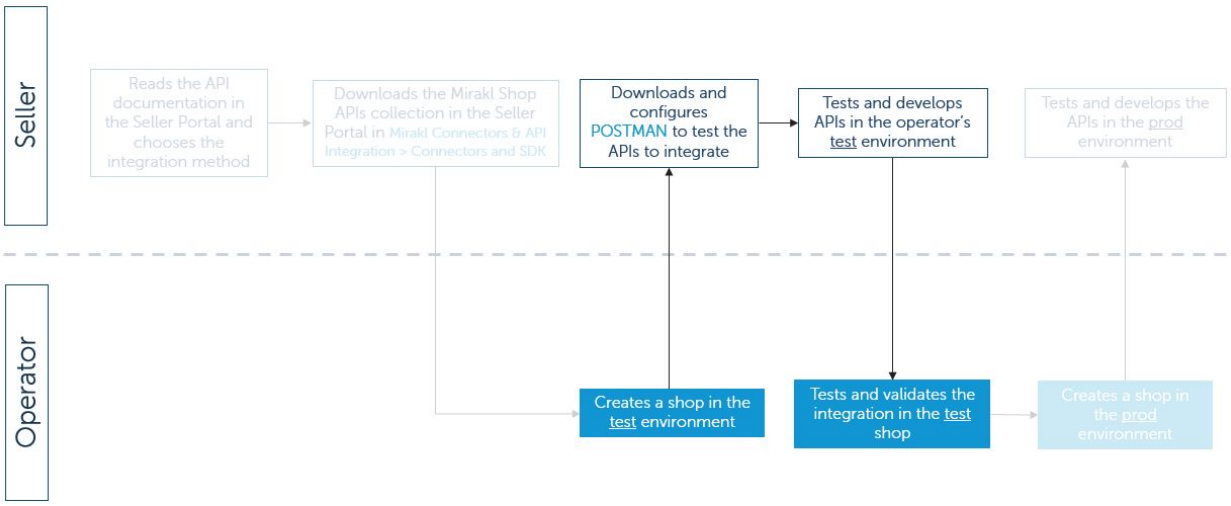
You can now paste your key where you need to use it.

Testing your API integration

Testing your API using Postman

After reading the API documentation and defining your integration strategy, you can make tests with Postman.

Test the main APIs that allow you to automate your activity on the Marketplace.



For more information, refer to: Installing Postman to simulate API calls

Examples of tests

You can test your integration with Postman, with the help of Digi-key's test environment.

Here are some basic actions related to your activity on a Marketplace that you can automate:

1. Before importing your catalog, check:
    a. the operator category tree, thanks to the API H11,
    b. the mandatory and optional attributes related to the categories with PM11,
    c. and the value lists associated with VL11.
2. Adapt your product format consequently. Then, proceed to the mapping with the operator product format, in our back office: **My Inventory > Import from File > Import products in your format > Open the configuration wizard**.
    a. Import our catalog thanks to P41.
    b. Download the information related to the import as well as the error report with P42 and P44.
3. Import your offers with OF01:
    a. Download the operator offer format in the back office: **My Inventory > Import from File > Create or update my offers (stock, price, and so on) > Download an Excel file template for offers**.
    b. Download the information related to the import as well as the error report with OF02 and OF03.
4. Once your catalog has been correctly imported, ask Digi-Key to create a test order in your store.
5. Get the order with OR11, accept it with OR21, and confirm shipping with OR24.
   You can download, then upload the documents related to one or more orders thanks to OR73 and OR74.
6. Ask Digi-Key to send a test message on the order.
7. Get the message with M11, and answer to it with M12.
8. Ask Digi-key to create a test incident on this order.
9. Proceed to the full or partial reimbursement of this same order, thanks to OR28.

All the Shop APIs are not described in this example. See the technical documentation to see all the available APIs.

Reminder: You can automate part or all your activity on the marketplace. You can choose the APIs you want to integrate.

Error reports on products and offers imports

For more information, refer to: Import troubleshooting FAQ

Postman: Simulating API Calls

The integration with Digi-Key is mainly carried out using our API.

By default, Digi-Key relies on secure channels, which is why the network corresponding to this channel is the protocol HTTPS (with port 443).

Installing Postman to test Digi-Key API

Digi-Key does not develop Postman

Postman is a third-party software that is not developed by Digi-Key.

We provide this user guide and the Digi-Key collections to import to Postman to help you integrate and test our API.

Digi-Key does not provide support on Postman.

About Postman

Postman is an HTTP client to help test web services easily and efficiently.

Use Postman to craft both simple and complex HTTP requests. It also saves requests for future use.

To use Postman, you need to use Google Chrome as your web browser.

Installing Postman

1. In Chrome, follow this link.
2. Click **Download** and follow the installation instructions.
3. Start Postman.

Using Postman to test Digi-Key API

Postman is an HTTP client to help test web services easily and efficiently.

Use Postman to craft both simple and complex HTTP requests. It also saves requests for future use.

To use Postman, you need to use Google Chrome as your web browser.

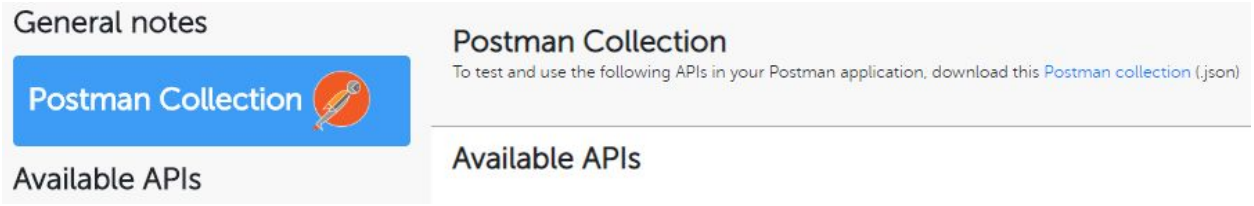
To get the latest postman collections:

1. Go to the API documentation page.
2. Select your Digi-Key product where you are redirected to the API documentation.

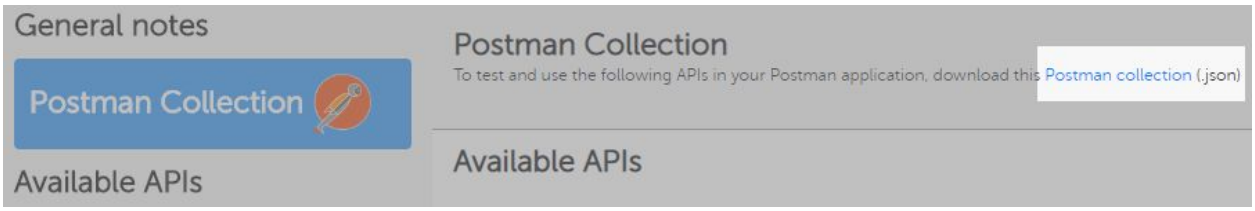
Marketplace for Products Seller API

Marketplace for Services Seller API

3. Click the **Postman Collection** tab.



4. Click the **Postman collection** link.



   Digi-Key downloads the file to your system.

5. Import the file into Postman.
6. Start testing.

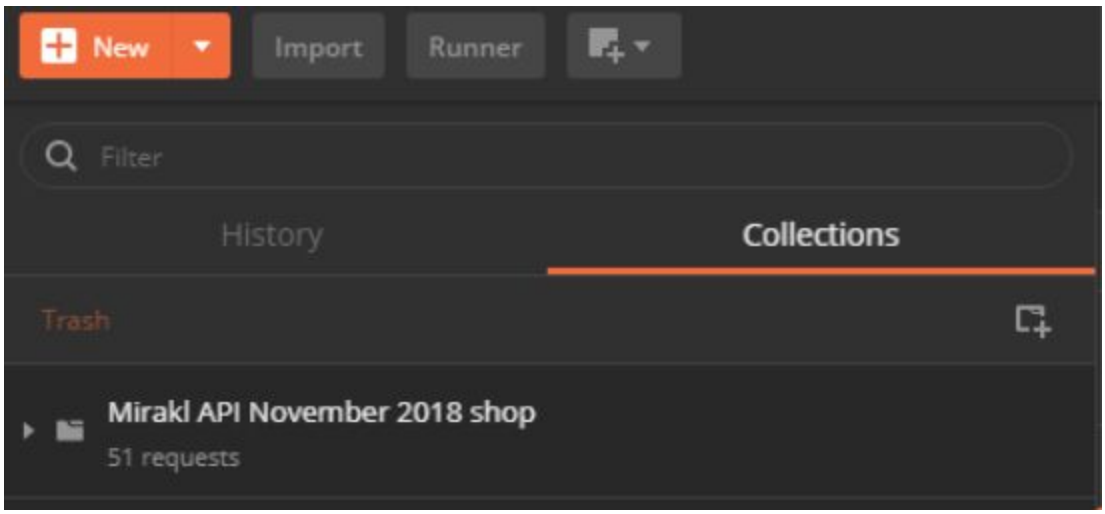
For more information about how to download and use Postman, refer to: Installing Postman to simulate API calls.

Importing the Digi-Key API library

You can now create your Digi-Key API or you can download Digi-Key collections for Postman.

1. In the Postman toolbar, click **Import**.
   The "Import" window appears.
2. From the **Import File** tab, click **Choose Files** to import the JSON file from Digi-Key collections.

The Digi-Key collection appears in the Collections list:

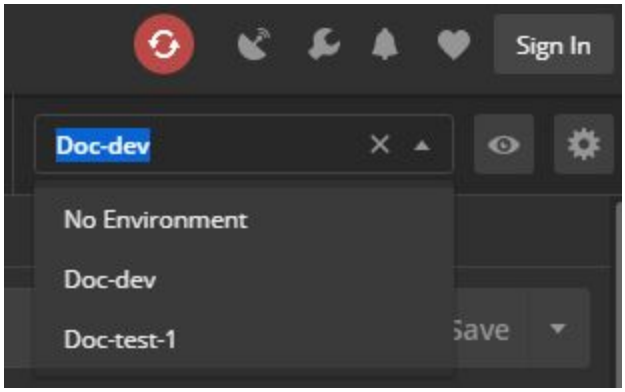


Configuring Postman

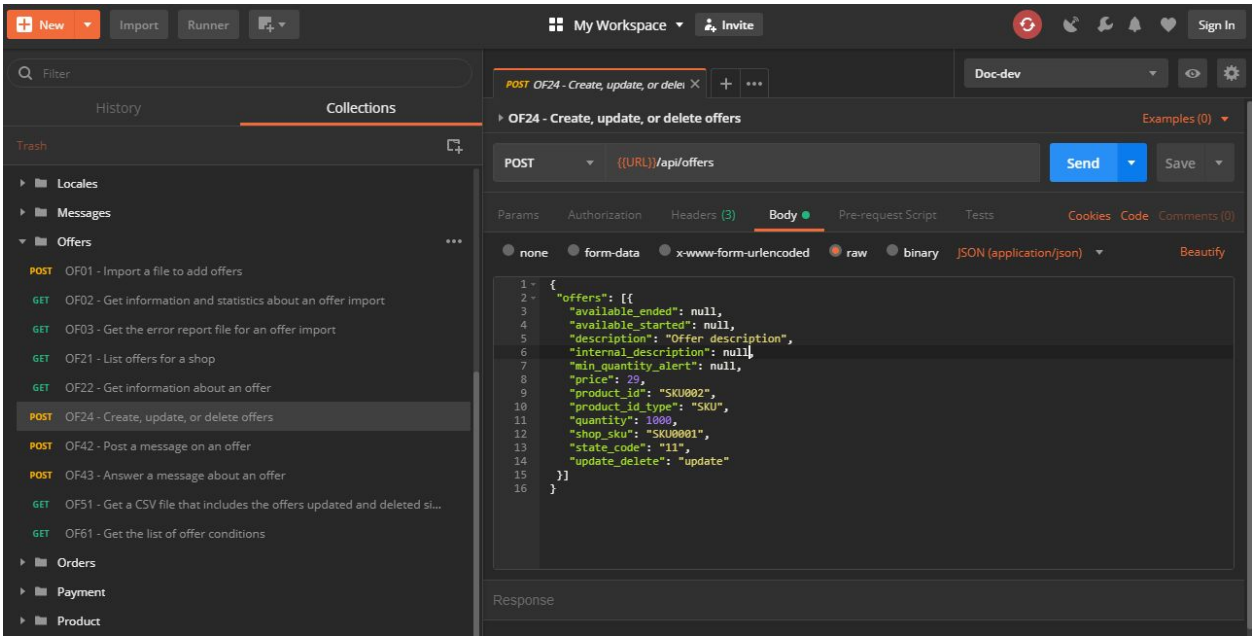
You must use Postman together with a live Digi-Key test environment.

Follow this procedure to allow Postman to interact with Digi-Key.

1. In the top right corner of Postman, click  .
   The "Manage environments" window appears.
2. Click **Add**.
3. Enter a name for the environment.
4. Click the **Variable** field to add:
   a. SHOP_KEY (generated for each store account)
   b. URL (URL for the Digi-Key environment (e.g. https://digikeyuat-dev.mirakl.net)
5. For each key, enter a value in the **Initial value** field.
6. Click **Add**.
7. Close the "Manage Environments" window.
8. Click the **No Environments** dropdown list and select your environment:



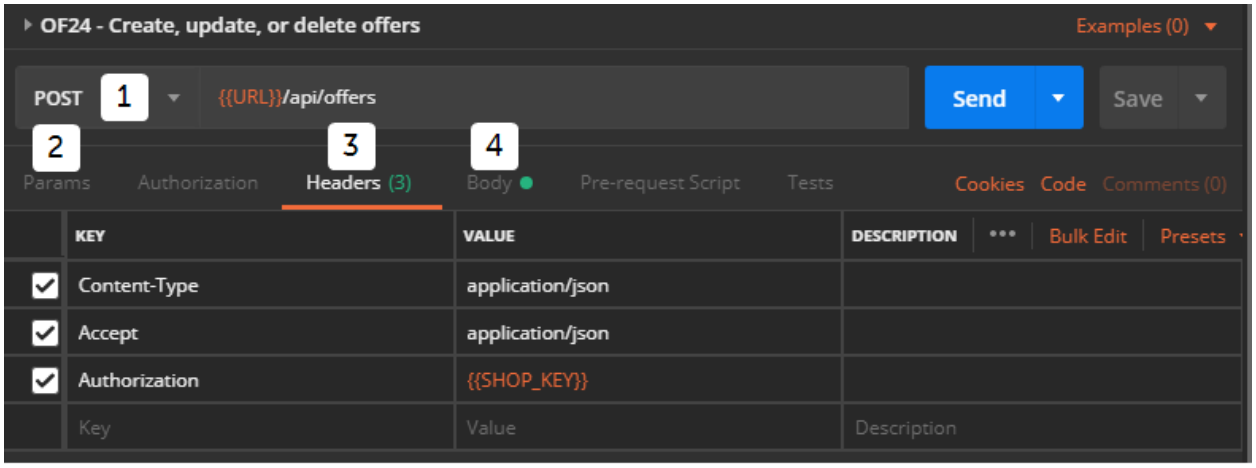
You are now ready to test Digi-Key API calls.



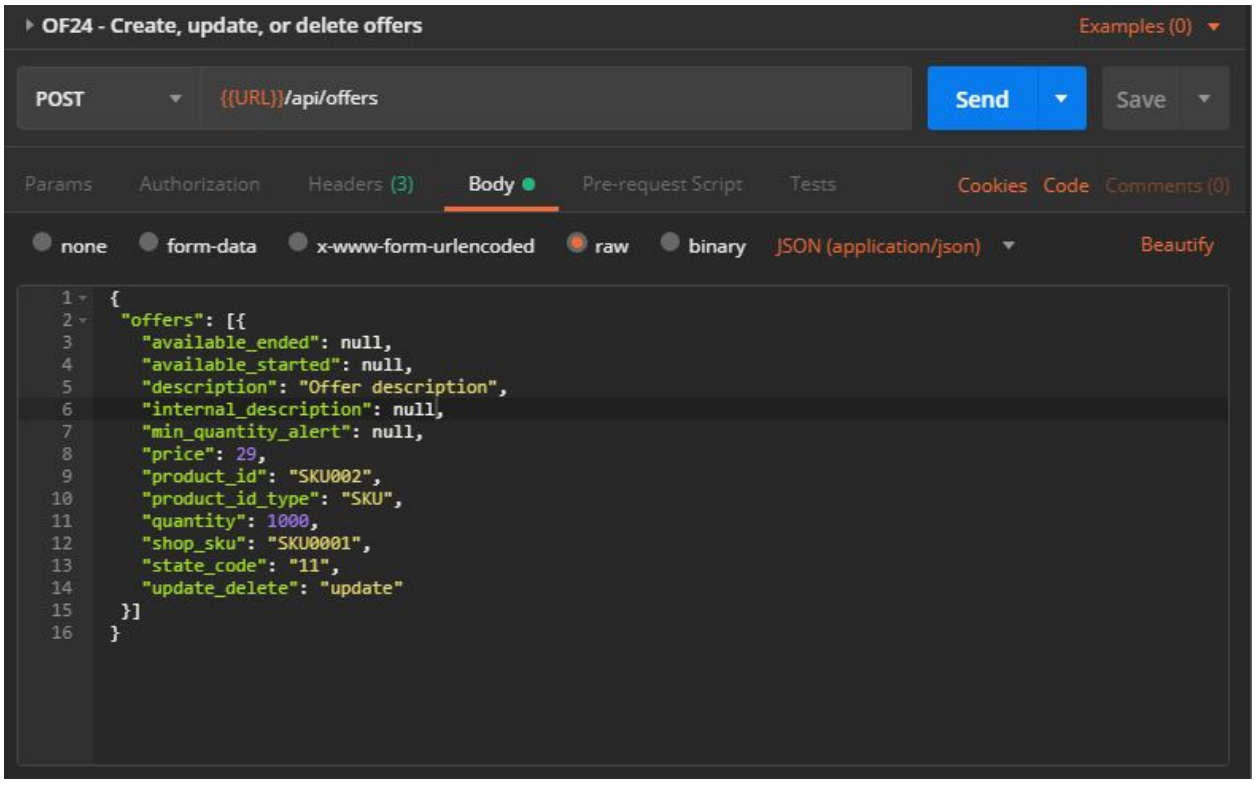
For each API in the collection, you can check how the HTTP request is built, for example by checking:

- method
- params
- headers
- body

As you can see in the following illustrations:



1 Method 2 Params (when available) 3 Headers 4 Body (when available)



Software Development Kit

Digi-Key provides a PHP software development kit (SDK) that wraps the Digi-Key REST API in a lightweight library. This enables you to develop a fast, flexible, and customized integration with Digi-Key and your e-commerce solution and IT systems.

The PHP SDK helps take the complexity out of coding by providing PHP methods for every step of your Store integration such as:

- catalog integration
- order management
- payment process
- and more

The Client Library (SDK) handles all the HTTP communications and JSON serialization thus no specific knowledge of REST API development is necessary.

Our PHP SDK comes as a single, downloadable package that includes:

- the Digi-Key PHP library
- code samples
- and PHPDoc documentation

The SDK is also available for Java platforms.

PHP SDK release notes and downloads

Download the latest version of the PHP SDK

- PHP SDK: shop APIs

Find PHP SDK release notes here.

Technical stack PHP SDK

Minimum requirements

The Digi-Key PHP SDK requires PHP 5.5+ or PHP 7+.

The SDK is built, assembled, and installed using Composer, a Dependency Manager for PHP.

Why using PHP 5.5?

The PHP SDK uses the latest GuzzleHTTP v6 component that is PSR-7 compliant and has the minimum PHP version needed to 5.5.

More information about PHP supported versions available here: http://php.net/supported-versions.php.

That is why we have to use an experienced PHP version and not a too recent one.

Installing the PHP SDK

Steps to install the SDK

1. First install Composer.
2. Then, use the installation from downloaded files.
3. At the end, test your installation.

Installing Composer

If Composer is not yet installed, run the following command from the installation root folder:

```
curl -sS https://getcomposer.org/installer | php
```

A file named composer.phar is created.

For more information about Composer global installation, refer to Composer website.

Installing from downloaded files

1. Download the latest version of the SDK:
   - PHP SDK: shop APIs
2. Copy the file in <working directory>/var/zip/.
3. Modify file composer.json like below:
4. ```
   {
     "repositories": [
       {
         "type": "artifact",
         "url": "var/zip/"
       }
     ],
     "require": {
       "mirakl/sdk-php": "*"
     }
   }
   ```
5. Run the following command:
6. php composer.phar update
   ```

The PHP SDK is installed under vendor/ directory.

Testing your installation

To test your installation, create a test.php file and paste the following code into it.

This test calls Digi-Key API H11 to list Catalog categories related to another Catalog category. By default, this test uses the "Front" API role (lines for the other API roles are commented out but you can remove comments to call the API with the role you want).

```
<?php
require_once(dirname(__FILE__) . '/vendor/autoload.php');

use Mirakl\MMP\Front\Client\FrontApiClient as Client;
use Mirakl\MCI\Front\Request\Hierarchy\GetHierarchiesRequest;

//use Mirakl\MMP\Operator\Client\OperatorApiClient as Client;
//use Mirakl\MCI\Operator\Request\Hierarchy\GetHierarchiesRequest;

//use Mirakl\MMP\Shop\Client\ShopApiClient as Client;
//use Mirakl\MCI\Shop\Request\Hierarchy\GetHierarchiesRequest;

$apiUrl = 'https://xxxxxxxxxx.mirakl.net/api';
$apiKey = 'xxxxxxxx-xxxx-xxxxx-xxxx-xxxxxxxxxxxx';

$client = new Client($apiUrl, $apiKey);
$request = new GetHierarchiesRequest();
echo sprintf('%d hierarchy found', count($client->getHierarchies($request)));
```

PHP SDK architecture

About Composer

All the final packages are handled in a single Composer dependency. It allows us to release all the packages together or to easily filter the packages we want to include.

This is why we have a unique composer.json file in project's root directory. The filtering is handled by a shell script described below.

```
{
        "name": "mirakl/sdk-php",
        "description": "Digi-Key provides a PHP SDK that wraps the Digi-Key REST APIs in a lightweight library. This enables you to develop a fast, flexible and custom integration for your existing e-commerce solution.",
   "type": "library",
   "version": "1.8.0",
   "license": "Digi-Key Platform",
   "authors": [
      {
```

```
            "name": "Digi-Key",
            "email": "info@mirakl.com"
          }
        ],
        "autoload": {
          "psr-4": {
            "Mirakl\\": "src/Mirakl/",
            "Mirakl\\Tests\\": "tests/Mirakl/"
          },
          "files": ["src/Mirakl/functions.php"]
        },
        "require": {
          "php": ">=5.5.0",
          "ext-curl": "*",
          "guzzlehttp/guzzle": "~6.0"
        },
        "require-dev": {
          "phpunit/phpunit": "~5.0"
        }
      }
```

SDK organization

In the following diagram:

- "MMP" refers to Mirakl Platform for Products
- "MCI" refers to the Mirakl Catalog Integrator

```
├─── bin                    Shell scripts and other binary files
├─── build                  Releases target dir
├─── src                    All Mirakl PHP SDK classes source dir (Composer generic)
│   └─── Mirakl
│       ├─── Core           Classes that are used by both MCI and MMP packages
│       ├─── MCI            MCI only classes
│       │   ├─── Common         MCI common classes between Front, Operator and
Shop packages
│       │   ├─── Front      MCI Front only classes
│       │   ├─── Operator   MCI Operator only classes
│       │   └─── Shop       MCI Shop only classes
│       └─── MMP            MMP only classes
│           ├─── Common         MMP common classes between Front, Operator and
Shop packages
│           ├─── Front      MMP Front only classes
│           ├─── FrontOperator  MMP Front + Operator common classes
│           ├─── FrontShop  MMP Front + Shop common classes
│           ├─── Operator   MMP Operator only classes
│           ├─── OperatorShop  MMP Operator + Shop common classes
```

```
│       └──── Shop          MMP Shop only classes
├──── tests                 Unit tests dir
│   ├──── Mirakl
│   │   ├──── Core          Core unit tests
│   │   ├──── MCI           MCI only unit tests
│   │   └──── MMP           MMP only unit tests
│   └──── resources         Resource files used for API requests/responses mocking
│       ├──── mci           MCI only resource files
│       └──── mmp           MMP only resource files
└──── vendor                Composer internal (dependency components installation)
```

How does the PHP SDK work?

```
/**
 * This example is for the role Frontend. For any other roles, there are other clients.
 * For the role Shop, the appropriate client is ShopApiClient (Mirakl\MMP\Shop\Client\ShopApiClient)
 */
require 'vendor/autoload.php';

use Mirakl\MMP\Front\Client\FrontApiClient as MiraklApiClient;
use Mirakl\MMP\Front\Request\Offer\GetOfferRequest;

try {
    // Instantiating the Mirakl API Client
    $client = new MiraklApiClient('https://your.env/api', 'your_frontend_api_key');

    // Building request
    $request = new GetOfferRequest('2000');

    // Calling the API
    $result = $client->getOffer($request);
    var_dump($result);

    // Alternatively, the following syntax is also valid:
    $result = $request->execute($client);
    var_dump($result);

} catch (\Exception $e) {
    // An exception is thrown if the requested object is not found or if an error occurs
    var_dump($e);
}
```

Manipulating raw response

The default behavior is to decorate the response with domain objects for easy manipulation. Additionally, you can retrieve more information about the API response, for debugging purposes, for example.

Use the raw() method of the API that returns a \Psr\Http\Message\ResponseInterface.

```
/** @var \Psr\Http\Message\ResponseInterface $result */
 // Retrieve a raw response
$result = $api->raw()->getOffer($request);
var_dump($result);
 // Decorate the response manually
$offer = $request->getResponseDecorator()->decorate($result);
```

Uploading files

Some API requests need files to be uploaded. You have several ways to provide a file to an object of Request type.

Here is an example for the API OF01 request that allows offer updates from a file:

```
use Mirakl\MMP\FrontOperator\Request\Catalog\Product\ProductSynchroRequest;

// Method #1
$request = new ImportOfferRequestt('CSV file body contents');

// Method #2
$request = new ImportOfferRequestt(new \SplFileObject('/path/to/file.csv'));

// Method #3
$request = new ImportOfferRequestt(['PHP array that represents the CSV contents to import']);
```

Downloading files

Some API requests produce files you are able to download. This is the case for the API IV02 that is used to download an invoice.

Here is how you can create the request and download the invoice:

```
use Mirakl\MMP\Shop\Request\Payment\Invoice\DownloadInvoiceRequest;
 // Create the request
$request = new DownloadInvoiceRequest('INVOICE_ID'); // IV02
 // Call API and get the response ($client is the API Client and has been initialized before)
$result = $client->downloadInvoice($request);
/** @var \Mirakl\Core\Domain\FileWrapper $result */
 // Download the invoice
$result->download();
```

Advanced PHP SDK usage

Magic methods

When you manipulate Request or Domain objects, you can easily use magic methods on them. This kind of object extends a basic object that handles the magic methods.

A Digi-Key data object is namespaced \Mirakl\Core\Domain\MiraklObject.

Getters

Example 1

```
$request = new \Mirakl\MCI\Front\Request\Hierarchy\HierarchyImportErrorReportRequest('2000');
$request->getImportId();
$request->getType();
// ...
```

Example 2

```
$order = new \Mirakl\MMP\FrontOperator\Domain\Order();
$order->getShopId();
$order->getShopName();
// ...
```

Setters

```
$request = new \Mirakl\MCI\Front\Request\Hierarchy\GetHierarchiesRequest();
$request->setMaxLevel(2);
// ...
```

Boolean values

```
$offerOnProduct = new \Mirakl\MMP\Common\Domain\Product\Offer\OfferOnProduct();
$offerOnProduct->isPremium();
// ...
```

Other methods

```
$request->getData();          // Returns all data of current object
$offer->toArray();            // Converts object to PHP array recursively
$order->getCollection();// Initializes a new Collection containing Order objects
// More details in file src/Mirakl/Core/Domain/MiraklObject.php
```

Asynchronous requests

You can send commands concurrently using the asynchronous features of the SDK (Guzzle feature).

You can send requests asynchronously by using the method async() on the $client object. This method initiates the request and returns a promise. On success, the promise is fulfilled with the result object; on failure, the promise is rejected with an exception. This allows you to create multiple promises and have them send HTTP requests concurrently when the underlying HTTP handler transfers the requests.

```
/** @var \GuzzleHttp\Promise\Promise $promise */
 // Retrieve a raw response
$promise = $client->async()->getOffer($request); // Non-blocking code
var_dump($promise);
 // Effectively run the request against Digi-Key API
$result = $promise->wait();
// $result will contains a decorated response.
// You can use both raw() and async() methods together
```

Running tests on a package

The Digi-Key PHP SDK comes with a set of unit tests built with the PHPUnit framework.

To run unit tests of a specific package:

1. Create a phpunit.xml file in the package root directory.
2. Here is an example of phpunit.xml:
3. ```
   <phpunit
           colors="true"
           stderr="true"
           convertErrorsToExceptions="true"
           convertNoticesToExceptions="true"
           convertWarningsToExceptions="true"
           stopOnFailure="false"
           bootstrap="tests/bootstrap.php">
       <testsuites>
           <testsuite name="Mirakl PHP SDK Test Suite">
               <directory>./tests</directory>
           </testsuite>
       </testsuites>
       <filter>
           <whitelist>
               <directory suffix=".php">./src/Mirakl</directory>
           </whitelist>
       </filter>
   </phpunit>
   ```
4. Run the following command to launch unit tests (still from the root directory):
5. $ phpunit [--debug] [--group=MMP|MCI] [--group=front|operator|shop] [--group=S20] ...

PHPUnit is declared as a dev dependency in composer.json file. It means that if you run the composer install command with --no-dev parameter, the PHPUnit is not available.
   ```

PHP SDK support information

Contacting a Support team

If you have any questions about the SDK, contact DKE Marketplace.

About support level for the SDK

The SDK is delivered "as-is". There are no SLA's which apply to resolution times for any issues with the current SDK.

The SDK is much more flexible and does not depend on the Digi-Key Platform for Products application: changes and turnaround times can be very quick if a problem is encountered.

About integration with the others Digi-Key modules

One of the main benefits of using the Digi-Key SDK is that it is easier and less costly to maintain over time. Version upgrades become “more” seamless as Digi-Key handles any changes to the API within the SDK. Digi-Key can continue to call the exact same method.

The PHP SDK supports the necessary APIs to integrate with the most recent Digi-Key modules: the U.S. Tax Connector and the Digi-Key Catalog Manager.

Java Software Development Kit

Digi-Key provides a Java software development kit (SDK) that wraps the Digi-Key REST API in a lightweight library. This enables you to develop a fast, flexible, and customized integration with Digi-Key and your e-commerce solution and IT systems.

The Java SDK helps take the complexity out of coding by providing Java methods for every step of your Store integration such as:

- catalog integration
- order management
- payment process
- and more

The Client Library (SDK) handles all the HTTP communications and JSON serialization thus no specific knowledge of REST API development is necessary.

Our Java SDK comes as a single, downloadable package that includes:

- the Digi-Key Java library
- code samples
- and Javadoc documentation

Java SDK release notes and downloads

Directly download the latest version of the Java SDK:

API Role	MCI/MCM	MMP
Java SDK: shop APIs	Library for MCI/MCM modules	Library for MMP module

Find JAVA SDK release notes here.

Technical stack Java SDK

The Java SDK is built, assembled, and installed using Apache Maven.

Digi-Key SDK releases are deployed to the help portal.

The library is compiled with Java SE 6 and you can use it with Java 6, 7, and 8.

The SDK relies on the Apache HttpClient 4.5 library and Jackson 1.9 is used to process JSON.

Migrating from 3.24.x or a previous version?

Since version 3.25.x, the Digi-Key Java SDK library has been overhauled. The focus of this new release was to reduce the complexity of external libraries used by the Digi-Key client.

Previously HTTP calls were made with Jersey, which turned out to be too complex to integrate due to the possibility of multiple versions in the classpath. This dependency has been removed and replaced by Apache HttpClient which is a much more lightweight library.

Upgrading to the 3.25.x version should be seamless, the only impacts are:

- Configuration made to the client through the #configure(MiraklClientConfigWrapper clientConfigWrapper) will change a little bit since:
  - we do not expose the ClientConfig from Jersey anymore
  - but we expose the HttpClientBuilder and the RequestConfig.Builder from Apache HttpClient
- When exceptions occurred during a call to a Digi-Key API, the Throwable is wrapped in MiraklApiException. What has changed is the cause of those exceptions. Jersey was throwing WebApplicationException, but now Apache HttpClient throws IOException.

Installing the Java SDK

Prerequisites

Retrieve the latest version of the Java SDK.

Add the Mirakl repository to your **pom.xml**:

```
<repositories>
  <repository>
    <id>mirakl-ext-repo</id>
    <name>Mirakl External Tools Repository</name>
    <url>https://artifactory.mirakl.net/artifactory/mirakl-ext-repo/</url>
  </repository>
```

1. </repositories>

Set the authentication configuration in your **settings.xml** (in your M2 repository):

```
<server>
  <id>mirakl-ext-repo</id>
  <username>username</username>
  <password>password</password>
```

2. </server>
3. Add each of the libraries to your **dependency list**.
   Mirakl Library
   To know which libraries you need to use, refer to the Mirakl API documentation.
   Mirakl's libraries relate to the ROLE of the API and the scope, whether it is part of the MMP (Mirakl Platform for Products) module or MCI (Mirakl Catalog Integrator and Mirakl Catalog Manager) modules.

Non-maven users

If you do not use Maven, we package an additional JAR that contains all the dependencies required by the SDK.

To download this package:

1. Log in to Artifactory.

2. Browse the repository to download the JAR ending by the -jar-with-dependencies.jar suffix.

Here is an example for mmp-sdk-front and version 3.25.17.

Seller library examples

For a seller who wants to integrate as a shop on the Mirakl Platform for Products, use:

```
<dependencies>
   <dependency>
      <groupId>com.mirakl</groupId>
      <artifactId>mmp-sdk-shop</artifactId>
      <version>${LAST_MIRAKL_SDK_VERSION}</version>
   </dependency>
</dependencies>
```

For a seller who wants to use Mirakl Catalog Integrator API, use:

```
<dependencies>
   <dependency>
      <groupId>com.mirakl</groupId>
      <artifactId>mci-sdk-shop</artifactId>
      <version>${LAST_MIRAKL_SDK_VERSION}</version>
   </dependency>
</dependencies>
```

Complete list of libraries on Artifactory

Here is the full list of all our artifactId available:

```
<artifactId>mmp-sdk-front</artifactId>
<artifactId>mmp-sdk-operator</artifactId>
<artifactId>mmp-sdk-shop</artifactId>
<artifactId>mci-sdk-front</artifactId>
<artifactId>mci-sdk-operator</artifactId>
<artifactId>mci-sdk-shop</artifactId>
```

How does the Java SDK work?

Example

Here is a code example to call our APIs:

-OR11 - List orders

```
package com.mirakl;
```

```
import com.mirakl.client.*;
import org.slf4j.*;
import java.io.IOException;
import java.util.*;

/**
 * Basic sample for calling Mirakl OR11 API
 */
public class TestMiraklClient {

   private static final String ENV_URL = "https://your-env.mirakl.net/api";
   private static final String API_KEY_FRONT = "your_api_key";
   private static final Logger LOGGER = LoggerFactory.getLogger(TestMiraklClient.class);

   public static void main(String[] args) throws IOException {
      // Configure the authentication
      MiraklCredential credential = new MiraklCredential(API_KEY_FRONT);
      // Instantiating the Mirakl Api Client
      MiraklMarketplacePlatformFrontApi client = new
MiraklMarketplacePlatformFrontApiClient(ENV_URL, credential);
      // Build the request
      MiraklGetOrdersRequest request = new MiraklGetOrdersRequest();
      List<String> commercialIdsFilter = new ArrayList<String>();
      commercialIdsFilter.add("94843IDJD");
      request.setCommercialIds(commercialIdsFilter);

      // Call the API OR11
      MiraklOrders result = client.getOrders(request);

      LOGGER.info("{} order(s) have been retrieved", result.getOrders().size());
   }
    }
```

Timeouts

By default, the connect timeout is set to 30 seconds on the SDK client. The read timeout is set to 60 seconds. Those values are default values and can be overridden when configuring the client.

To set timeouts, see the following example:

-Example: Configuring timeout for SDK

```
MiraklCredential auth = new MiraklCredential("e67db8ef-0387-4bb3-9903-c2b323e78e19");
```

```
MiraklMarketplacePlatformFrontApi client = new
MiraklMarketplacePlatformFrontApiClient("https://client.mirakl.com/api", auth) {
  @Override
  protected void configure(MiraklClientConfigWrapper clientConfigWrapper) {
    super.configure(clientConfigWrapper);
    clientConfigWrapper.getRequestConfigBuilder().setConnectTimeout(1 * 60_000); // 1 min
    clientConfigWrapper.getRequestConfigBuilder().setSocketTimeout(10 * 60_000); // 10 mins
  }
   };
```

Using the SDK through a proxy

If you want to call Mirakl APIs through a proxy, use features provided by the Apache HttpClient library.

Large file / Chunked mode

By default, the SDK is using buffered upload to send files. This means that the file is read and stored in memory before being sent to the server. This can be an issue with very large files (more than 1 gigabyte), and you will get a java.lang.OutOfMemoryError.

To avoid this, the Mirakl SDK provides some extra methods returning an InputStream instead of a list of pojos. We recommend that you use those methods for APIs with large amount of data like OF51 API.

Java SDK support information

Contacting a Support team

If you have any questions about the SDK, contact DKE Marketplace.

About support level for the SDK

The SDK is delivered "as-is". There are no SLA's which apply to resolution times for any issues with the current SDK.

The SDK is much more flexible and does not depend on the Mirakl Platform for Products application: changes and turnaround times can be very quick if a problem is encountered.

About integration with the others Mirakl modules

One of the main benefits of using the Mirakl SDK is that it is easier and less costly to maintain over time. Version upgrades become "more" seamless as Mirakl handles any changes to the API within the SDK. Digi-Key can continue to call the exact same method.

The Java SDK supports the necessary APIs to integrate with the most recent Mirakl modules: the U.S. Tax Connector and the Mirakl Catalog Manager.

Appendix A: Digi-Key API - General notes

- Authentication
- Request Content-Type
- Response Content-Type
- SSL Only
- UTF-8 Encoding
- Date Format
- Locale Format
- API Return Codes
- Pagination and Sort
- cURL Samples
- Shop Selection
- Shop Roles
- Operator Roles
- Security
- Resources

Authentication

You can authenticate through API by sending your API key in a header:

curl -H "Authorization: 123456" https://[HOSTNAME]/api/[API-URL]

Request Content-Type

You can choose between json or xml format.

Use Content-Type Header to set the request content type.

curl -H "Content-Type: application/json" https://[HOSTNAME]/api/[API-URL]

Accepted values are:

- application/json
- application/xml
- multipart/form-data*(Allow to send file and form data. See the cURL samples at the end of this page)*

Response Content-Type

For non-binary responses you can choose between json or xml format.

Use Accept Header to choose the response content type.

curl -H "Accept: application/json" https://[HOSTNAME]/api/[API-URL]

Accepted values are:

- application/json
- application/xml
- text/csv

SSL only

All requests should use the SSL protocol.

UTF-8 Encoding

All text data is encoded in UTF-8. The binary content is not concerned.

Date Format

All timestamps are returned in ISO 8601 format:

yyyy-mm-dd'T'HH:mm:ss'Z'

ISO 8601 considers the time zone.

The examples below are equivalent:

2015-08-29T04:34:00Z
2015-08-29T06:34:00+02:00

We strongly advise you to specify timestamps with timezone. You can indicate partial timestamps, but be aware that we will use our server's timezone instead.

Please note that query parameters must be URL encoded, i.e. when used into a query parameter, character '+' must be encoded as '%2B'. For instance, to send a date in a query parameter, the following syntax should be used: 2012-08-29T04:09:00%2B08:00

Locale Format

Some resources return internationalized data. In this case, you can use the locale parameter:

Example: get assessments information in english (US)

_/api/evaluations/assessments?locale=en_US

Locale format is usually <ISO-639>_<ISO-3166> (e.g. "en_US"). There are some exceptions where the Locale format can be <ISO-639> (e.g. "en").

When a locale is returned in a response, it also uses this format.

Accepted locales are the ones corresponding to the languages that have been activated by Digi-Key in the back-office.

API Return Codes

Digi-Key API uses standard HTTP codes.

Success Codes

200

OK - After a successful GET.

Returns the resource

201

Created - After a successful POST.

Returns The URI of the new resource. This URI is in the Location response header.

204

No Content - After a successful PUT or DELETE.

Does not return anything.

Error Codes

400

Bad Request - Parameter errors or bad method usage.

Bad usage of the resource. Example: a required parameter is missing, a parameter with bad format, query on data which are not in a expected state.

Response example

{"status":400,"message":"Parameter 'offer' is required"}

401

Unauthorized - Call of API without authentication

Add authentication information or use a valid authentication token.

Response example

{"status":401,"message":"Authentication Failed: Request does not have authentication token"}

403

Forbidden - Access to the resource is denied

Current user can not access the resource.

Response example

{"status":403,"message":"Access Denied"}

404

Not Found - The resource does not exist

The resource URI does not exist or the asking resource does not exist (for the current user).

Response example

{"status":404,"message":"Not Found"}

405

Method Not Allowed - The http method (GET, POST, PUT, DELETE) is not allowed for this resource

Refer to the documentation for the list of accepted methods.

Response example

{"status":405,"message":"Method Not Allowed"}

406

Not Acceptable - The requested response content type is not available for this resource.

Refer to the documentation for the list of correct values of the Accept header for this request.

Response example

{"status":406,"message":"Not Acceptable"}

410

Gone - Resource is gone.

The resource requested is no longer available and will not be available again.

Response example

{"status":410,"message":"Resource is gone."}

415

Unsupported Media Type - The entity content type sent to the server is not supported.

Refer to the documentation for the list of correct values of the Content-type header to send data.

Response example

{"status":415,"message":"Unsupported Media Type"}

429

Too many requests - Rate limit exceeded.

The user has sent too many requests in the last hour. Refer to the documentation for the maximum calls count per hour.

Response example

{"status":429,"message":"Rate limit exceeded. Please try again in 10 minutes."}

500

Internal Server Error - The server encountered an unexpected error.

Response example

{"status":500,"message":"Internal Server Error"}

Pagination and Sort

Paginate the results

Some resources support pagination. In this case, you can use the max and offset parameters:

max

The max parameter is used to indicate the maximum number of items returned per page. This parameter is optional. The default value of this parameter is 10. The maximum value of this parameter is 100.

Example: get only the first 5 offers

_/api/offers?max=5

offset

The offset parameter is used to indicate the index of the first item (among all the results) in the returned page. This parameter is optional. The default value of this parameter is 0, which means that no offset is applied.

Example: get offers starting from the 5th offer

_/api/offers?offset=4

Example:

_/api/offers?max=10&offset=0 (returns the first 10 offers)
_/api/offers?max=10&offset=10 (returns 10 offers starting from the 11th)
_/api/offers?max=10&offset=20 (returns 10 offers starting from the 21st)
_/api/offers?max=10&offset=30 (returns 10 offers starting from the 31st)

With pagination, the URL of the previous and/or next page can be returned in the header's attribute Link of the response.

Example: get only 5 offers from the 6th offer

Request

_/api/offers?max=5&offset=5

Header of the response

Link: <_/api/offers?max=5&offset=0>; rel="previous", <_/api/offers?max=5&offset=10>; rel="next"

Sort the results

Some resources support sorting. In this case, you can use sort and order parameters:

The parameter sort is used to indicate how the results should be sorted. This parameter is optional. The possible values for this parameter are defined in resources. The default value of this parameter is defined in resources.

Example: get offers sorted by price

_/api/offers?sort=price

order

The parameter order is used to indicate the sort direction. This parameter is optional. The possible values for this parameter are asc (default) or desc.

Example: get offers sorted by price in descending order

_/api/offers?sort=price&order=desc

cURL Samples

Here is the cURL parameters you need to use when testing the API.

GET

curl -X GET https://[HOSTNAME]/api/[API-URL]

PUT (JSON or XML content type)

curl -X PUT -H "Content-Type: application/json" -H "Accept: application/json" -d @data.json https://[HOSTNAME]/api/[API-URL]

PUT (CSV content type)

curl -X PUT -H "Content-Type: multipart/form-data" -H "Accept: application/json" -F "file=@data.csv;type=text/csv" https://[HOSTNAME]/api/[API-URL]

POST (JSON or XML content type)

curl -X POST -H "Content-Type: application/json" -H "Accept: application/json" -d @data.json https://[HOSTNAME]/api/[API-URL]

POST (CSV content type)

curl -X POST -H "Content-Type: multipart/form-data" -H "Accept: application/json" -F "file=@data.csv;type=text/csv" https://[HOSTNAME]/api/[API-URL]

Shop Selection

When calling APIs as a shop, a request parameter shop_id is available. This parameter is useful when a user is associated to multiple shops and should be specified to select the shop to be used by the API. If this parameter is not specified, the first shop associated to the user will be used. When called as a user which is not a shop user, APIs ignore this parameter.

curl https://[HOSTNAME]/api/[API-URL]?shop_id=123456

Shop Roles

Only users with the "Shop Administration" role can call APIs as a Shop. Calls to any API as a Shop without this role will result in a 403 response.

Security

API output may contain HTML and javascript, before displaying data on your website make sure to escape it.

Appendix B: Digi-Key Seller API Documentation

**Note - Must be logged in to seller account for links to work*

API Documentation - Specific details about each API call

- Product Services
  - Invoicing and Accounting
  - Messages
  - Offers
  - Orders
  - Platform Settings
  - Products
  - Promotions
  - Stores