Order Migration Guide

Order Migration Overview

This section covers in-depth information on best practices related specifically to migrating orders. It assumes you have already migrated Products and Customers, but not B2B information (if applicable). Some features have been omitted, usually because they are read-only, to limit the scope and complexity of this guide.

If you are migrating B2B information, it is highly recommended that you follow the B2B guide in lieu of this guide.

Contents of this guide

The following sections provide the relevant information for migrating orders:

- Complexity Considerations - factors to consider in preparing your data and how best to handle the individual aspects of migrating order information
- Prepare Data - relevant structures and data for how to handle orders during the migration phase
- Migration Phase - basic structure of migration process including some measure of time complexity and the order-specific steps relevant to the migration process

Special considerations

Because order data often contains customer information, it is important to handle your data with care. If you or your customers are in a region affected by the GDPR, be sure to take privacy into account when migrating order data. Generally, it is enough to anonymize only that data that has been specifically requested, but you may want or need to do more.

In addition to privacy concerns, BigCommerce billing is based on total orders and trailing order value. In order to prevent billing errors, there is a field in the order creation process that must have the appropriate flag set. Specifically, the field `external_source` should have the value `M-MIG` when creating the order, or it will be included in the billing data, which will cause a potential billing mismatch.

If you have not configured shipping options or payment options, you can add orders without this information, but it may cause some issues with record-keeping if that data is skipped during migration. As such, you should consider configuring both your shipping and payments information ahead of order migration.

Unlike with product migration, only one endpoint should be necessary for initial order migration, although other endpoints may become relevant in delta migration if necessary. Both endpoints are outlined in the Orders API Reference.

FAQs

1. **What is the recommended way to avoid billing errors during order migration?**
   Always set `external_source` to `M-MIG` when creating migrated orders to prevent billing mismatches.
2. **Can I migrate orders without shipping or payment data?**
   Yes, but it may cause record-keeping issues. For completeness, configure both ahead of migration.
3. **Is there a separate guide for B2B order migration?**
   Yes, refer to the dedicated B2B migration guide (coming soon).
4. **How do privacy laws (like GDPR) affect order migration?**
   Anonymize personal data as requested. See GDPR and BigCommerce for details.
5. **Where can I get help for large or complex order migrations?**
   Contact BigCommerce Data Migration Services.

Additional resources

- Orders API Reference
- GDPR and BigCommerce
- Shipping Setup
- Available Payment Gateways
- API Best Practices
- Data Migration Services

Complexity Considerations

While order migration can be considered as a strictly “move information from here to here” sort of operation, there are factors that result in higher or lower levels of complexity that will change the ideal strategy used in migration planning.

The content in this page outlines various considerations that go into planning against complexity for your migration strategy. During the preparation phase, you can refer back to the tables and flowcharts in this document to help build a clear understanding of the steps necessary to ensure a successful migration.

While this stage isn’t explicitly necessary, it does help to mitigate potential pain-points in the migration process as a whole.

Products

If your catalog is primarily simple products, the process only requires selection of products by `product_id`, while catalogs with primarily variable or modified products will require both `product_id` and some identification of the variants, modifiers, and possibly other data points to properly build up the order data. Here’s a breakdown of relevant factors to consider with regard to products:

Factor	**Data**	**Use Case Example**
Simple Products	`product_id`	Product with only one available ‘type’
Products with variants only	`product_id`, and `variant_id` or `product_options` array	Product with colors or sizes `variant_id`
Products with modifiers only	`product_id` and `product_options` array	Product with custom text or other factors unaffected by inventory.
Products with both	`product_id`, `product_options` array, and optionally `variant_id`.	Product with color selections and custom text

This splitting is reflected in the flowchart below, but it isn’t strictly necessary.

As these features occur on a product-to-product basis, your migration code should account for each possible case that affects your catalog. Ideally you should aim for the minimal implementation that accomplishes your catalog’s needs, but a simple flow-chart for product handling at any complexity follows:

```
flowchart LR
    B[Get<br />Product]
    B --> C[Add<br /> ID & Name]
    C --> D{Variants?}
    D -->|No| F{Modifiers?}
    D -->|Single| G[Add<br />Variant ID] --> F
    D -->|Multiple| H[Add<br />Options] --> F
```

```
    F -->|No| J[Output]
    F -->|Yes| I[Add<br />Options] --> J

    J --> K{More?}
    K -->|Yes| B
    K -->|No| M[Complete]
```

More information on this will be explored when we cover the fields related to the Create an Order endpoint.

Customers

Customer complexity may be introduced in a few ways, but the primary sources are in address and contact data. Specifically, if billing and shipping information differ, separate objects will need to be built to manage the addresses when building an order's total object data, but if they are the same a deep copy of one into the other will suffice.

Likewise, if an order includes multiple separate shipping addresses, the array for shipping addresses will contain an address object for each. For older orders that have been completely fulfilled, this is less likely to impact your workflow, but outstanding shipping will require some consideration.

Complexity considerations for customers are outlined here:

Factor	**Data**	**Use Case Example**
Guest customers	omit `customer_id` field	Customer does not log in, or does not have an account, before or during checkout.
Billing address matches shipping address	`billing_address`	Customer ships and bills to a single address, usually attached to an account.
Billing address differs from shipping address	`billing_address` and `shipping_addresses` fields	Customer shipping to PO Box or to alternate residence/office.
Multiple shipping addresses	`shipping_addresses` array	Customer ordering for multiple locations at once.
Consignments	`consignments.pickups` array	Customer picking up whole orders 'in store'.

Just as with products, customer order data may be mixed complexity. As a result, either a pre-migration analysis or an adaptive order workflow should be implemented to capture the full scope of your project. While the latter is more effective, the former can help establish the scope of the adaptive workflow ahead of time and decrease complexity overall.

```
flowchart LR
    B{Has<br />`customer_id`} -->|No| C[Add Billing<br />Address]
    B -->|Yes| D[Add ID] --> C

    C --> E{Has<br />Consignments}
    E -->|No| F[Add<br />Products]
    E -->|Yes| G[Add<br />Consignments] --> H[Complete]
```

```
    F --> I --> H
    subgraph I[Shipping Addresses]
        I1{Has Addresses?} -->|No| I2[End]
        I1 -->|Yes| I3[Add to Array] --> I1
    end

    click F "#products"
```

Order-level

Once customer-specific and product-specific considerations are handled, some order-level issues may arise that increase complexity. For example, orders may have discounts applied that are no longer valid or haven't been configured within BigCommerce. Alternatively, a specific order might have an order status that is no longer valid for any number of reasons.

There are too many possible causes for order-level complexity to lay them out, but the following table gives several examples of fields that exist, the context they apply, and a use-case example for each.

Factor	**Data**	**Use Case Example**
Order-level Fees	`fees` array	Shipping to Colorado and applying the Retail Delivery Fee for record-keeping.
Customer Comments	`customer_message` field	Customers leave notes for shipping instructions or to thank your team.
Staff Notes	`staff_notes` field	Your team leaves comments about order management. Not visible to customers.
Gift Wrapping	`*_wrapping_cost` fields	Customers request gift wrapping, if you provide the option.
Incomplete Payments	varies	Customers using offline payment methods or customized split payment systems.

FAQs

1. **What should I do if my catalog contains custom products not present in BigCommerce?**
   Use the custom product workflow, providing name, quantity, and price fields. See Orders API - Create an Order for details.
2. **Can I migrate guest orders?**
   Yes. Omit the `customer_id` field for guest orders; use billing address/contact info for identification.

3. **How do I handle orders with multiple shipping addresses?**
   Include an array under `shipping_addresses` with each address object. Ensure each address is mapped correctly.
4. **What if my order includes both consignments (pickups) and deliveries?**
   Consignments and shipping addresses/products are mutually exclusive in a single order. Split such orders into separate records if both are needed.
5. **How do I map product variants and modifiers in order data?**
   Include the relevant `product_id`, `variant_id`, and/or `product_options` array per product line item.
6. **What fields are required for a minimal order migration?**
   At minimum: `external_source`, `external_order_id`, `status_id`, `date_created`, `billing_address`, `products`. See Prepare Data for more.
7. **How do I ensure order data is not duplicated?**
   Use unique identifiers and check for existing orders before creation. Log all migrated order IDs for auditing.

Resources

- Orders API - Create an Order
- Catalog API
- Order Management (Help Center)
- Payments API
- API Best Practices

Prepare Data

Prior to attempting a migration the following are essential steps to ensure data are ready. Skipping these steps may result in missing or incomplete order information post-migration.

Minimal order example

The following example is intended to show the absolute minimum amount of information you can provide when creating orders in order to migrate your data smoothly, efficiently, and without undue data collisions.

```
{
  "external_source": "M-MIG",
  "external_order_id": "AB123456",
  "status_id": 0,
  "date_created": "Tue, 20 Nov 2012 00:00:00 +0000",
  "customer_message": "Thank you",
  "staff_notes": "Ship sunday",
  "total_ex_tax": "23.74",
  "total_inc_tax": "25.61",
  "billing_address": {
      /*... billing address info ...*/
  },
  "products": [
      /*... product data objects ...*/
  ],
}
```

The other fields that should be included depend on the complexity of your migration and should be added as necessary depending on your needs.

In the remaining sections, all order data mapping is covered in detail to allow a full and complete picture of the process. Your needs will vary depending on the complexity of your migration, customer data, and catalog.

Product data links

It's assumed that you have already transferred your products following the instructions in Product Migration. During that process, you should have ensured that your products have the BigCommerce `product_id` attached so that order information is able to attach the correct products within BigCommerce. For products that have variations or modifications that might change the inventory tracking or other relevant data access, the appropriate `variant_id`, `modifier_id`, and `sku` information should be attached to relevant product data within your orders.

As covered in the fields outline from the product migration page, these fields may be labelled differently depending on the specific API endpoint used to access them during product migration. As such, it's essential that you label the fields clearly in your data source with respect to the data they belong to.

Field	Description	General Label	Create an Order API Label	Notes
Product ID	Server enforced	`product_id`	`product_id`	Read-only field.

Field	Description	General Label	Create an Order API Label	Notes
	numeric ID assigned to the parent product on creation.			
Variant ID	Server enforced numeric ID assigned to the variant on creation.	`variant_id`	`Product_options[idx].id` or none	Read-only field.
Variant Option ID	Server enforced ID assigned to specific selections available on a variant.	`variant_option_id`	`Product_options[idx].value` or `variant_id`	Read-only field. Editing the variant or its options **may** change the ID values. This usually occurs because editing will sometimes require deleting existing options. The `product_options` element may be skipped in this use-case if `variant_id` is provided.
Modifier ID	Server enforced numeric ID assigned to the modifier on creation.	`modifier_id`	`Product_options[idx].id` or none	Read-only field.
Modifier Option ID	Server enforced ID assigned to specific selections available on a modifier.	`modifier_option_id`	`product_options[idx].value`	Read-only field. Editing the modifier or its options **may** change the ID values. This usually occurs because editing will sometimes require deleting existing options. For modifiers without explicit selections, this is the value provided by the customer.

Field	Description	General Label	Create an Order API Label	Notes
Product SKU	SKU assigned to the parent product, usually used as a “family SKU” as in the case where size or color only change the last few digits.	`product_sku`	N/A	Nullable field. If this is left blank, the primary loss is that the parent product is not filterable on SKU.
Variant SKU	SKU assigned to specific variants of the product. May be wholly different than the `product_sku` value.	`variant_sku`	N/A	Not nullable. These values should be set when creating or editing variants. If they are not, the server will assign them.

Other product-level fields to map include `product_name`, `variant_name`, and the various `*_price` fields on the product. These should have a uniform name between both source and destination data, since their names are specific enough to make clear what field they reference.

For custom products those defined by you when creating an order or those no longer in your catalog the only necessary fields are `name`, `quantity`, `price_inc_tax`, and `price_ex_tax`. Other data may be included as necessary, but the data structure is different. See Create an Order for more information.

The following is an example of the product data structure for orders:

```
{
  /*... other order fields ...*/
  "products": [
    {
      /*... Custom Product ...*/
      "name": "Custom Product Name",
      "quantity": 1,
      "price_inc_tax": 10.98,
      "price_ex_tax": 10
    },
    {
      /*... Product with Variants ...*/
      "product_id": 118,
      "quantity": 1,
      "variant_id": 93
    },
    {
      /*... Product with Modifiers ...*/
      "product_id": 184,
      "product_options": [
        {
          "id": 200,
          "value": "180"
        },
        {
```

```
          "id": 230,
          "value": "192"
        }
      ]
    },
    {
      /*... Product with Variants and Modifiers ...*/
      "product_id": 184,
      "variant_id": 93
      "product_options": [
        {
          "id": 200,
          "value": "180"
        },
        {
          "id": 230,
          "value": "Custom Text"
        }
      ]
    }
  ]
  /*... other order fields ...*/
}
```

Customer data links

Unlike product data, customer data has much more straightforward fields to connect between your data source and BigCommerce. Specifically, customer data requires customer_id and address information. Beyond that, mapping is fairly straightforward.

Regardless of the specific structure used, the following example `billing_address` is the basis for all address options:

```
{
  "billing_address": {
    "first_name": "Jane",
    "last_name": "Doe",
    "company": "Company name",
    "street_1": "123 Main Street",
    "street_2": "Suite B",
    "city": "Austin",
    "state": "TX",
    "zip": "12345",
    "country": "United States",
    "country_iso2": "US",
    "phone": "987654321",
    "email": "example@example.com"
  }
}
```

There are two important callouts about the example above. First, the `zip` field is required within the billing address, as it is a mandatory field for most payment processors. It must be at least two characters and a valid postal code for the country. Second, shipping addresses optionally include a `shipping_method` field, which identifies the method by which the customer requested shipment. It can be any of your store’s enabled shipping methods.

When mapping customer data, there are five main cases to prepare for, as outlined in Complexity Considerations. Even so, they fall back on one of two main structures so far as mapping customer data.

Orders with consignments

Orders with consignments are treated as, essentially, pickup orders. All product data is paired directly with the relevant address data. In the default use-case, this address data is meant to pair with specific locations of your business and specific ‘pickup’ methods.

An example of this structure is outlined below.

```
{
  /*... other order fields ...*/
  "customer_id": 11,
  "billing_address": {
    /*... billing address fields ...*/
  },
  "consignments": {
    "pickups": [
      {
        "pickup_method_id":1,
        "pickup_method_display_name": "Pick Up",
        "collection_instructions": "Bring your ID",
        "collection_time_description": "9am - 6pm",
        "location": {
          /* location fields use a different structure */
          "name": "Location 1",
          "code": "LOCATION-1",
          "address_line_1": "123 Main Street",
          "address_line_2": "Suite 101",
          "city":"Austin",
          "state":"Texas",
          "postal_code":"78726",
          "country_alpha2":"US",
          "email":"location1@example.com",
          "phone":"+1 111-111-1111"
        },
        "line_items": [
          /*... product data ...*/
        ]
      }
    ]
  }
  /*... other order fields ...*/
}
```

The fields listed in this example may or may not align with your expected structure. Verify ahead that your fields follow the appropriate structure to use this data type.

Orders without consignments

Rather than using an order with consignments, order data may be split with product data in one array and shipping addresses in another, both of which are collections of objects of their specific data types. In this case, there is no direct mapping between products and the addresses to which they are shipped. As such, only older order data should be imported via this method.

An example of this structure is outlined below.

```
{
  /*... other order fields ...*/
  "customer_id": 11,
  "billing_address": {
    /*... billing address fields ...*/
  },
  "shipping_addresses": [
    {
      /*... other address fields ...*/
      "shipping_method": "UPS"
    }
  ],
  "products": [
    /*... product data ...*/
  ]
  /*... other order fields ...*/
}
```

All order data fields

Once product- and customer-specific data have been mapped properly, there are many other fields to manage in mapping. The table below contains all order fields (including for customer and product data) by name, including their data types, descriptions, and examples where possible.

There are fields listed in the Create an Order documentation not found in this table. Those fields that have been omitted are to be considered read-only and will throw server errors if they are included in the request.

Name	Data Type	Description	Example
`customer_id`	`integer`	The unique customer ID assigned by the server on customer creation. Sets base customer contact data for the order.	123
`status_id`	`integer`	The status of the order. Values are laid out in the table at Order Status	0
`date_created`	`string`	RFC-2822 formatted date when the order was placed.	Tue, 20 Nov 2012 00:00:00 +0000
`order_source`	`string`	Indicates how the order came in. This can be api, www, or other examples.	www
`order_is_digital`	`boolean`	Indicates whether all products in the order are digital. If `false`, at least one product is physical.	true
`billing_address`	`object`	Billing address for the	(see above)

Name	Data Type	Description	Example
		order. Treated as default address.	
shipping_addresses	array	Collection of shipping addresses for the order.	(see above)
products	array	Collection of products in the order data.	(see above)
consignments	object	Collection of pickups assigned to an order.	(see above)
fees	array	Collection of fees charged on the order.	
default_currency_code	string	The currency code the shopper will pay in. Only available when multi-currency is enabled.	USD
base_handling_cost	string[number]	Base handling cost before modifications are applied.	1.223
handling_cost_ex_tax	string[number]	Final handling cost, excluding tax.	2.223
handling_cost_inc_tax	string[number]	Final handling cost, including tax.	2.333
base_wrapping_cost	string[number]	Base gift wrapping cost before modifications are applied.	1.223
wrapping_cost_ex_tax	string[number]	Final gift wrapping cost, excluding tax.	2.223
wrapping_cost_inc_tax	string[number]	Final gift wrapping cost, including tax.	2.333
base_shipping_cost	string[number]	Base shipping cost before modifications are applied.	12.223
shipping_cost_ex_tax	string[number]	Final shipping cost, excluding tax.	12.223
shipping_cost_inc_tax	string[number]	Final shipping cost, including tax.	12.333
discount_amount	string[number]	Positive valued discount to the order subtotal.	12.223
refunded_amount	string[number]	Value refunded so far from the order total.	12.223
subtotal_ex_tax	string[number]	Order value after products and fees, but	32.221

Name	Data Type	Description	Example
		before discounts, excluding tax. Serves to override other fields.	
subtotal_inc_tax	string[number]	Order value after products and fees, but before discounts, including tax. Must be included if the _ex_tax field is.	32.333
total_ex_tax	string[number]	Final order value, excluding tax. Serves to override other fields.	19.998
total_inc_tax	string[number]	Final order value, including tax. Must be included if the _ex_tax field is.	20.011
payment_method	string	Display name of the payment method on the order.	Manual
payment_provider_id	string OR number	Payment provider's ID code. May be a numeric ID or a character code.	5
channel_id	integer	The unique channel ID for the sales channel on which the order was placed.	456
customer_locale	string	The 2, 3, 5, or 6 character locale active when the customer checked out.	en-US
items_shipped	integer	The number of items for which shipments have been created.	17
items_total	integer	The total number of items in the order (includes line items and quantities).	23
customer_message	string	Message provided in the order comments section at checkout.	Thank you for your help today!
staff_notes	string	Message provided to/by your staff. Not visible to customers.	Invoice number 123546
ebay_order_id	string	Order ID reported by	0

Name	Data Type	Description	Example
		eBay if the order originated there. `0` otherwise.	
`external_merchant_id`	`string`	Order ID from your data's source of truth. Only writable at time of order creation.	`null`
`external_order_id`	`string`	Order ID from an external system. Writable and updatable.	`null`
`external_source`	`string`	Indicates how the order originated. For all orders in your initial migration, this should have the value `M-MIG`.	M-MIG
`ip_address`	`string[ipv4]`	IP address of the customer, if available. Mutually exclusive with `ip_address_v6`.	127.0.0.1
`ip_address_v6`	`string[ipv6]`	IP address of the customer, if available. Mutually exclusive with `ip_address`.	2001:db8:3333:4444:5555:6666:7777:8888
`geoip_country`	`string`	Full name of the country where the order was placed. Determined from the IP address.	United States
`geoip_country_iso2`	`string`	ISO2 country code. Determined from the IP address.	US

FAQs

1. **What is the minimum data needed to create a migrated order?**
   See the minimal example in this document; at minimum you need `external_source`, `external_order_id`, `status_id`, `date_created`, `billing_address`, and `products`.
2. **Do product and customer IDs need to match those in BigCommerce?**
   Yes. Use the server-assigned `product_id` and `customer_id` for correct mapping.

3. **How do I handle custom products or products no longer in my catalog?**
   Use the custom product workflow—provide at least name, quantity, price. See Orders API - Create an Order.
4. **How do I migrate orders with consignments (pickups)?**
   Use the `consignments` object and omit `products` array. See example above.
5. **What if an order has multiple shipping addresses?**
   Use the `shipping_addresses` array. Each address can have a `shipping_method`.
6. **Do I need to include all possible fields in my order data?**
   Only required fields are mandatory, but including all relevant fields—even if null—prevents missing data issues.

Resources

- Product API Field Reference
- Orders API - Create an Order
- Order Complexity Considerations
- API Best Practices

Migration Phase

Just as with products, your order migration should include at minimum: testing, initial migration, delta migration. The initial testing process and delta migration will look very similar to those steps in the products migration process, so you can refer back to them as needed. (Load Complete Data) [/docs/start/migration/products/complete-migration] and (Go Live and Delta Migration)[/docs/start/migration/products/delta-migration].

There are some callouts specific to the order migration process, which will be addressed here, but otherwise, use your best judgement when considering how to manage your own work.

Load test data

This step is essential in verifying you've appropriately mapped your order data from your previous data source (ERP, CRM, or other) to BigCommerce. When you've finished your data preparation step we encourage you to do the following in loading test data:

- **Ensure ALL order data include the `external_source` field with the value `M-MIG` before beginning.**
- Turn off your order notifications in both your sandbox and production environments
- Prepare a representative sample from your existing data. Sanitize this data for testing, in case of errors or overlap.
- Load the sanitized data into your sandbox environment, logging any errors encountered.
- Verify the sample data loaded without issue.
- Identify inconsistencies with sample data and revise your mapping strategy accordingly.

Once you've successfully migrated your initial test data, move on to the full migration step.

Migrate your full order history

Once you've successfully tested your order migration process, verified mappings, and gained confidence in your approach, you're ready to proceed with loading your complete order dataset into BigCommerce.

Pre-Migration checklist

- Remove all extraneous data (test orders, unnecessary attributes) from your dataset
- ⚠️ Disable or turn OFF any third-party applications that might automatically ingest product data
- Verify the API rate limits for your plan level
- Scheduling your migration during off-peak hours, if possible, to minimize potential platform congestion

During migration

- Actively monitor for status error codes and implement your error handling protocols
- Track progress through logging to ensure you transfer all products correctly.
- Maintain a separate record of any failed transfers for later remediation

Post-Migration verification

- Compare order counts between your source platform and BigCommerce
- Spot-check a representative sample of orders across your complete history
- Check that you have transferred complex orders (refer to complexity considerations) correctly.
- Test order visibility by using customer impersonation to ensure appropriate account linkage.

Next Steps

- Once order data is successfully transferred, proceed with migrating design data (theme, etc.) if desired.
- Document your migration process thoroughly for future reference or troubleshooting
- Consider implementing a synchronization strategy or data freeze if you'll be operating multiple platforms simultaneously during transition

Estimated time to transfer

The following table provides approximations of migration times for various order history sizes. These estimates are based on platform limits and assumed order complexity. If you have particularly complex orders, your results will vary from these provisions.

Based on this information and the guidance provided above, plan your migration with enough time allotted to prevent issues.

Resources	**Estimated time (store with no live integrations)**
> 5,000	Around 30 minutes
> 40,000	Around 7 hours
> 300,000	Around 23 hours
> 500,000+	Recommended to work with our dedicated data migration services team on data processing

Delta migration

This step is only required if you are still receiving orders in a live environment during your migration. As with the testing phase, this step has a few callouts to be aware of. We recommend the following steps to ensure a successful delta migration with the fewest errors.

- Before initial migration, take a snapshot of your current order status.
- Before performing delta migration, take your store down for maintenance if possible.
- Compare snapshot data to newest data, and only update orders for which there is new data to report.
- If you took your store down for maintenance for delta migration, reactivate it once you have finished.
- Verify data successfully transferred via spot check.

It's important to note that your order delta migration will vary in size based on your store's activity. As such, you should plan enough time to account for the full scope of the project, following the table above.

FAQs

1. **How do I prepare test data for order migration?**
   Prepare a representative and sanitized sample of orders, ensure required fields are present, and load into your sandbox for validation.
2. **How can I avoid duplicate orders during migration?**
   Use unique order identifiers and check for existing orders before retrying API calls.
3. **What should I do if my migration fails or is interrupted?**
   Log all migrated order IDs and failed records. Use logs to identify and correct only affected orders.
4. **How do I handle delta migrations if new orders are placed during migration?**
   Take a snapshot before initial migration, compare to new data after, and only migrate newly placed or updated orders.
5. **How can I verify a successful migration?**
   Compare source and destination order counts, spot-check complex orders, and validate correct customer and product linkages.
6. **What are platform limits for order migration?**
   See API Rate Limits for details. For very large datasets, contact Data Migration Services.

Resources

- API Best Practices | BigCommerce Dev Center
- API Rate Limits
- Data Migration Services
- Orders API Reference