

# Product Migration Guide

## Product data migration: Best practices and process overview

Migrating product data is a foundational part of launching or replatforming an ecommerce store. This section provides practical guidance and best practices for migrating your product catalog into BigCommerce, focusing on accuracy, efficiency, and minimizing common migration errors.

## Why focus on data migration?

Product data migration affects core areas of your storefront, including product listings, search, inventory, pricing, and overall customer experience. Early planning and process validation help reduce the risk of errors and delays during launch. The following pages walk through the technical and operational steps of migration to support a smooth transition.

## Migration workflow pages

- [Prepare Your Migration Data](#)
- [Test a Limited Dataset](#)
- [Error Handling](#)
- [Load Complete Data](#)
- [Go Live and Delta Migration](#)

## Getting started

### Recommended preparatory tasks

- **Test your migration workflow early**  
Once you have defined your migration plan and scope, test your migration process in a non-production environment. Address any issues found during testing before moving to a full migration.
- **Align store settings and source data**  
Before importing product data, configure your store's [store settings](#) including product weight units, dimensions, currency, and timezone. Check that your source data matches these settings to avoid inconsistencies.
- **Configure categories and brands in advance**  
Categories and brands must be created in BigCommerce before importing products. Category and brand assignments are done by numeric ID, not by name. Prepare a mapping

between your source data and BigCommerce IDs to ensure products are assigned correctly. See the [Catalog API Reference](#) for details.

## **Additional resources**

For more technical details and API documentation, see:

- [Catalog API Reference](#)
- [API best practices](#)
- [API rate limits](#)
- [WebDAV file access](#)
- [Data migration services](#)

By following these recommendations and the linked migration workflow pages, you can manage your BigCommerce product migration with greater confidence and fewer disruptions.

## Prepare Your Migration Data

While some systems may allow a direct or near-direct connection between your current store and BigCommerce, using such a connection will likely cause unforeseen errors. For best results, some preparation will help reduce migration time and roadblocks.

### Identify Source Data

Export all data from your current platform. This will look different for all ecommerce solutions, so refer to your documentation for specific steps.

- For best results, use either a spreadsheet (CSV format is sufficient) or a SQL database if possible to ensure straightforward data mapping.
- Carefully review headers and data for accuracy and completeness after export. This will provide a higher level of familiarity with the data, informing next steps when mapping from your current data to BigCommerce.

Review the following BigCommerce API resources before starting your migration:

- [Catalog API](#) - the family of endpoints related to adding, updating, and deleting products and product data.
- [API Best Practices](#) - information on recommended usage.
- [API Rate Limits](#) - information on platform limits that will affect your migration.

To avoid potential issues and errors, verify each endpoint's limitations prior to your active migration.

Before migration, remove any data you don't want transferred to BigCommerce. Filtering this data before migration reduces complexity and saves time.

- Document the filtering criteria and back up any removed data. This minimizes the risk of accidental data loss and ensures you can replicate the process for future syncs.
- Examples of products that may need to be excluded: disabled, permanently out-of-stock, and products that don't easily map to BigCommerce automatically.

### Create a Mapping Plan

Some BigCommerce fields may map differently than in your current system, especially for large catalogs or simpler data models.

- The following fields are required for [product creation](#):
  - Name - the name of the product as displayed on the storefront
  - Type - whether the product is physical or digital
  - Weight - the shipping weight of the product (set to 0 for digital products)
  - Price - the base price of the product in the store's default currency
- The following product fields are read-only, unavailable for direct editing:
  - ID - the server-assigned product ID used in all automated BigCommerce operations

- Date Created - the timestamp saved when the product was created in BigCommerce, regardless of the method used
- Date Modified - the timestamp saved during the most recent product update
- Calculated Price - the price of the product once set adjustments are applied based on options and other features
- Base Variant ID - the server-assigned variant ID treated as default for the product

Some data fields will likely function differently in BigCommerce than in your current ecommerce solution or source of truth. A few examples include

- **Custom Fields**
  - In BigCommerce, custom fields serve as static filterable data for products.
  - Each custom field requires a field name and a specific value, both of which are text information.
  - Custom fields are displayed by default. Data not intended to display should be implemented with [Metafields](#) instead.
  - Some unstructured data such as notes, extra attributes, and non-native fields will need to be transformed into either custom fields or metafields prior to migration if they are to be preserved in BigCommerce.
- **Related Products**
  - Related products are assigned to a given product by product ID
  - Explicitly setting products as related requires direct assignment of IDs
- **URLs and 301 Redirects**
  - By default, BigCommerce creates [SEO Optimized, short](#) URLs for new products and categories.
  - If you prefer to maintain your existing URLs, you can migrate them directly using the `custom_url` field during product creation.
- Price Mapping
  - BigCommerce supports several distinct price fields, which may differ from your current source of truth:
    - Price - the price you normally charge for a single unit of the product
    - Retail Price - the product's manufacturer suggested retail price
    - Map Price - the minimum advertised price of the product
    - Sale Price - the price you are charging for the product while it's on sale
    - Cost Price - the price required for keeping a single unit of the product in stock
    - Calculated Price - the price of the product once set adjustments are applied based on options and other features
  - Carefully analyze your product pricing fields prior to migration to ensure correct mapping.

## Field Mapping Reference

Product options and channel assignments are not managed in the main products API.

- To assign channels, use [Channel Assignments](#).
- For information on product options, see [Product Modifiers](#), [Product Variants](#), and [Product Variant Options](#).

A list of the most commonly needed fields is provided in the following table.

Field Name	API Field Identifier	Data Type	Description
Name (required)	name	string	The name of the product, as displayed on the storefront.
Type (required)	type	string	The type of the product, either "physical" or "digital".
SKU	sku	string	The stock keeping unit. Only accepts numbers, letters, hyphens, underscores, and spaces.
Description	description	string	The product description as shown on product detail pages.
Weight (required)	weight	number	The shipping weight of the product, in the units configured in your store settings.
Width	width	number	The shipping width of the product, in the units configured in your store settings.
Depth	depth	number	The shipping depth of the product, in the units configured in your store settings.
Height	height	number	The shipping height of the product, in the units configured in your store settings.
Price (required)	price	number	The product's default price, provided in the store's default currency, with taxes included or excluded based on your store settings.

<b>Field Name</b>	<b>API Field Identifier</b>	<b>Data Type</b>	<b>Description</b>
Cost Price	cost_price	number	The product's cost price, provided in the store's default currency.
Retail Price	retail_price	number	The manufacturer suggested retail price, provided in the store's default currency, with taxes included or excluded based on your store settings.
Sale Price	sale_price	number	The product's current sale price, provided in the store's default currency, with taxes included or excluded based on your store settings.
Tax Class	tax_class_id	integer	The ID of the tax class assigned to the product. Tax classes are configured in your store settings. If you're using an automated tax provider, you may also need the <code>product_tax_code</code> .
Categories	categories	array	An array of the category IDs to which the product is assigned.
Brand ID	brand_id	integer	The brand to which the product is assigned.
Current Stock	inventory_level	integer	The current number of product units in stock, if tracked.
Low Stock	inventory_warning_level	integer	The stock level at which restocking becomes necessary.
Inventory Tracking	inventory_tracking	string	One of "product", "variant", or "none", indicating how or if inventory is tracked on the product.
Fixed Shipping Cost	fixed_cost_shipping_price	number	The per-unit shipping cost, if the product uses it. Set to 0 if the feature

Field Name	API Field Identifier	Data Type	Description
			is not used.
Free Shipping	is_free_shipping	boolean	Indicates whether the product is excluded from other shipping calculations. If <code>true</code> the product will ship for free and be excluded from other calculations unless your shipping settings explicitly ignore this field.
Is Visible	is_visible	boolean	Indicates whether the product is visible on the storefront. If <code>false</code> , the product will be inaccessible, even if it is assigned to a specific channel.
Is Featured	is_featured	boolean	Indicates whether the product will be included in the Featured Products section on your storefront.
Warranty	warranty	string	Warranty terms and conditions to be displayed on the product detail page. If no warranty is provided, the section will not display by default.
Bin Picking Number	bin_picking_number	string	The bin picking number used in a fulfillment workflow. If you don't use bin picking as a fulfillment process, this may be omitted.
UPC/EAN	upc	string	The UPC or other related product identifier.
Search Keywords	search_keywords	string	Comma separated search keywords that can be used by the built-in BigCommerce

Field Name	API Field Identifier	Data Type	Description
			storefront search to find the product. This field may be ineffective if you're using a <a href="#">headless commerce solution</a> or a custom search tool.
Sort Order	sort_order	integer	A 32-bit integer indicating the product's relative position on product listing pages when the Featured sort option is selected. Negative values place the product closer to the front, while positive values place the product closer to the back.
Product Condition	condition	string	One of "New", "Used", or "Refurbished", indicating the condition of the product
Show Product Condition	is_condition_sh own	boolean	Indicates whether the product condition is displayed on the storefront.
Page Title	page_title	string	The SEO title for the product detail page.
Meta Keywords	meta_keywords	array	An array of SEO keywords used for page ranking on some search engines.
Meta Description	meta_descriptio n	string	The SEO description for the product detail page.
Product URL	custom_url	object	An object indicating the state of the product's custom URL and whether a redirect is necessary.
Global Trade Number	gtin	string	The <a href="#">Global Trade Item Number</a> of the product used by some sales channels for proper tracking and identification.

Field Name	API Field Identifier	Data Type	Description
Manufacturer Part Number	mpn	string	The manufacturer part number of the product, used to guarantee that customers are viewing the correct item.
<a href="#">Custom Fields</a>	custom_fields	array	An array of objects that include the name and value of each custom field associated with the object.
<a href="#">Images</a>	images	array	An array of objects containing image data.
<a href="#">Videos</a>	videos	array	An array of objects containing video data.
Variants	variants	array	An array of objects containing variant data.
<a href="#">Bulk Pricing</a>	bulk_pricing_rules	array	An array of objects containing bulk pricing data.
Related Products	related_products	array	An array of numeric product IDs for related products.

## FAQ

### 1. What are the most efficient ways to export product data from my legacy store for migration into BigCommerce?

Use your source platform's native export tools to generate CSV or SQL files, or leverage its API if available for direct data extraction. Choose the format that best matches your mapping and transformation needs. Always verify data accuracy post-export.

### 1. How do I map complex or custom product fields from my source system to BigCommerce's schema?

Identify all custom fields in your source data and review BigCommerce's [custom fields](#) and [metafields](#). Map each source field to the most appropriate BigCommerce field, transforming formats or structures as needed. Document your mappings for consistency.

### 1. What is the best way to handle images and digital assets during product migration?

Ensure all product images and files are organized and accessible, with URLs or file names correctly referenced in your data. Upload images via the BigCommerce API or through bulk import features, and validate that each product record correctly links to its images after migration.

### 1. How do I handle product variants, options, or bundles when the source and target systems structure them differently?

Analyze how your source platform represents variants, options, and bundles. In BigCommerce, use the [variants](#), [modifiers](#), and related APIs to recreate these relationships.

- 1. How should I document and track my data mapping and transformation rules for future reference or audits?**

Maintain a mapping spreadsheet or document that clearly records each source field, its transformation logic, and the target BigCommerce field. Include examples and version your mapping documentation to track changes over time.

- 1. What are common issues encountered during product data migration to BigCommerce, and how can I troubleshoot them?**

Common issues include data validation errors, missing required fields, mismatched categories, or broken image links.

Troubleshoot by reviewing error logs, running validation scripts pre-import, and performing test imports with small data batches to catch issues early.

- 1. How do I ensure SEO continuity, including URLs and redirects, during migration?**

Map your legacy URLs to BigCommerce's structure using the `custom_url` field for products and categories. Set up 301 redirects for any changed URLs using the [Redirects API](#) or admin tools to preserve SEO rankings and avoid broken links.

## Resources

- [BigCommerce Catalog API Documentation](#) — Reference for all endpoints related to product data
- [API Best Practices](#) — Recommended approaches for robust API use
- [API Rate Limits](#) — How to avoid and handle API rate limiting

## Test a Limited Dataset

In order to properly prepare data for migration, data validation and testing are essential steps to provide improved efficiency during the full implementation.

## Prepare a Representative Sample

Before attempting to load any data, be sure to familiarize yourself with the [BigCommerce API Best Practices and Rate Limits](#).

To ensure a smooth execution during migration, prepare a subset of your catalog to perform a partial migration as a test. This should be around 50 to 1000 data records, depending on the size and complexity of your catalog, that cover a broad range of potential product configurations

If your catalog contains any of the following, ensure that your representative sample contains at least one product with each to test the associated BigCommerce APIs.

- SKU modifying product options - Product Variants and Product Variant Options
- Non-SKU modifying product options - Product Modifiers
- Static data visible to customers - Custom Fields
- Static data **not** visible to customers - Metafields

Select a mix of data for testing, intentionally including records with inconsistencies, missing fields, edge-case values, or legacy formatting, as these samples are most likely to surface real-world migration issues.

As incorrect order migration can directly impact billing, it is imperative to fully understand these distinctions and their migration implications.

Conduct all initial data migration tests in a sandbox environment if possible.

Once you've identified a representative sample, begin the partial migration.

If you use Postman as your primary API client, you can use the [BigCommerce V3 API library](#).

1. Create one or more API accounts with [appropriate scopes](#) to make requests to the Catalog API.
  - Product creation requires API authentication keys to ensure secure data management.
2. Create base products.
  - Use the "Create a Product" endpoint to establish individual product records, making sure to associate the product ID with your representative dataset.
  - For optimal error handling during large-scale operations, the best practice is to create individual products rather than using batch creation, even though it is supported.
  - Product variants created simultaneously with the base product will default to display as rectangle lists. For more information on variant display options, see [Variants and Modifiers \(Help Center\)](#).
3. After base product creation, add additional attributes, images, and other data.

- While some data may be loaded simultaneously with the base product, performing the operation in multiple steps ensures clear error logging and a more consistent product creation experience.

Each Catalog API endpoint’s specification is accompanied by one or more example request bodies to clarify format and structure. For more information, review the [Catalog API Reference](#).

## Performing Data Validation and User Acceptance Testing (UAT)

Given potential datatype mismatches and platform limitations, data validation is a key step in ensuring a successful migration. After migrating your representative dataset, you should

- Manually verify a small subset of records against your source of truth to catch any discrepancies; and
- Automate data validation by retrieving migrated data using the `Get All Products` endpoint and comparing it programmatically to your source dataset.

By combining manual and automated validation methods, you ensure that your data is complete, accurate, mapped as intended, and behaves as expected.

- Consider using [jsonschema](#) (Python) or [deep-diff](#) (Node.js) for schema validation and record comparison.
- Leverage CSV/JSON diff tools to automate dataset checks. Some such applications include the following:
  - [json-diff](#): A widely-used command-line tool and library for comparing JSON files, highlights structural and value differences.
  - [csvdiff](#): A Python library and CLI tool that compares CSV files and highlights row and cell-level differences for tabular data.
  - [daff](#): A versatile data diff tool supporting CSV/TSV formats, available as a command-line tool and with a web interface, ideal for comparing tables and spreadsheets.

### User Acceptance Testing Resource Checklist

Once you’ve finished data validation on your sample dataset, user acceptance testing (UAT) is critical. This step ensures your data migration mapping is accurate and functions as expected in both the BigCommerce control panel and storefront.

As part of the UAT process, review the following product fields and storefront behaviors:

Field/Area	What to Review
<b>Product Name and SKU</b>	Ensure product names and SKUs are accurate and mapped correctly.
<b>Product Prices</b>	Confirm product, MSRP, and sale prices are correct in the control panel and storefront. Check price change rules if applicable.
<b>Product/Variant Visibility</b>	Verify visibility settings (hidden/visible) and purchasing behavior (purchasable/not

Field/Area	What to Review
	purchasable) for both products and variants.
<b>Inventory Tracking</b>	Check inventory tracking settings and inventory levels for accuracy.
<b>Options &amp; Variants</b>	Confirm product options, variant SKUs, and option types (swatch, dropdown, etc.) are structured and displayed as intended.
<b>Custom Fields</b>	Ensure custom fields are present, correctly populated, and that no unwanted fields have been migrated.
<b>Descriptions</b>	Review product descriptions for correct formatting, including HTML, images, videos, PDFs, and links. Ensure links resolve to the correct BigCommerce URLs.
<b>Images / Videos</b>	Check that all product and variant images and videos appear properly in both the control panel and storefront.
<b>Metadata</b>	Review SEO metadata for completeness and correctness.
<b>Product URL Links</b>	Confirm product URLs are formatted according to your migration and cut-over plan.
<b>Recommended Products</b>	If used, verify recommended/related products are set up as expected. <a href="#">See Product Panels documentation</a>
<b>Categories</b>	Ensure categories are created, named, visible, and assigned to the correct channels or storefronts.

Record your findings and any issues in a tracking sheet or ticketing system. Plan for iterative corrections and re-tests as needed before migrating your full catalog.

## Special Considerations

To improve the overall migration experience, the following optional steps will allow more flexibility in automation.

- **Proactively Managing API Traffic**
  - During peak platform usage times, the BigCommerce API may return HTTP 429 responses. Implement rest-and-retry logic to handle rate limiting effectively.
  - To reduce the likelihood of HTTP 429 responses, disable any third-party API applications on your store during migration, as the BigCommerce API does not distinguish between individual clients when managing rate limits on a single store.
- **Error Handling and Reporting**
  - Implement comprehensive error reporting to track failed resource transfers.

- Save all API requests during testing to aid in building clarity when unexpected outcomes occur.
- Follow error handling guidance based on specific error types. See Error Handling for more information.
- **Pre-Plan for Data Integrity and Availability**
  - Begin with a smaller dataset to verify functionality and data quality while limiting API requests and retries in testing.
  - External resources that are unavailable via public HTTP requests cannot be fetched via the BigCommerce API. Upload images and other resources via WebDAV, using resource links local to BigCommerce during product creation.
  - Ensure your test dataset includes edge cases such as products with maximum/minimum field values, missing optional data, special characters, and legacy formats to reveal potential issues early.

## FAQ

1. **How can I automate the generation of representative and edge-case test datasets for my catalog?**

Use scripting languages (such as Python or Node.js) to extract real records from your source system, and programmatically mutate copies to include edge cases (e.g., missing fields, maximum/minimum values, special characters). Consider sampling actual problematic records from production logs or validation failures.

1. **What are the best practices for scripting API calls to validate data integrity post-migration?**

Automate GET requests to retrieve migrated data from BigCommerce, then compare the results field-by-field with your source dataset. Log differences and use schema validation to catch datatype or structure mismatches.

1. **How do I programmatically compare source and destination data for consistency after a partial migration?**

Export both source and destination datasets as structured data (JSON/CSV), normalize formats, and use diff tools or custom scripts to check for field-level matches and missing/extra records.

1. **How can I handle and retry failed API requests efficiently when rate limits or transient errors occur during test runs?**

Implement exponential backoff and retry logic in your scripts for HTTP 429 (rate limit) or 5xx errors, and monitor response headers for rate limit information to throttle requests as needed.

1. **What is the recommended approach for cleaning up test data (including products, images, and related resources) after validation in a sandbox or production environment?**

Track IDs of all created test resources during migration, and use API DELETE calls or batch operations to remove them once tests are complete. For production, use clear naming conventions or tagging to identify test data for cleanup.

## Resources

- [BigCommerce Catalog API Reference](#)
- [Handling API Errors](#)
- [API Rate Limits Guide](#)
- [Creating a BigCommerce Sandbox Store](#)
- [Using Postman with BigCommerce APIs](#)
- [Community Thread: Data Validation Scripts & Tips](#)
- [WebDAV Overview and Usage](#)
- [Sample Product Import CSV](#)
- [API Accounts and OAuth Scopes](#)

## Error Handling

API operations at scale often return errors instead of successfully completing a given request. The BigCommerce API has two primary classes of error messages that can impact data migration: 4XX and 5XX.

For an in-depth list of the various status codes, including errors, and what they mean, see [API Status Codes](#).

### 5xx Error Codes

5XX status codes indicate server-side issues. These errors generally cannot be resolved by the client. For transient errors such as 500, 502, 503, or 504, implement exponential back-off retries, up to a set limit (e.g., 10 attempts), increasing the wait time between each retry. However, other 5XX errors like 501 or 505 are persistent and should be logged and investigated rather than retried.

While rate limiting typically returns a 429, you may occasionally receive a 500 for similar scenarios. Handle these with the same retry strategy. For more on handling server errors, see [API Request Architecture](#).

### 4xx Error Codes

4XX errors indicate malformed requests or invalid data. For automated operations, log and skip most 4XX errors for later review, but note that some require immediate attention.

Implement robust rate limiting for automated API calls to avoid exceeding [platform rate limits](#).

#### **Status code 429**

The API returns this error when you exceed the platform rate limits.

- Once the API returns a 429 status, it will likely return 429 for subsequent requests unless the rate limit window elapses.
- As a means to avoid this error, BigCommerce APIs include [rate limit headers](#) in responses that provide the information necessary for you to adjust call rates.
- If you do receive a 429 status in your response, use either the X-Rate-Limit-Time-Reset-MS or the X-Retry-After header from the response to wait before retrying the request.
- For example, in PHP:

```
$milliseconds = $response->getHeader("X-Rate-Limit-Time-Reset-MS");  
usleep($milliseconds * 1000);
```

#### **Status code 404**

The API returns this error either when a request's path does not match an existing endpoint, when a provided parameter does not match existing data, or less commonly when the method of the request is invalid.

- If a 404 status is returned for individual API calls, this usually means that some referenced data does not match existing data. In general, this should be logged and skipped, including the exact path, body, and response of the request to investigate the cause later.

## **Batch Endpoints**

When employing batch endpoints, for example [Update Products \(Batch\)](#), there may be a single item that causes the whole request to fail. Often, the response for batch endpoints will include the index of the failed item. Use this index to remove the item from the request, then retry.

Do not assume the entire batch succeeded or failed. Process successful items, and retry or log errors for failed ones.

## **Status code 422**

The API returns this error when a request is formatted incorrectly or contains invalid data.

- A 422 error should typically only occur during testing. If you see it during a large data migration, check your dataset and the data mapping for potential problems.
- You can likely log and skip a single 422 error. However, repeated 422 errors indicate a persistent problem that you must fix before continuing the migration.

<Tabs items={[Example Bad Request, Example 422 Response]}>

Below is an example of a malformed request to the endpoint [Update Products \(Batch\)](#). Due to the requirements of the endpoint, a 422 status code is expected from this request.

```
[
  {
    "name": "test1"
  },
  {
    "name": "test2"
  }
]
```

For this endpoint, the error response doesn't list the specific items that failed. Therefore, you'll have to separate the request into individual items and resend them.

In this particular case, the errors listed apply to each item, but that is not likely to be the case in general.

```
{
  "status": 422,
  "title": "Missing Required Fields",
  "type": "https://developer.bigcommerce.com/api-docs/getting-started/api-status-codes",
  "errors": {
    "name": "Please provide a name.",
    "price": "Please provide a price.",
    "type": "Please provide a type.",
    "weight": "Please provide a weight."
  }
}
```

## FAQs

### 1. What is the recommended strategy for implementing exponential backoff retries with the BigCommerce API?

Use exponential backoff for retrying transient errors, starting with a short delay and doubling it after each failed attempt, up to a maximum number of retries (e.g., 10). Add a small random jitter to each delay to reduce collision risk. Always respect any `X-Retry-After` or other rate limit headers from the response.

### 1. What headers should I monitor to avoid hitting rate limits, and how do I interpret them?

Monitor response headers such as `X-Rate-Limit-Requests-Left`, `X-Rate-Limit-Time-Reset-MS`, and `X-Retry-After`. These indicate how many requests remain and when you can resume making requests. Adjust your request rate accordingly to avoid being throttled or blocked. For more information, see [BigCommerce Specific Headers](#)

### 1. What should I do if I repeatedly receive 422 errors for apparently valid data?

Repeated 422 errors suggest a problem with your data or mapping logic. Pause your migration, review your data transformation, and ensure all required fields and formats match the API's expectations before resuming.

### 1. How do I handle network errors or timeouts when communicating with the API?

Implement generic error handling for network issues and timeouts. Retry failed requests using exponential backoff, and ensure your code checks for and gracefully handles situations where no response or a malformed response is received.


## Resources

- [API Status Codes](#)
- [API Request Architecture – 5xx Errors](#)
- [API Rate Limits](#)
- [Catalog API Error Responses](#)
- [API Client Libraries](#)
- [Logging and Monitoring Best Practices](#)
- [BigCommerce Dev Community](#)
- [MDN HTTP Status Codes](#)
- [Exponential Backoff and Jitter \(AWS Blog\)](#)

## Load Complete Data

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

### Pre-Migration checklist

- Remove all extraneous data (hidden products, unnecessary attributes) from your dataset
-  Disable or turn OFF any third-party applications that might automatically ingest product data
- Verify you have sufficient API rate limits for your plan level
- Consider scheduling your migration during off-peak hours to minimize potential platform congestion  
If your source catalog undergoes regular updates which may coincide with your migration, consider implementing a data freeze or define a clear “cut-off” time to avoid missing late changes.
- Backup your source data before proceeding

Use unique product identifiers to check if a product exists before retrying an API call to create it.

For batch uploads, log each product's status whenever possible to ensure complete migration.

### 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 BigCommerce enforces strict platform limits, such as maximum variants per product, image sizes, and total catalog size. Exceeding these can cause migration jobs to fail or result in incomplete imports. Review BigCommerce’s documentation on limitations before migration to avoid unexpected errors.

### Post-Migration verification

- Compare product counts between your source platform and BigCommerce
- Spot-check a representative sample of products across different categories and types
- Check that you have transferred complex products (those with variants, custom fields, or multiple images) correctly.
- Test product visibility and searchability on your storefront
- Confirm pricing, inventory levels, and product relationships are accurate

## Next Steps

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

## Estimated time to transfer

The following table provides approximations of migration times for various catalog sizes. These estimates are based on platform limits and assumed catalog complexity. If you have a particularly complex catalog, 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 with variants	Around 1 hour
> 40,000	Around 15 hours
> 300,000	Around 100 hours
> 500,000+	Recommended to work with our dedicated data migration services team on data processing

## FAQ

- 1. How can I avoid creating duplicate products or corrupting my data if the migration is interrupted or retried?**

Ensure your migration script checks for existing products (by unique identifier, like SKU) before creating or updating records, so that repeated or retried requests don't result in duplicates. Always use migration logs to track the status of each product upload and re-run only failed or incomplete batches.

- 1. What should I do if the migration fails or needs to be reverted?**

Always make a full backup of your source data before starting the migration. If you encounter issues, use your migration logs to identify and remove or correct only the affected products in BigCommerce. For large rollbacks, you may need to restore from your backup or script deletions based on the IDs created during the migration. Document the rollback steps taken for future reference.

- 1. How do I ensure that products, categories, brands, and images are properly linked after migration?**

Migrate related entities (such as categories, brands, and images) before importing products. Maintain a mapping between original and BigCommerce-assigned IDs for these entities, and update

product references accordingly to ensure all links remain valid. After migration, verify all relationships and assignments are intact in the BigCommerce store.

### 1. **How can I verify that my migration is complete and accurate?**

In addition to product counts, use your migration logs to reconcile all transferred data. Perform spot checks on complex or critical products, and check that all related entities and references are present. Use API queries or platform reports to systematically verify catalog integrity.

## **Resources**

- [API Best Practices | BigCommerce Dev Center](#)
- [API Rate Limits](#)
- [Monitoring API Usage](#)
- [Data Migration Services](#)

## Go Live and Delta Migration

While a complete "data freeze" during migration is ideal, we recognize that business continuity requirements often make this impractical. For this reason, a well-structured delta migration plan is essential.

### Delta Migration Timeline

- Schedule your delta migration to occur no more than 2 weeks after your BigCommerce go-live date
- Consider multiple delta migrations for high-volume stores (e.g., 24 hours post-launch, 1 week post-launch, and final sweep)
- Coordinate delta migrations during low-traffic periods to minimize impact on operations

### Delta Migration Process

- Utilize the same tooling and processes established during your initial migration
- Focus only on net-new or modified data since your initial migration
- Note that the Create Products API will automatically reject duplicate data that exactly matches existing records
- For updated products, use the Update Product endpoint rather than Create Product to avoid duplication

### Data Reconciliation Strategy

- Implement a reliable method to identify which products were created or modified since your initial migration
- Consider using timestamps, database flags, or changed-record logs to identify delta content
- Create a verification process to ensure no critical data is missed between migrations

### Business Operations During Transition

- Establish clear protocols for order processing during the transition period
- Determine how inventory will be managed across both platforms until migration is complete
- Create a communication plan for staff to understand which system is authoritative at each stage

### Contingency Planning

- Develop rollback procedures in case critical issues arise during go-live
- Establish decision criteria for when to activate contingency plans
- Prepare communication templates for both internal teams and customers if delays occur

## Post-Migration Maintenance

- Implement a regular data validation schedule to ensure ongoing data integrity
- Document any manual adjustments made during the delta migration process
- Create a standard operating procedure for future data synchronization needs

## FAQ

### 1. How do I identify which products or data have changed since the initial migration?

Use timestamps, change logs, or database flags from your source system to extract only records that have been created or modified since the initial migration. This ensures that only delta changes are considered for transfer.

### 1. What is the safest way to prevent duplicate records during delta migration?

Always check for existing records in BigCommerce using unique identifiers (such as SKU or product ID) before creating or updating records. Use the Update Product endpoint for modified records and Create Product only for new items.

### 1. How should I handle conflicting updates between the source system and BigCommerce during the delta window?

Establish clear rules for conflict resolution, such as “latest timestamp wins,” or flagging discrepancies for manual review. Communicate these rules to all stakeholders prior to migration.

### 1. What should I do if an error or failure occurs during delta migration?

Rely on detailed logging and migration status tracking to identify failed records. Retry only the failed operations using your logs, and ensure that your process is idempotent to avoid duplicate or partial updates.

### 1. How can I verify that all intended changes were successfully migrated to BigCommerce?

After the delta migration, compare the updated product counts, use spot checks on high-importance records, and reconcile migration logs with both source and destination systems to ensure that all intended changes are present.

## Resources

- [BigCommerce API Documentation](#)
- [API Rate Limits | BigCommerce Dev Center](#)
- [Best Practices for Data Migration | BigCommerce](#)
- [BigCommerce Webhooks Guide](#)
- [Monitoring API Usage](#)
- [Data Migration Services | BigCommerce](#)
- [BigCommerce Catalog API Reference](#)