



Elastic Path™ Commerce 6.1.1

Import-Export Tool User Guide

Copyright © 2009 Elastic Path Software, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means (photocopying, electronic, mechanical, recording, or otherwise), without permission in writing from Elastic Path Software, Inc.

Elastic Path™ and the Elastic Path logo are trademarks or registered trademarks of Elastic Path Software, Inc.

All other trademarks are the property of their respective owners.

Contents

| | |
|---|----|
| Introduction | 6 |
| What's New in 6.1.1? | 6 |
| System Requirements | 6 |
| Installation | 6 |
| Concepts | 7 |
| Exportable Object Types | 7 |
| Catalogs | 7 |
| Categories | 7 |
| Products | 8 |
| Promotions | 10 |
| System Configuration Settings | 10 |
| Import Conflict Resolution | 10 |
| Import Strategies | 10 |
| Dependency Conflicts | 13 |
| Integrity Constraints and Import Order | 15 |
| Exporting Data | 16 |
| Configuring Environment Settings | 16 |
| Configuring Export Options | 17 |
| Exporter Configuration | 17 |
| Packager Configuration | 18 |
| Delivery Configuration | 18 |
| Creating the Search Configuration File | 18 |
| Running the Batch Script to Export Data | 19 |
| Export Package Structure | 19 |
| Importing Data | 21 |
| Configuring Environment Settings | 21 |
| Configuring Import Options | 21 |
| XML Validation Configuration | 22 |
| Import Strategy Configuration | 22 |
| Packager Configuration | 23 |
| Retrieval Configuration | 23 |

| | |
|---|----|
| Running the Batch Script to Import Data | 23 |
| Appendix A: Export Configuration | 24 |
| delivery | 24 |
| Parent elements | 24 |
| Attributes | 24 |
| Child elements | 24 |
| exportconfiguration | 24 |
| Parent elements | 24 |
| Attributes | 24 |
| Child elements | 25 |
| exporter | 25 |
| Parent elements | 25 |
| Attributes | 26 |
| Child elements | 26 |
| include | 26 |
| Parent elements | 26 |
| Attributes | 27 |
| Child elements | 27 |
| option | 28 |
| Parent elements | 28 |
| Attributes | 28 |
| packager | 28 |
| Parent elements | 28 |
| Attributes | 28 |
| Child elements | 28 |
| transformer | 29 |
| Parent elements | 29 |
| Attributes | 29 |
| Child elements | 29 |
| transformerschain | 29 |
| Parent elements | 29 |
| Attributes | 29 |
| Child elements | 29 |
| type | 29 |
| Parent elements | 30 |
| Attributes | 30 |

| | |
|---|----|
| Child elements | 30 |
| Appendix B: Import Configuration | 31 |
| dependentelement | 31 |
| Parent elements | 31 |
| Attributes | 32 |
| Child elements | 32 |
| dependentelements | 32 |
| Parent elements | 32 |
| Attributes | 32 |
| Child elements | 33 |
| importconfiguration | 33 |
| Parent elements | 33 |
| Attributes | 33 |
| Child elements | 33 |
| importer | 33 |
| Parent elements | 33 |
| Attributes | 34 |
| Child elements | 34 |
| importstrategy (child of importconfiguration) | 34 |
| Parent elements | 35 |
| Attributes | 35 |
| Child elements | 35 |
| importstrategy (child of importer) | 35 |
| Parent elements | 35 |
| Attributes | 35 |
| Child elements | 35 |
| method | 35 |
| Parent elements | 36 |
| Attributes | 36 |
| Child elements | 36 |
| packager | 36 |
| Parent elements | 36 |
| Attributes | 36 |
| Child elements | 36 |
| retrieval | 36 |
| Parent elements | 37 |

| | |
|------------------------------------|----|
| Attributes | 37 |
| Child elements | 37 |
| source | 37 |
| Parent elements | 37 |
| Attributes | 37 |
| Child elements | 37 |
| transformer | 37 |
| Parent elements | 37 |
| Attributes | 38 |
| Child elements | 38 |
| transformerschain | 38 |
| Parent elements | 38 |
| Attributes | 38 |
| Child elements | 38 |
| type | 38 |
| Parent elements | 38 |
| Attributes | 38 |
| Child elements | 39 |
| xmlvalidation | 39 |
| Parent elements | 39 |
| Attributes | 39 |
| Child elements | 39 |
| Appendix C: Search Configuration | 40 |
| epql | 40 |
| Parent elements | 40 |
| Attributes | 40 |
| Child elements | 40 |
| searchconfiguration | 40 |
| Parent elements | 40 |
| Attributes | 40 |
| Child elements | 41 |
| Appendix D: Query Language | 42 |
| Query Format | 42 |
| Case Sensitivity | 43 |
| Data Types and Supported Operators | 43 |
| Supported Fields | 44 |

| | |
|---------------------------------------|----|
| Product Fields | 44 |
| Category Fields | 44 |
| Catalog Fields | 44 |
| Promotion Fields | 45 |
| System Configuration Setting Fields | 45 |
| Localized Fields and Attributes | 45 |
| The Price Field | 46 |
| System Configuration Setting Metadata | 46 |
| The Setting Namespace Field | 46 |
| Combining Expressions | 47 |
| Limiting the Result Set Size | 47 |
| Specifying the First Match | 47 |
| Appendix E: Troubleshooting | 49 |

Introduction

The Import-Export tool lets you import and export data to and from an Elastic Path deployment. Typical uses include:

- Migrating data between Elastic Path environments (development to staging, staging to production, etc.)
- Exporting data to third party applications
- Importing data from third party applications.

This document explains how to use the Import-Export tool.

What's New in 6.1.1?

The following features are new in version 6.1.1 of the Import-Export tool:

- You can now import and export system configuration settings.
- You can now import and export promotions.
- Error codes are now used to provide feedback when errors occur.

System Requirements

- Elastic Path 6.1.1 or later

Installation

Create a directory and extract the Import-Export distributable archive to that directory.

Important: Make sure that the appropriate JDBC driver libraries are on the class path of the Import-Export tool. Files placed in the `libs` subdirectory are automatically added to the classpath. For example, if your Elastic Path deployment is using a MySQL database, copy the MySQL JDBC connector jar file to the `libs` subdirectory.

Concepts

There are many complex interdependencies between the various objects in Elastic Path. When moving data between Elastic Path deployments, data exported from one deployment may cause conflicts with data in the deployment where you want to import it.

Before using the Import-Export tool, you need to understand how objects in Elastic Path relate to one another. This section describes those objects and their relations, and provides an overview of the strategies you can use to prevent import conflicts.

Exportable Object Types

The following objects can be exported from Elastic Path:

- products
- categories
- catalogs
- promotions
- system configuration settings.

Many of these objects are linked to other objects in the system. Some of those linked objects are required dependencies. During export, these dependencies are retrieved and exported automatically. For example, a product must be associated with at least one category and at least one catalog. So, when a product is exported, all its associated categories and catalogs are also exported.

Other linked objects are optional dependencies. You can specify whether you want to include optional dependencies. For example, assets, such as images, are optional dependencies, and you can choose whether you want to include them when exporting objects.

Catalogs

Catalogs are located at the top of the product hierarchy. They define the SKU options and brands that they can contain. A catalog may also include assets, such as images.

Catalogs do not have any required dependencies. When a catalog is exported, the associated assets may be optionally exported as well.

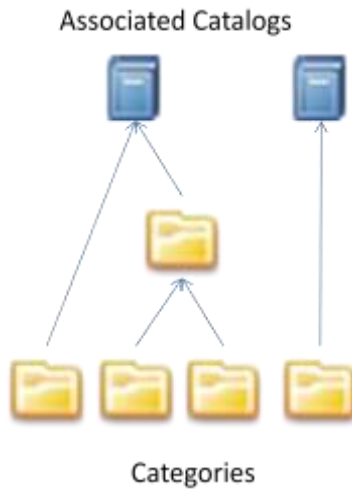


Note: Elastic Path supports virtual catalogs, which can contain products from multiple catalogs. For the purpose of the Import-Export tool, virtual catalogs are the same as regular catalogs.

Categories

Categories are located below catalogs in the product hierarchy. They are used to organize products within a catalog, or within another category. They contain category attributes that define how products are represented within the category.

When a category is exported, the associated catalogs and parent categories (if any) are automatically exported.

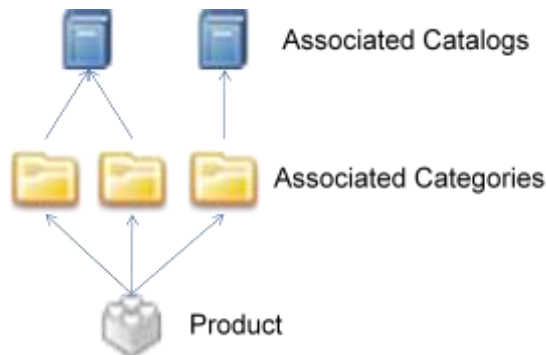


Categories may also have associated assets, which can optionally be exported as well.

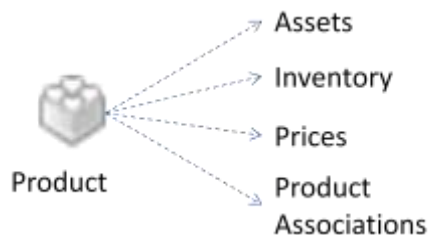


Products

A product is an object that can be purchased. It is always associated with one or more categories, so when a product is exported, its associated categories are automatically exported, as well as the catalogs associated with those categories.



Products may also have associated assets, prices, inventory, and product associations, which can optionally be exported as well.



Product Merchandising Associations

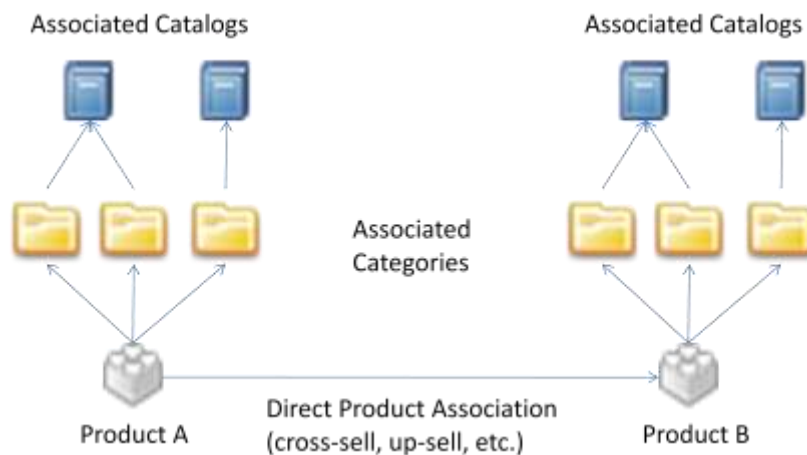
When exporting products, the product merchandising associations (cross-sells, up-sells, warranties, etc.) can be exported as well.

Note: Only products that are targets in associations with the initial product are exported. (The target of an association is the product that appears in the source product's merchandising area.)

There are two ways to export product merchandising associations:

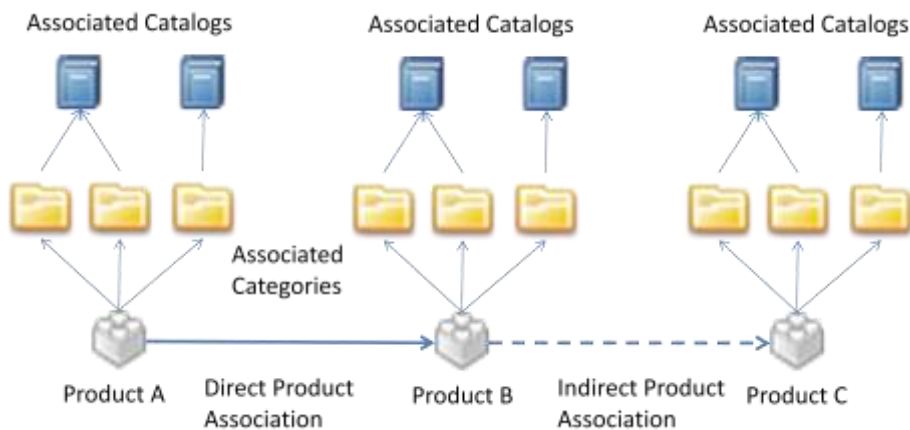
- Direct only
- Direct and indirect.

If you select the direct only option, only the products that are directly associated will be exported. For example, if you export product merchandising associations for Product A, and Product B is associated with Product A as a cross-sell, Product B is exported as well.



If you select the direct and indirect option, all products directly associated with the selected products are exported. Products associated with the associated products are also exported.

For example, if you export product merchandising associations for Product A with the direct and indirect option, Product B is also exported because it is associated with Product A. Next, if Product B has any associated products, those are exported as well, in this case, Product C. If Product C has any associated products, those are also exported, and so on.



Important: This option can potentially match a large amount of data.

Promotions

A promotion is a marketing tool used to increase sales of a certain product or set of products. There are two types of promotions:

- *Shopping cart promotions*, which are applied to items after they are added to shopping carts. Shopping cart promotions can be restricted to certain customers, based on configurable eligibilities.
- *Catalog promotions*, which are applied to specific products in the catalog.

When exporting promotions, you can optionally include their associated eligibilities (in the case of shopping cart promotions), conditions, and actions.



System Configuration Settings

System configuration settings determine the behavior of an Elastic Path deployment. These settings are stored in the database and can be configured through the Commerce Manager client.

Import Conflict Resolution

When importing data into an Elastic Path deployment, there is the possibility of conflicts. If an object being imported has the same ID as an object in the target deployment, you need to indicate which one you want to keep. You can configure how the Import-Export tool resolves conflicts by specifying import strategies. You can also configure how dependencies of those objects are handled in the event of conflicts.

Import Strategies

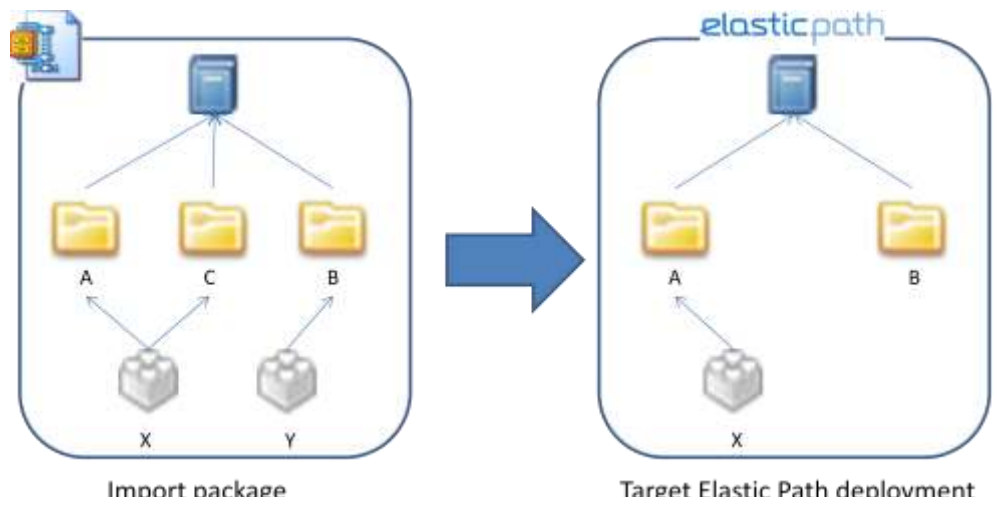
You can configure different import strategies for each type of object (catalog, category, product) being imported.

Supported import strategies are:

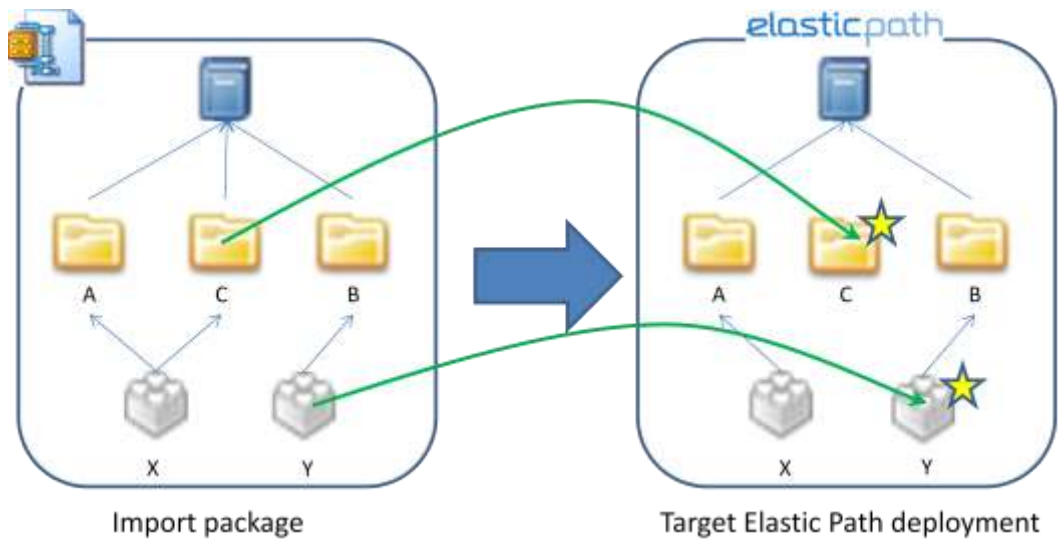
- *Insert*. This option only imports new objects. If the ID of an object being imported matches the ID of an existing object, the object is not imported.
- *Update*. This option only imports objects whose IDs match existing object. The existing object is overwritten with the object being imported.
- *Insert or Update*. Combines the *Insert* and *Update* options. This option imports new objects and overwrites existing objects when objects being imported have the same IDs as existing objects.

For example, the following diagram shows an Elastic Path deployment that contains two categories, A and B. Category A contains one product, product X.

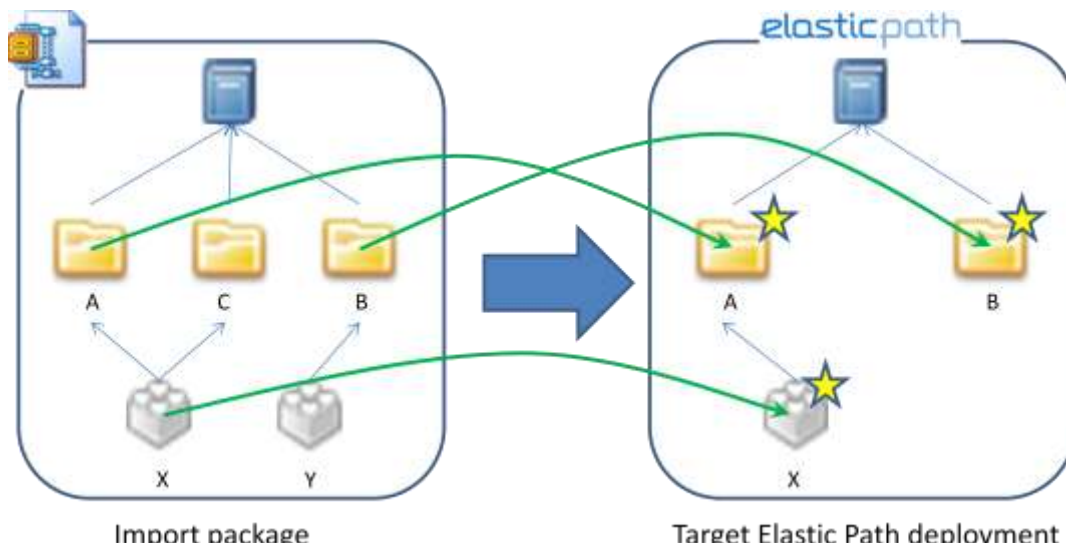
The package being imported contains categories A and B, and a new category named C. It also contains a new product, product Y, which is associated with category B.



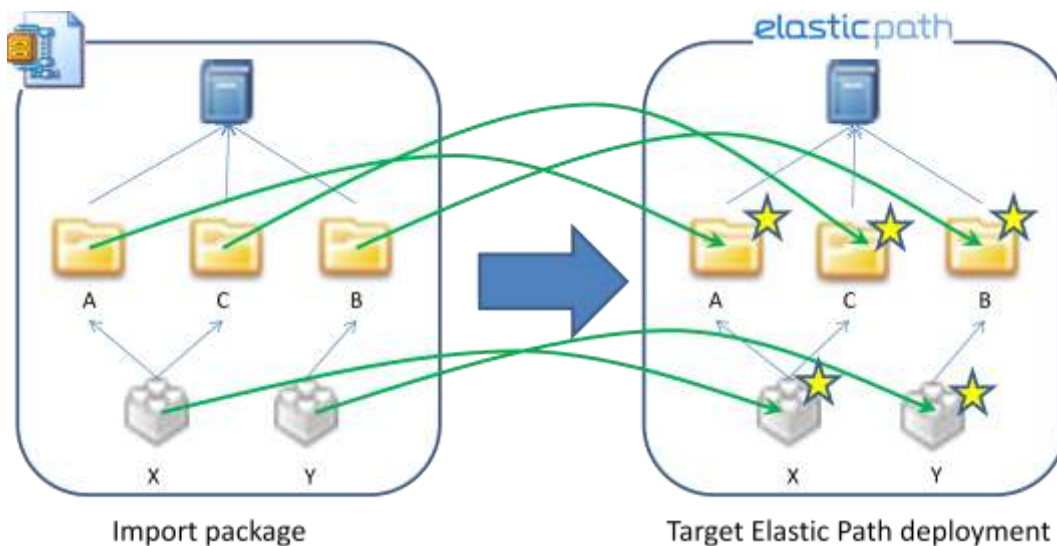
If the *Insert* import strategy is selected, category C and product Y are imported into Elastic Path. Category A, Category B, and Product X remain unchanged. (The stars indicate objects that have been updated after the import.)



If the *Update* import strategy is selected, categories A and B are imported into Elastic Path. The imported versions replace the existing categories A and B. Category C and product Y are not imported.



If the *Insert or Update* strategy is used, all the objects are imported. The existing objects are overwritten with the versions in the import package.



Combining Import Strategies

You can select different import strategies for products, categories, and catalogs. For example, you can select the *Insert or Update* strategy for categories and catalogs, and *Update* for products. This flexibility can be useful in some situations, but it can also cause problems.

For example, assume that catalogs are set to use *Update* and products are set to use *Insert or Update*. If the import package contains a new catalog and new products associated with that catalog, the following will occur when you import the package:

1. The new catalog will not be imported because the *Update* strategy is selected.
2. The products will fail to import because their catalog was not imported.

To prevent problems like this, do not use *Update* for catalog or category object types if any object types below them in the hierarchy are using *Update* or *Insert or Update*.

Dependency Conflicts

In addition to specifying conflict resolution, you can configure how to handle dependencies of those objects when there are conflicts. A dependency conflict occurs when an object exists in both the import package and the target system and each version has different dependencies.

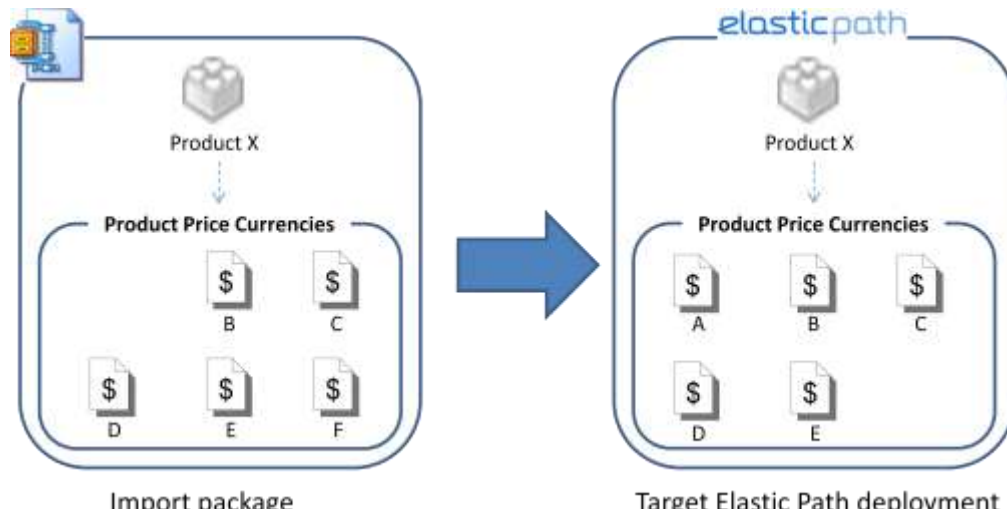
The dependencies affected by this are:

- Product SKUs
- Product catalog prices
- SKU catalog prices
- Category attributes
- Product attributes
- SKU attributes
- Product price currencies
- SKU price currencies
- Product price tiers
- SKU price tiers
- Product category assignments
- Product associations
- Product SKU pricing
- Setting values
- Setting metadata.

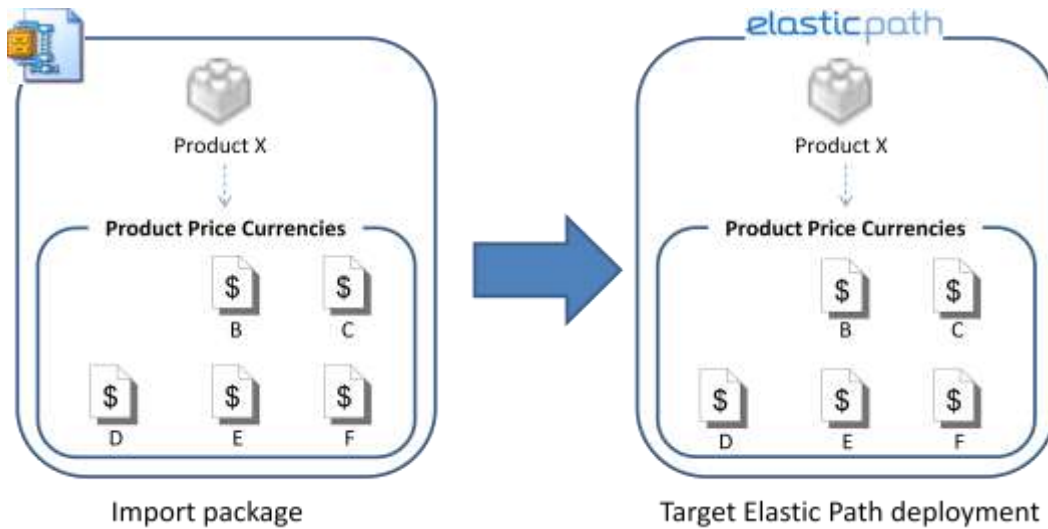
There are two options for handling dependency conflicts:

- *Clear collection.* This option removes all existing dependencies of the specified type from the target object and adds any new dependencies.
- *Retain collection.* This option keeps existing dependencies and adds any new dependencies to the target object. If the import package contains updates to existing dependencies the updates are applied.

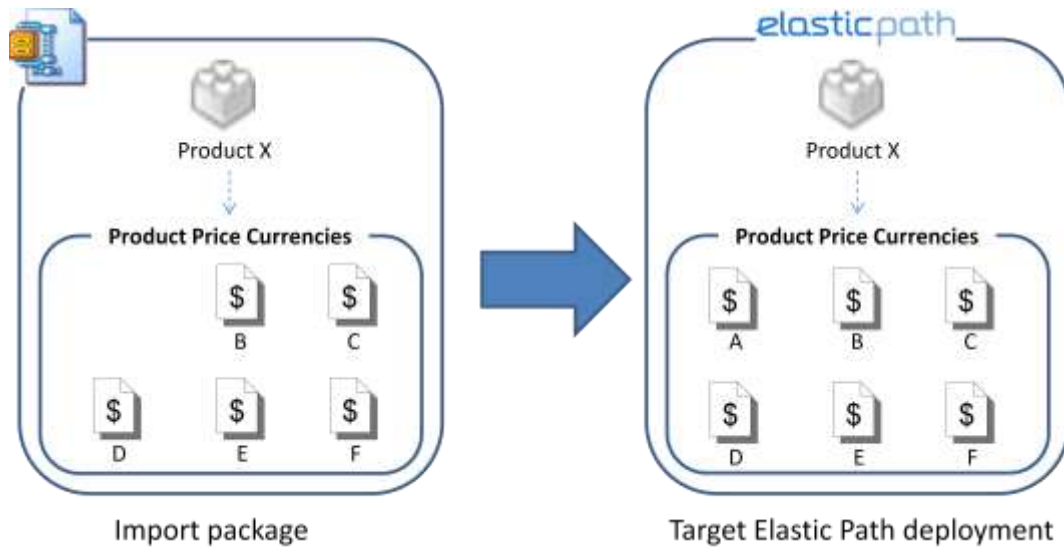
For example, an Elastic Path deployment contains product X, which has product price currencies A, B, C, D, and E. The package being imported contains product X as well, but has a slightly different set of product price currencies; it does not have A and it has a new product price currency, F.



If the *Clear collection* option is selected, the existing dependencies are removed. The net effect of this is that product price currency A is removed, F is added, and any changes made to the other product price currencies are applied.



If the *Retain collection* option is selected, the existing product price currencies remain the same and F is added. If the import package contains updates to existing product price currencies, the updates are applied.



Integrity Constraints and Import Order

When importing data into Elastic Path, it is critical to ensure that objects are imported in the correct order. For example, every category must be associated with a catalog. In the Elastic Path database, a constraint is defined to ensure that a category cannot be created without specifying a parent catalog that already exists in the database. Thus, if you attempt to import a category and its associated catalog does not exist, an error will occur and the category is not imported.

The order in which the objects in a package must be imported is automatically determined by the Import-Export tool at the time that the objects are exported. This information is stored in the manifest XML file inside the import package. The contents of the manifest XML file must not be modified manually under any circumstance.

Exporting Data

Note: Before exporting data from Elastic Path, make sure you have read the *Concepts* section, in particular, the sections relating to exportable object types.

The basic steps to export data from an Elastic Path deployment are:

1. Configure the environment settings.
2. Configure the export options.
3. Create a query to find the object or objects that you want to export.
4. Run the command line utility.

Configuring Environment Settings

Edit the `conf/misc/importexporttool.config` file and configure the environment settings for the system from which you are exporting data. You need to configure:

- The database connection details. For example, for a MySQL database named EP6 on the local machine running on port 3306, your settings should look similar to this:

```
db.connection.url=jdbc:mysql://localhost:3306/EP6?AutoReconnect=
true&useUnicode=true&characterEncoding=utf-8
db.connection.username=root
db.connection.password=root
db.connection.driver_class=com.mysql.jdbc.Driver
```

- The search server URL. Set the `search.host` property to the URL of the Elastic Path search server web application. For example:

```
search.host=http://localhost:8080/searchserver
```

- The VFS configuration for the Elastic Path deployment that contains the data you want to export. For example:

```
asset.vfs.protocol=ftp
asset.vfs.host=localhost
asset.vfs.port=21
asset.vfs.username=ftpuser
asset.vfs.password=password
asset.vfs.rootpath=c:/localassets/assets
image.asset.subfolder=images
digitalgoods.asset.subfolder=digitalassets
```

Note: These settings should match the corresponding settings in Elastic Path. To view the setting values, log in to Commerce Manager, choose *Activity->Configuration*, and click *System Configuration*.

Configuring Export Options

The Import-Export tool reads export options from an export configuration file. This XML file specifies:

- the type of data to export
- how the exported data will be packaged
- where the exported data will be stored on the file system.

There is a default export configuration file named `exportconfiguration.xml` located in the directory where you installed the Import-Export tool. Change the settings in this file to meet your export requirements. Alternatively, you can make a copy of this file, change the settings in the copy, and specify the filename of the copy when you run the batch script.

The following is an example of an export configuration file:

```
<?xml version="1.0"?>
<exportconfiguration>
  <exporter type="PRODUCT">
    <include type="PRODUCTASSOCIATION">
      <option key="DIRECT_ONLY" value="true"/>
    </include>
  </exporter>

  <packager type="ZIP" packagename="products.zip" />
  <delivery>
    <method>FILE</method>
    <!-- Specify target directory in the file system:
         "." - current directory, ".." - parent directory -->
    <target>./</target>
  </delivery>
</exportconfiguration>
```

The export configuration file contains three sections:

- exporter configuration
- packager configuration
- delivery configuration.

Exporter Configuration

The `exporter` element specifies the type of object you want to export. Supported values for the `type` attribute are:

- PRODUCT
- CATEGORY
- CATALOG
- PROMOTION
- SYSTEMCONFIGURATION

To include optional dependencies, add `include` elements to the `exporter` element. For example, the following configuration will export products, including their associated assets and product merchandising associations (direct only):

```
<exporter type="PRODUCT">
  <include type="ASSET"/>
  <include type="PRODUCTASSOCIATION">
    <option key="DIRECT_ONLY" value="true"/>
  </include>
</exporter>
```

The type of dependencies you can include depends on the type of object you are exporting. For the list of supported include types, see the description of the `include` element in *Appendix A: Export Configuration*.

Packager Configuration

The `packager` element specifies how the exported data is packaged. The `type` attribute can be set to:

- `ZIP`, which will create a zip archive of the exported data
- `NONE`, which will store the data in a directory.

The `packagename` attribute specifies the name of the file where the exported data will be written. This is only required if the `type` is set to `ZIP`.

Delivery Configuration

The `target` element under the `delivery` element specifies the directory where the exported data files will be created.

For complete descriptions of the elements in the export configuration file, see *Appendix A: Export Configuration*.

Creating the Search Configuration File

The search configuration file contains the criteria that determine which data is exported. There is a default search configuration file named `searchconfiguration.xml` located in the directory where you installed the Import-Export tool.

The file contains a query that defines the search criteria you want to use to fetch objects from the Elastic Path database. You can change the query to meet your export requirements. Alternatively, you can make a copy of this file, change the settings in the copy, and specify the filename of the copy when you run the batch script.

The following is an example of a search configuration file containing a query that matches all products associated with the specified catalog:

```
<?xml version="1.0"?>
<searchconfiguration>
  <epql>FIND Product WHERE CatalogCode='SNAPITUP'</epql>
</searchconfiguration>
```

For a complete description of the query language, see *Appendix D: Query Language*.

Tip: You can test product queries in the Commerce Manager's Advanced Search tool. This will allow you to quickly locate and fix problems and fine tune queries before performing the actual export. (Currently, only product queries are supported in the Advanced Search tool.)

Running the Batch Script to Export Data

Important: Before running the batch script, make sure the Elastic Path database server and web applications are running.

Open a DOS prompt and change to the directory where you extracted the Import-Export tool distributable. Launch the batch script:

```
importexport.bat -e <search_config_file> [-c <export_config_file>]
```

where:

- <search_config_file> is the path to the file that contains the search criteria.
- <export_config_file> is the path to the file that contains the export configuration. This parameter is optional. If it is not specified, the Import-Export tool will look for a file named `exportconfiguration.xml` in the Import-Export tool installation directory.

During the export operation, information and error messages are printed to the console. Errors are also logged in a file in the `logs` directory.

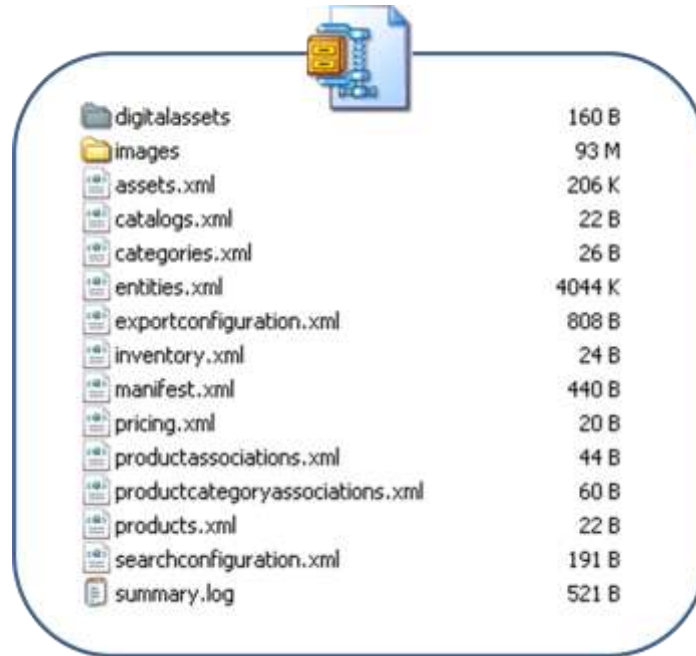
If you experience problems during export, see *Appendix E: Troubleshooting*.

Export Package Structure

When an export operation is successful, the exported data is written to the file system either in a directory or in a compressed archive. An export package contains the following:

- The export configuration file that was used to create the package
- The search configuration file containing the criteria that matches the objects that were exported
- A set of XML files, one file for each type of object that was exported
- Assets associated with the exported objects, if the option to include assets was specified
- A summary log, containing information about the results of the export operation
- The manifest file, which contains the list of the XML files in the export package.

Note: Data files in the package are imported in the order that they are listed in the manifest. Changing the order may cause integrity constraint violations, which could result in failure of the import operation. The manifest file should not be modified manually.



| | |
|---------------------------------|--------|
| digitalassets | 160 B |
| images | 93 M |
| assets.xml | 206 K |
| catalogs.xml | 22 B |
| categories.xml | 26 B |
| entities.xml | 4044 K |
| exportconfiguration.xml | 808 B |
| inventory.xml | 24 B |
| manifest.xml | 440 B |
| pricing.xml | 20 B |
| productassociations.xml | 44 B |
| productcategoryassociations.xml | 60 B |
| products.xml | 22 B |
| searchconfiguration.xml | 191 B |
| summary.log | 521 B |

Importing Data

Note: Before importing data into Elastic Path, make sure you have read the *Concepts* section, in particular, the sections relating to conflict resolution.

The basic steps to import data from an exported XML package are:

1. Configure the environment settings.
2. Configure the import options.
3. Run the command line utility.

To import data from an Elastic Path deployment, you must create an XML file containing the import configuration. Then, you launch the Import-Export batch script.

Configuring Environment Settings

Edit the `conf/misc/importexporttool.config` file and configure the environment settings for the system from which you are exporting data. For more information, see *Configuring Environment Settings* earlier in this document.

Configuring Import Options

The Import-Export tool reads import options from an import configuration file. This XML file specifies:

- the type(s) of data to import
- how to resolve conflicts that may occur during the import process
- where the data package to import is stored on the file system.

There is a default import configuration file named `importconfiguration.xml` located in the directory where you installed the Import-Export tool. You can change the settings in this file to meet your import requirements. Alternatively, you can make a copy of this file, change the settings in the copy, and specify the filename of the copy when you run the batch script.

The following is an example of an import configuration file:

```
<?xml version="1.0"?>
<importconfiguration>
  <xmlvalidation>true</xmlvalidation>
  <importstrategy>
    <importer type="PRODUCT">
      <importstrategy>INSERT_OR_UPDATE</importstrategy>
      <dependentelements>
        <dependentelement
type="PRODUCT_ASSOCIATIONS">CLEAR_COLLECTION</dependentelement>
```

```

    <dependentelement
type="PRODUCT_CATEGORY_ASSIGNMENTS">CLEAR_COLLECTION</dependentelement>
    <dependentelement
type="PRODUCT_ATTRIBUTES">CLEAR_COLLECTION</dependentelement>
    <dependentelement
type="PRODUCT_SKUS">RETAIN_COLLECTION</dependentelement>
  </dependentelements>
</importer>
</importstrategy>
<packager type="ZIP" />
<retrieval>
  <method>FILE</method>
  <source>c:/temp/importexport/export.zip</source>
</retrieval>
</importconfiguration>

```

The import configuration file contains four sections:

- XML validation configuration
- import strategy configuration
- packager configuration
- retrieval configuration.

XML Validation Configuration

The `xmlvalidation` element specifies whether to validate the individual XML files you are importing before attempting to import the package. The value must be either `true` or `false`.

Import Strategy Configuration

The outermost `importstrategy` element contains one or more `importer` elements.

Each `importer` element contains information about what data to import. The `type` attribute specifies the type of object to import. Supported values are `PRODUCT`, `CATEGORY`, `CATALOG`, `PROMOTION`, and `SYSTEMCONFIGURATION`. The `importstrategy` element within `importer` specifies how conflicts are handled during the import operation. The following values are supported:

| Import strategy | Description |
|------------------|---|
| INSERT | Only new objects are imported. If an object with the same identifier already exists in the target system, the object is not imported. |
| UPDATE | Only updated objects are imported. If an object being imported does not exist in the target system, the object is not imported. |
| INSERT_OR_UPDATE | Both new and updated objects are imported. If no import strategy is specified, the default is <code>INSERT_OR_UPDATE</code> . |

The optional `dependentelements` element lets you configure conflict resolution for dependencies. It can contain one or more `dependentelement` elements. The `type` attribute specifies the type of dependency. Supported values are:

- `PRODUCT_CATEGORY_ASSIGNMENTS`
- `PRODUCT_ATTRIBUTES`

- PRODUCT_SKUS
- SKU_ATTRIBUTES
- PRODUCT_SKU_PRICING
- PRODUCT_CATALOG_PRICES
- PRODUCT_PRICE_CURRENCIES
- PRODUCT_PRICE_TIERS
- SKU_CATALOG_PRICES
- SKU_PRICE_CURRENCIES
- SKU_PRICE_TIERS
- SETTING_VALUES
- SETTING_METADATA

Each dependent element contains either `CLEAR_COLLECTION` or `RETAIN_COLLECTION`, to indicate how dependencies of the specified type are handled in the event of a conflict during import.

Packager Configuration

The `packager` element specifies how the data being imported is packaged. The `type` attribute must be set to either `ZIP` or `NONE`. If the type is `ZIP`, the retrieval method must specify a zip file. If the type is `NONE`, the retrieval method must specify a directory.

Retrieval Configuration

The `retrieval` element specifies how to retrieve the import data. The `method` element must always be set to `FILE`. The `source` element must contain the path to the package.

Note: Use the forward slash as the path separator inside the `source` element.

For a complete description of the elements in the import configuration file, see *Appendix B: Import Configuration*.

Running the Batch Script to Import Data

Important: Before running the batch script, make sure the Elastic Path database server and web applications are running.

Open a DOS prompt and change to the directory where you extracted the Import-Export tool distributable. Launch the batch script:

```
importexport.bat -i -c <import_config_file>
```

where `<import_config_file>` is the path to the file that contains the import configuration.

During the import operation, information and error messages are printed to the console. Errors are also logged in a file in the `logs` directory.

If you experience problems during import, see *Appendix E: Troubleshooting*.

Appendix A: Export Configuration

This appendix describes the elements in the export configuration schema. The export configuration schema (`schemaExportConfig.xsd`) is located in the `schema` subdirectory of the Import-Export installation directory and can be used to validate your export configuration files.

delivery

Specifies where the exported data package is stored.

Parent elements

`exportconfiguration`

Attributes

None.

Child elements

Note: Child elements are ordered.

| Name | Number of occurrences | Description |
|---------------------|-----------------------|--|
| <code>method</code> | 1 | Specifies the data export method. The only supported value is <code>FILE</code> . |
| <code>target</code> | 1 | Specifies the directory where exported files will be located. Can be an absolute path or relative to the Import-Export tool directory. If the directory does not exist, it is created automatically. |

exportconfiguration

The root element in the export configuration XML file.

Parent elements

None.

Attributes

None.

Child elements

Note: Child elements are ordered.

| Name | Number of occurrences | Description |
|--------------------------------|-----------------------|--|
| <code>exporter</code> | 1 | Specifies what type of data to export. |
| <code>packager</code> | 1 | Specifies how the exported data is packaged. |
| <code>delivery</code> | 1 | Specifies where the exported data package is located. |
| <code>transformerschain</code> | 0..1 | Specifies a set of transformations to apply to the data before it is packaged. |

exporter

Specifies what type of data to export.

Parent elements

`exportconfiguration`

Attributes

| Name | Number of occurrences | Description |
|------|-----------------------|---|
| type | 1 | <p>Specifies the type of data to export. Supported values are:</p> <p>PRODUCT Export products that match the search criteria. The exported product data is stored in a file named <code>products.xml</code>. The associated category and catalog data is exported as well.</p> <p>CATALOG Export catalogs that match the search criteria. The exported catalog data is stored in a file named <code>catalogs.xml</code>.</p> <p>CATEGORY Export categories that match the search criteria. The exported category data is stored in a file named <code>categories.xml</code>. The associated catalog data is exported as well.</p> <p>PROMOTION Export promotions that match the search criteria. The exported promotion data is stored in a file named <code>promotions.xml</code>. If the associated rule data is exported, it is stored in a separate file named <code>condition_rules.xml</code>.</p> <p>SYSTEMCONFIGURATION Export system configuration settings that match the search criteria. The setting data is stored in a file named <code>system_configuration.xml</code>.</p> |

Child elements

| Name | Number of occurrences | Description |
|---------|-----------------------|--|
| include | 0..* | Specifies one or more optional data subtypes to include in the export. |

include

Specifies an optional data subtype to include in the export.

Parent elements

exporter

Attributes

| Name | Number of occurrences | Description |
|------|-----------------------|--|
| type | 1 | <p>Specifies the optional object type. Supported values are:</p> <p>ASSETS Assets, such as images, PDFs, MP3s, and other files. The exported asset data is stored in a file named <code>assets.xml</code>.</p> <p>INVENTORY Inventory information by SKU and warehouse. The exported inventory data is stored in a file named <code>inventory.xml</code>. This type is only supported when the <code>exporter type</code> is set to <code>PRODUCT</code>.</p> <p>PRICING Price information by product and catalog. The exported pricing data is stored in a file named <code>pricing.xml</code>. This type is only supported when the <code>exporter type</code> is set to <code>PRODUCT</code>.</p> <p>PRODUCTASSOCIATION Merchandising associations, such as cross-sells, up-sells, warranties, and recommendations. The exported product association data is stored in a file named <code>productassociations.xml</code>. This type is only supported when the <code>exporter type</code> is set to <code>PRODUCT</code>.</p> <p>CONDITIONRULE Eligibilities and conditions for promotions. The exported rule data is stored in a file named <code>condition_rules.xml</code>. This type is only supported when the <code>exporter type</code> is set to <code>PROMOTION</code>.</p> |

Child elements

| Name | Number of occurrences | Description |
|--------|-----------------------|--|
| option | 0..* | Sets an export option specific to the object type. |

option

The `option` element specifies which optional object types to include in the export.

Parent elements

`include`

Attributes

| Name | Number of occurrences | Description |
|--------------------|-----------------------|---|
| <code>key</code> | 1 | The unique identifier of the option. |
| <code>value</code> | 1 | The value associated with the option for this object. |

packager

Specifies where the exported data package is stored.

Parent elements

`exportconfiguration`

Attributes

| Name | Number of occurrences | Description |
|--------------------------|-----------------------|--|
| <code>type</code> | 1 | Specifies how the exported files are packaged. Supported values are: <code>ZIP</code> The exported files are packaged in a zip archive. <code>NONE</code> The exported files are not packaged. |
| <code>packagename</code> | 0..1 | Specifies the name of the destination zip file. Not required if <code>type</code> is set to <code>NONE</code> . |

Child elements

None.

transformer

Specifies a transformation to apply to the data before it is packaged.

Parent elements

`transformerschain`

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|-------------------|-----------------------|---|
| <code>type</code> | 0..1 | The qualified class name of the transformer. The specified class must be on the Import-Export tool's classpath. |

transformerschain

The `transformerschain` element specifies a set of transformations to apply to the data before it is packaged.

Parent elements

`exportconfiguration`

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|--------------------------|-----------------------|--|
| <code>transformer</code> | 0..* | Specifies what type of data to export. |

type

The fully qualified class name of the transformer. The specified class must be on the Import-Export tool's classpath.

Parent elements

`transformer`

Attributes

None.

Child elements

None.

Appendix B: Import Configuration

This appendix describes the elements in the import configuration schema. The import configuration schema (`schemaImportConfig.xsd`) is located in the `schema` subdirectory of the Import-Export installation directory and can be used to validate your import configuration files.

dependentelement

Determines how dependencies of the specified item type are handled. For example, a product has a set of associated attributes. During an import operation, the product being imported may have a different set of associated attributes than the product in the destination system. This setting specifies whether the existing attributes on the product are removed and replaced with the imported attributes, or if the new attributes are added to the existing set of attributes.

Note: If the import configuration does not contain a `dependentelement` for a particular item type, then the default behavior for dependencies of that item type is `CLEAR_COLLECTION`.

Supported values are:

| | |
|--------------------------------|--|
| <code>CLEAR_COLLECTION</code> | Remove all existing dependencies of this type before adding new dependencies from the import package. |
| <code>RETAIN_COLLECTION</code> | Keep existing dependencies of this type and add only new dependencies from the import package, if any. |

Parent elements

`dependentelements`

Attributes

| Name | Number of occurrences | Description |
|------|-----------------------|--|
| type | 1 | <p>Specifies the type of dependency.</p> <p>For categories, possible values are:</p> <ul style="list-style-type: none">• CATEGORY_ATTRIBUTES <p>For products, possible values are:</p> <ul style="list-style-type: none">• PRODUCT_CATALOG_PRICES• PRODUCT_PRICE_TIERS• PRODUCT_PRICE_CURRENCIES• PRODUCT_ATTRIBUTES• PRODUCT_SKUS• PRODUCT_CATEGORY_ASSIGNMENTS• PRODUCT_ASSOCIATIONS• PRODUCT_SKU_PRICING• SKU_ATTRIBUTES• SKU_PRICE_TIERS• SKU_PRICE_CURRENCIES• SKU_CATALOG_PRICES <p>For system configuration settings, possible values are:</p> <ul style="list-style-type: none">• SETTING_VALUES• SETTING_METADATA |

Child elements

None.

dependentelements

Specifies how the dependencies of each item type are handled.

Parent elements

importer

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|------------------|-----------------------|---|
| dependentelement | 0..* | Specifies how the dependencies of an item type are handled. |

importconfiguration

The `importconfiguration` element is the root element in the import configuration XML file.

Parent elements

None.

Attributes

None.

Child elements

Note: Child elements are ordered.

| Name | Number of occurrences | Description |
|-------------------|-----------------------|---|
| xmlvalidation | 1 | Specifies whether to validate all the XML files in the import package before importing. |
| importstrategy | 0..1 | Specifies how the Import-Export handles conflicts between the data it is importing and the existing data. |
| packager | 1 | Specifies how the data being imported is packaged. |
| retrieval | 1 | Specifies the location of the data being imported. |
| transformerschain | 0..1 | Specifies a set of transformations to apply to the data before it is imported. |

importer

Specifies the type of object to import. Dependencies of those objects are also imported (see below).

Parent elements

`importstrategy`

Attributes

| Name | Number of occurrences | Description | | | | | | | | | | | | |
|---------------------|--|--|-------|-------------|---------|---|---------|--|----------|---|------------|--|---------------------|---|
| type | 1 | <p>Specifies the type of objects to be imported. Supported values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>PRODUCT</td> <td>Import products and their associated categories and catalogs. The product data must be stored in the import package in a file named <code>products.xml</code>.</td> </tr> <tr> <td>CATALOG</td> <td>Import catalogs. The catalog data must be stored in the import package in a file named <code>catalogs.xml</code>.</td> </tr> <tr> <td>CATEGORY</td> <td>Import categories and their associated catalogs. The category data must be stored in the import package in a file named <code>categories.xml</code>.</td> </tr> <tr> <td>PROMOTIONS</td> <td>Import promotions and (optionally) their associated conditions and eligibilities. The promotion data must be stored in the import package in a file named <code>promotions.xml</code>. Condition and eligibility data, if included, must be stored in the import package in a file named <code>condition_rules.xml</code>.</td> </tr> <tr> <td>SYSTEMCONFIGURATION</td> <td>Import system configuration settings. The setting data must be stored in the import package in a file named <code>system_configuration.xml</code>.</td> </tr> </tbody> </table> | Value | Description | PRODUCT | Import products and their associated categories and catalogs. The product data must be stored in the import package in a file named <code>products.xml</code> . | CATALOG | Import catalogs. The catalog data must be stored in the import package in a file named <code>catalogs.xml</code> . | CATEGORY | Import categories and their associated catalogs. The category data must be stored in the import package in a file named <code>categories.xml</code> . | PROMOTIONS | Import promotions and (optionally) their associated conditions and eligibilities. The promotion data must be stored in the import package in a file named <code>promotions.xml</code> . Condition and eligibility data, if included, must be stored in the import package in a file named <code>condition_rules.xml</code> . | SYSTEMCONFIGURATION | Import system configuration settings. The setting data must be stored in the import package in a file named <code>system_configuration.xml</code> . |
| Value | Description | | | | | | | | | | | | | |
| PRODUCT | Import products and their associated categories and catalogs. The product data must be stored in the import package in a file named <code>products.xml</code> . | | | | | | | | | | | | | |
| CATALOG | Import catalogs. The catalog data must be stored in the import package in a file named <code>catalogs.xml</code> . | | | | | | | | | | | | | |
| CATEGORY | Import categories and their associated catalogs. The category data must be stored in the import package in a file named <code>categories.xml</code> . | | | | | | | | | | | | | |
| PROMOTIONS | Import promotions and (optionally) their associated conditions and eligibilities. The promotion data must be stored in the import package in a file named <code>promotions.xml</code> . Condition and eligibility data, if included, must be stored in the import package in a file named <code>condition_rules.xml</code> . | | | | | | | | | | | | | |
| SYSTEMCONFIGURATION | Import system configuration settings. The setting data must be stored in the import package in a file named <code>system_configuration.xml</code> . | | | | | | | | | | | | | |

Child elements

Note: Child elements are ordered.

| Name | Number of occurrences | Description |
|-------------------|-----------------------|--|
| importstrategy | 0..1 | Specifies how the Import-Export tool resolves conflicts between data in the import package and existing data in the destination. |
| dependentelements | 0..1 | Specifies how dependencies of each item type are handled. |

importstrategy (child of importconfiguration)

Specifies the data to import.

Parent elements

importconfiguration

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|----------|-----------------------|---------------------------------------|
| importer | 0..* | Specifies the type of data to import. |

importstrategy (child of importer)

Specifies how the Import-Export tool resolves conflicts during an import operation. Conflicts occur if the code of the item in the import package matches the code of an item in the destination system.

Possible values are:

| Value | Description |
|------------------|---|
| INSERT | Only import items that do not exist in the system. |
| INSERT_OR_UPDATE | Import items that do not exist in the system and update items that already exist. |
| UPDATE | Only update items that already exist in the system. |

Parent elements

importer

Attributes

None.

Child elements

None.

method

Specifies the data import method. The only supported value is `FILE`.

Parent elements

retrieval

Attributes

None.

Child elements

None.

packager

Specifies how the data being imported is packaged.

Parent elements

importconfiguration

Attributes

| Name | Number of occurrences | Description | | | | | | |
|-------------|---|--|-------|-------------|-----|---|------|---|
| type | 0..1 | Supported values are: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>ZIP</td><td>The data being imported is packaged in a zip archive.</td></tr><tr><td>NONE</td><td>The data being imported is not packaged. The individual data files are stored in uncompressed format on the</td></tr></tbody></table> | Value | Description | ZIP | The data being imported is packaged in a zip archive. | NONE | The data being imported is not packaged. The individual data files are stored in uncompressed format on the |
| Value | Description | | | | | | | |
| ZIP | The data being imported is packaged in a zip archive. | | | | | | | |
| NONE | The data being imported is not packaged. The individual data files are stored in uncompressed format on the | | | | | | | |
| packagename | 0..1 | Specifies the name of the zip file to import. Not required if <code>type</code> is set to <code>NONE</code> . | | | | | | |

Child elements

None.

retrieval

Specifies the location of the data being imported.

Parent elements

`importconfiguration`

Attributes

None.

Child elements

Note: Child elements are ordered.

| Name | Number of occurrences | Description |
|---------------------|-----------------------|--|
| <code>method</code> | 0..1 | Specifies the data import method. The only supported value is <code>FILE</code> . |
| <code>source</code> | 0..1 | Specifies the directory that contains the files to be imported. Can be an absolute path or relative to the Import-Export tool directory. |

source

Specifies the directory that contains the files to be imported. Can be an absolute path or relative to the Import-Export tool directory.

Parent elements

`retrieval`

Attributes

None.

Child elements

None.

transformer

Specifies a transformation to apply to the data before it is imported.

Parent elements

`transformerschain`

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|------|-----------------------|---|
| type | 0..1 | The qualified class name of the transformer. The specified class must be on the Import-Export tool's classpath. |

transformerschain

Specifies a set of transformations to apply to the data before it is imported.

Parent elements

`importconfiguration`

Attributes

None.

Child elements

| Name | Number of occurrences | Description |
|-------------|-----------------------|--|
| transformer | 0..1 | Specifies a transformation to apply to the data before it is imported. |

type

The fully qualified class name of the transformer. The specified class must be on the Import-Export tool's classpath.

Parent elements

`transformer`

Attributes

None.

Child elements

None.

xmlvalidation

Specifies whether to validate all the XML files in the import package before importing. If set to true, the import operation will not proceed if any of the files are invalid. If set to false, all valid files are imported and errors are logged for all invalid files.

Disabling validation may improve performance when importing large data files, but should not be disabled if the data files have been modified manually.

Possible values are `true` (1) or `false` (0).

Parent elements

`importconfiguration`

Attributes

None.

Child elements

None.

Appendix C: Search Configuration

This appendix describes the elements in the search configuration schema. The search configuration schema (`schemaSearchConfig.xsd`) is located in the `schema` subdirectory of the Import-Export installation directory and can be used to validate your search configuration files.

epql

Contains the query that will be used to select the products and categories to export. For more information on the query language, see *Appendix D: Query Language*.

Parent elements

`searchconfiguration`

Attributes

None.

Child elements

None.

searchconfiguration

The root element of the search configuration file.

Parent elements

None.

Attributes

None.

Child elements

Note: Child elements are not ordered.

| Name | Number of occurrences | Description |
|------|-----------------------|---|
| epq1 | 1 | Contains the query that will be used to select the products and categories to export. |

Appendix D: Query Language

The Elastic Path Import-Export tool relies on a special query language to locate the products, catalogs, and categories to be exported. This appendix describes the query language, including supported search fields and operators.

Query Format

A simple query has the following form:

```
FIND <objectType> WHERE <expression>
```

where

- *<objectType>* is the type of object you want to retrieve. Supported values are:
 - Product
 - Category
 - Catalog
 - Promotion
 - Configuration

Note: The object type must match the exporter type specified in the export configuration file.

- *<expression>* specifies the criteria that determines which objects to retrieve. An expression has the following form:

```
<field> <operator> <value>
```

where

- *<field>* is the field that contains the values you want to compare. A field represents a common characteristic of the object. For example, if you want to look for products of a specific brand, you would include the `BrandCode` field in your query. The supported fields are described in *Supported Fields* further in this appendix.
- *<operator>* is the operator you are using to perform the comparison.
- *<value>* is the literal value you want to compare to the field values.

Note: The `WHERE` clause is optional. If omitted, all products are matched.

For example, the following query matches the product whose code is 10030205:

```
FIND Product WHERE ProductCode = '10030205'
```

In addition to searching for field values, you can also search for attribute values on products and categories. To search for a value in an attribute, the expression has the following form:

```
AttributeName{<attribute_name>} <operator> <value>
```

where *<attribute_name>* is the name of a product attribute.

For example, the following query matches all products that have the Header / Model attribute set to MX:

```
FIND Product WHERE AttributeName{Header / Model} = 'MX'
```

To search for a value in a product SKU attribute, use `SkuAttributeName`. For example, the following query matches all products that have the Header / Model SKU attribute set to MX:

```
FIND Product WHERE SkuAttributeName{Header / Model} = 'MX'
```

Case Sensitivity

The following are case-sensitive:

- Keywords, such as `FIND`, `WHERE`, `AND`, `START`, and `AttributeName`
- Field names
- Attribute names
- Currency codes (must be in upper case letters)
- Store codes (when querying on the `Price` field)

The following are not case-sensitive:

- Literal string values (on the right side of the comparison)
- Language codes (when querying on localized fields)

Data Types and Supported Operators

Every field and attribute has a data type. The data type determines what kind of data the field or attribute can contain. The following table shows the data types and the operators supported for each one:

| Data type | Supported operators | Notes | Examples |
|-----------|---------------------|---|---|
| DateTime | =, !=, <, =<, >, >= | The DateTime format is YYYY-MM-DDThh:mm:ss. | '2008-08-30T17:19:00' '2009-12-21T07:06:00' |
| Double | =, !=, <, =<, >, >= | The decimal point is optional. | 100.00 20199 |
| Integer | =, !=, <, =<, >, >= | | 1 |
| String | =, != | Strings must be surrounded by single quotes ('). If a string contains a single quote, it must be preceded by a backslash. | 'StringValue' 'Canon - Kit d\'accessoires pour appareil photo' |

Supported Fields

Product Fields

The following fields can be used in product queries:

| Field name | Description | Data type | Localized |
|------------------|----------------------------|-----------|-----------|
| BrandCode | Brand code | String | No |
| BrandName | Brand name | String | Yes |
| CatalogCode | Catalog code | String | No |
| CategoryCode | Category code | String | No |
| CategoryName | Category name | String | Yes |
| CreateDate | Product created date | Datetime | No |
| LastModifiedDate | Product last modified date | Datetime | No |
| Price | Product price | Double | No |
| ProductActive | Product active | String | No |
| ProductCode | Product code | String | No |
| ProductName | Product name | String | Yes |
| SKUCode | Product SKU code | String | No |
| StoreCode | Store code | String | No |

Category Fields

The following fields can be used in category queries:

| Field name | Description | Data type | Localized |
|--------------|---------------|-----------|-----------|
| CatalogCode | Catalog code | String | No |
| CategoryCode | Category code | String | No |
| CategoryName | Category name | String | Yes |

Catalog Fields

The CatalogCode field is the only field supported in catalog queries.

Promotion Fields

The following fields can be used in promotion queries:

| Field name | Description | Data type | Localized |
|---------------|--|-----------|-----------|
| StoreCode | Store code | String | No |
| PromotionName | Promotion name | String | No |
| PromotionType | Promotion type (ShoppingCart or Catalog) | String | No |
| State | The state of the promotion (ACTIVE, EXPIRED, or DISABLED, no quotes) | Constant | No |

System Configuration Setting Fields

The following fields can be used in setting queries:

| Field name | Description | Data type | Localized |
|------------|---------------|-----------|-----------|
| Namespace | Setting path | String | No |
| Context | Context value | String | No |

Localized Fields and Attributes

Some fields contain localized values. To include a localized field in your query, you must specify the language that you want to search. For example, the following query matches the product whose French name is Canon - Kit d'accessoires pour appareil photo:

```
FIND Product WHERE ProductName[fr] = 'Canon - Kit d\'accessoires pour  
appareil photo'
```

The value between the square brackets indicates the language. You can use either the two-letter language code or the full language name.

Note: The apostrophe in d'accessoires must be preceded by a backslash. Whenever a string contains the apostrophe character, it must be escaped with a backslash. Otherwise it will be interpreted as the end of the string and cause a parsing error when the query is validated.

Attributes may also contain localized values. For example, the following query matches all products that have the English value of the Lens System / Type attribute set to Zoom lens:

```
FIND Product WHERE AttributeName{Lens System / Type}[en] = 'Zoom lens'
```

The Price Field

To query on the `Price` field, you need to specify a store code and a currency code. The format is as follows:

```
Price{<storeCode>} [<currencyCode>]
```

where

- `<storeCode>` is the store code
- `<currencyCode>` is the currency code.

For example, the following query matches all products in the Snap It Up store that cost less than 100.00 USD:

```
FIND Product WHERE Price{SNAPITUP}[USD] < 100.00
```

System Configuration Setting Metadata

To query for settings with specific metadata values, you use the `MetadataKey` keyword followed by curly braces. Put the metadata key within the curly braces.

When querying for metadata, you can use the `*` wildcard to look for all settings that have a particular metadata key, regardless of the value assigned to it. For example, the following query matches all settings that have a value associated with the `sfRefreshStrategy` metadata key.

```
FIND Configuration WHERE MetadataKey{sfRefreshStrategy} = *
```

The following query produces the same results.

```
FIND Configuration WHERE MetadataKey{sfRefreshStrategy} != ''
```

The Setting Namespace Field

System configuration settings in Elastic Path are identified by a path-like structure. For example, the store theme setting is identified by `COMMERCE/STORE/theme`. The path-like structure is intended to allow hierarchical setting definitions. Note that this is a purely logical organization and does not represent how settings are physically stored; settings are stored in a single, non-hierarchical relation in the database.

To query for a setting by its path, use the `Namespace` field. The following query matches the `COMMERCE/STORE/theme` setting:

```
FIND Configuration WHERE Namespace = 'COMMERCE/STORE/theme'
```

You can also specify partial paths. To match all settings that begin with a particular path, you must specify the path up to and including the slash character (`/`).

For example, the following query matches all settings whose path begins with `COMMERCE/STORE`:

```
FIND Configuration WHERE Namespace = 'COMMERCE/STORE/'
```

In this case, there are 17 settings that match this partial path.

For a more refined search, you can use the `%` character in the last part of the path. The following query matches all settings immediately under `COMMERCE/STORE` that begin with the letter "s".

```
FIND Configuration WHERE Namespace = 'COMMERCE/STORE/s%'
```

This matches the two immediate children of `COMMERCE/STORE` that begin with "s":
`COMMERCE/STORE/storefrontUrl` and `COMMERCE/STORE/seoEnabled`.

Combining Expressions

You can use `AND` or `OR` to combine multiple expressions.

For example, the following query uses `AND` to match all zoom lens items (based on the value of the `Lens System / Type` attribute) that cost more than 100.00 USD in the Snap It Up store:

```
FIND Product WHERE AttributeName{Lens System / Type}[en] = 'Zoom lens' AND Price{SNAPITUP}[USD] > 100.00
```

The following query uses `OR` to match all Pentax and Kodak products:

```
FIND Product WHERE BrandName[en] = 'Pentax' OR BrandName[en] = 'Kodak'
```

You can use parentheses (the `(` and `)` characters) to set the order in which expressions are evaluated. Expressions in parentheses are evaluated first. You can nest expression groups.

For example, the following query matches all products that are either Pentax or Kodak brand and that are under 100.00 USD:

```
FIND Product WHERE Price{SNAPITUP}[USD] < 100 AND (BrandName[en] = 'Pentax' OR BrandName[en] = 'Kodak')
```

The following query matches all products that are either Pentax brand products under 100 USD or Kodak brand products:

```
FIND Product WHERE (Price{SNAPITUP}[USD] < 100 AND BrandName[en] = 'Pentax') OR BrandName[en] = 'Kodak'
```

Limiting the Result Set Size

You can limit the number of results returned by adding `LIMIT <number>`, where *<number>* specifies the maximum number of items to include in the results. For example, to return the first ten Pentax products, execute the following:

```
FIND Product WHERE BrandName[en] = 'Pentax' LIMIT 10
```

Specifying the First Match

You can specify the position of the first match to return within the results by adding `START <number>`, where *<number>* is the position of the first match you want to return. For example, the following query returns the first ten matches starting at the twentieth match:

```
FIND Product WHERE BrandName[en] = 'Pentax' LIMIT 10 START 20
```

Note: Currently, it is not possible to sort results. This is primarily used in search queries executed by the Import-Export tool to split result sets into more manageable "chunks". For example, the following query returns 615 matches for the Snap It Up demo store:

```
FIND Product WHERE Price{SNAPITUP}[USD] < 50.00
```

You can split those matches into three separate result sets by executing the following three queries:

```
FIND Product WHERE Price{SNAPITUP}[USD] < 50.00 LIMIT 200  
FIND Product WHERE Price{SNAPITUP}[USD] < 50.00 START 201 LIMIT 200  
FIND Product WHERE Price{SNAPITUP}[USD] < 50.00 START 401
```

The first query returns matches 1 to 200. The second returns 201 to 400. The third returns from 401 to the last match (615).

Appendix E: Troubleshooting

| Problem | Solution |
|---|---|
| The export or import operation fails and the error log contains this message: <code>java.sql.SQLException: Access denied for user 'root'@'localhost'</code> | Make sure the database is running and that the database connection properties in <code>conf/misc/importexporttool.config</code> are set correctly. |
| The export or import operation fails and the error log contains a message similar to the following: <code>java.lang.ClassNotFoundException: com.mysql.jdbc.Driver</code> | Make sure the appropriate JDBC library is on the Import-Export tool's class path. Copy the JDBC library used by the Elastic Path web applications to the Import-Export tool's <code>lib</code> directory. |
| The export or import operation fails and the error log contains <code>ConfigurationException</code> and/or <code>SAXParseException</code> messages. | Make sure that the export or import configuration file contains well formed XML and that it validates against the corresponding schema in the <code>schema</code> directory (<code>schemaExportConfig.xsd</code> or <code>schemaImportConfig.xsd</code>). |
| The export or import operation fails and the error log contains this message: <code>java.net.ConnectException: Connection refused: connect</code> | Make sure the database and the Elastic Path web applications are running and that the search server URL is correct. |
| The export operation completes successfully, but the wrong data or no data is exported. | Make sure that the search query in the search configuration file is correct. |
| The export operation fails and the error log contains the following: <code>EpQLParseException: Cannot parse</code> | There is a syntax error in the search query string. Check the syntax of your query against the syntax described in <i>Appendix D: Query Language</i> . |
| The export operation completes successfully, but no files are created or the files are in the wrong location. | Verify the package name and delivery target settings in the export configuration file. |
| The import operation fails because it can't find the data files to be imported. | Verify the retrieval source setting in the import configuration file. |
| The export operation completes successfully, but the summary report indicates that the total number of objects is 0 and the generated files contain no data. | Make sure that the type of the exporter in the export configuration file matches the export criteria in the search configuration file. If the types do not match, no data will be exported. |
| A product or category fails to import because the parent catalog does not exist. However, the parent catalog exists in the package. | Make sure that the import strategy for the parent object type is set to use <i>Insert</i> or <i>Insert or Update</i> . For more information, see the section <i>Combining Import Strategies</i> . |

| Problem | Solution |
|---|---|
| <p>The export operation fails and the log contains a message indicates that the currency code could not be found.</p> | <p>This is may be caused by conflicts between the JVM used by Import-Export and the JVM that was used by the Commerce Manager client when the data was originally created in Elastic Path. Newer versions of the JVM include currency code updates for several countries and the older currency codes are no longer recognized.</p> <p>To prevent the problem, make sure the Commerce Manager client and Import-Export tool are both using the most recent JVM supported by Elastic Path.</p> <p>For more information, see the following: http://bugs.sun.com/view_bug.do?bug_id=6570259</p> |