Magento®

Magento 2 Migration Overview

Technical overview of data migration from Magento 1.x to Magento 2.0

Table of Contents

Introduction

We're pleased you're considering moving from the world's #1 eCommerce platform—Magento 1.x—to the eCommerce platform for the future, Magento 2. We're also excited to share the details about this process, which we refer to as *migration*.

Magento 2 migration involves four components: data, extensions, themes, and customizations.

Data

We've developed the Magento 2 Data Migration Tool to help you efficiently port all of your key product, customer, and order data, store configurations, promotions and more to Magento 2. This paper provides information on the tool and best practices for using it to migrate your data.

Extensions

We are actively working with the Magento development community to help ensure the most widely used extensions will be updated, vetted, and available on Magento Connect when Magento 2 becomes generally available. More information on developing extensions for Magento 2 is available in the *Magento 2 Extension Developer Guide*.

Themes and Customizations

Magento 2 uses several new approaches and technologies that give merchants an unmatched ability to create innovative shopping experiences and scale to new levels. To take advantage of these advances, developers will need to make changes to their themes and customizations. Documentation is available online for creating Magento 2 *themes*, *layouts*, and customizations.

Just like an upgrade between 1.x versions (for example, from Enterprise Edition 1.12 to Enterprise Edition 1.14) , the level of effort to migrate from Magento 1.x to Magento 2.0 depends upon how you have built your site and its level of customization.

Initial estimates indicate that an average Magento 2 migration is only about 20% larger than a Magento 1.x upgrade. Over the coming months, as we proceed with testing and the merchant beta program, we will be able to refine this number so you can plan your budgets and timelines.

High-Level Data Migration Roadmap

As mentioned previously, the primary focus of this paper is on the data migration process between Magento 1.x and Magento 2.0. To assist you with your migration, Magento is creating the Magento 2 Migration Tool, a data migration command-line interface (CLI) tool with auxiliary verification, progress tracking, logging, and testing functions that allows direct data copying from a Magento 1.x database to a Magento 2 database. The tool is expected to be available by Q3 2015 and it efficiently and reliably migrates all core Magento data entities and any custom data entities that do not change between versions. The Migration Tool will support the following Magento versions:

- Community Edition (CE) 1.6, 1.7, 1.8, and 1.9
- Enterprise Edition (EE) 1.11, 1.12, 1.13, and 1.14

Architectural Overview

After considering several alternatives, we decided the most efficient way to migrate is using extract, transform, and load (ETL) because it migrates almost everything as quickly as possible while incurring the least amount of downtime of all options we considered. To help migrate customized data, we use declarative language in our database mapping files. We also use migration *steps* to make it easier to write complex code for your custom needs. Mapping files and migration steps are discussed **here**.

Best Practices and Benchmarking

This section provides our best information about how to speed up and simplify your migration, and provides guidance about how much time you can expect migration to require.

Best Practices and Recommendations

- **Migrate data** from a replicated Magento 1.x database instance.
- **Remove outdated and redundant data** from your Magento 1.x database (for example, you could remove logs, order quotes, recently viewed or compared products, visitors, event-specific categories, promotional rules, and so on).
- **Stop all administrative activity** on both Magento 1.x and Magento 2.0 during your migration test runs and during the actual data migration (including configuration changes; and maintaining customers, orders, and inventory).
- **For Magento Enterprise Edition, archive unarchived orders.**

Benchmarking Estimates

Migration Tested on Systems:

- **Environment:** Virtual Box VM, CentOS 6, 2.5Gb RAM, CPU 1 core 2.6GHz
- **Database** had 177k products, 355k orders, 214k customers

Performance Results:

- **Settings migration time:** ~10 mins
- **Data migration time:** ~9 hrs (all data except URL Rewrites, ~85% of total data)
- **Site downtime estimate:** A few minutes to reindex and change DNS settings. Additional time required to “warm up” the page cache.

Creating a Migration Plan

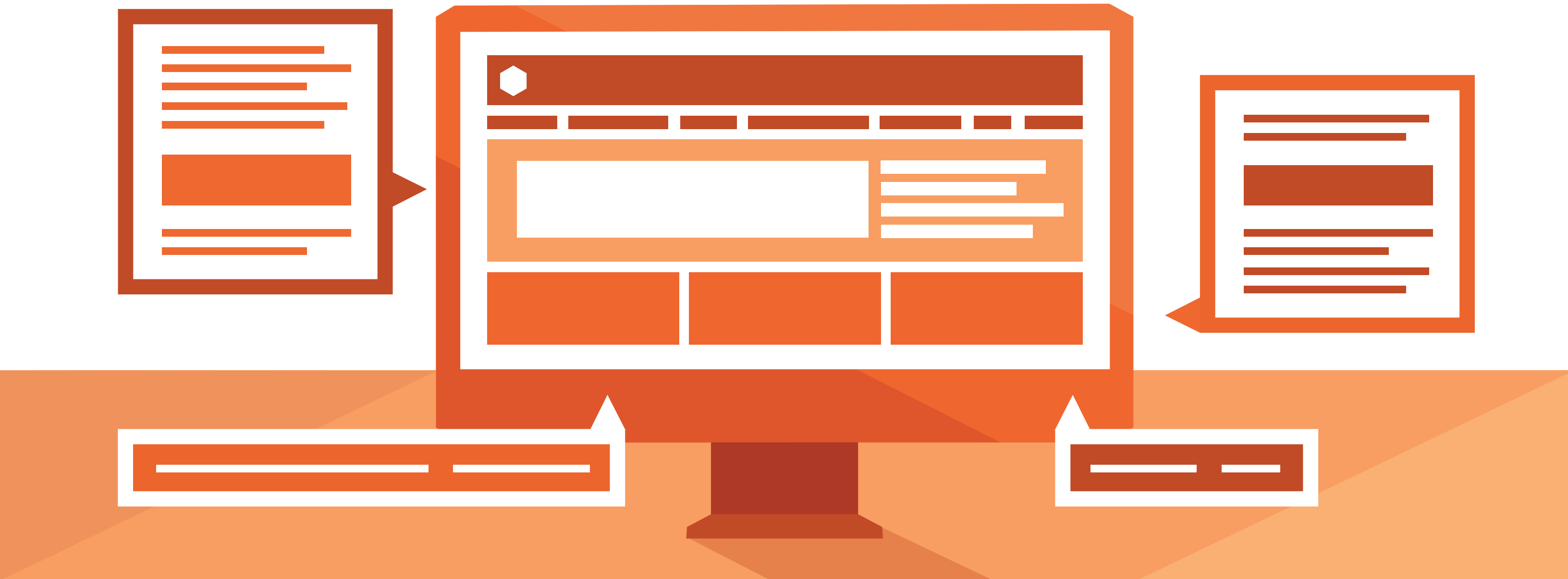
To have a successful migration, you must plan it and test it thoroughly. Use the following guidelines to get started.

Preconditions: Before You Run the Migration Tool

- Check that the data migration tool has a network connection to the Magento 1.x and Magento 2 databases.
- Open ports in your firewall so the migration tool can communicate with the databases and so the databases can communicate with each other.
- Disable Magento 2.0 cron jobs
- Set up the configuration and mapping files as discussed **here**

Step 1: Review Your Current Site

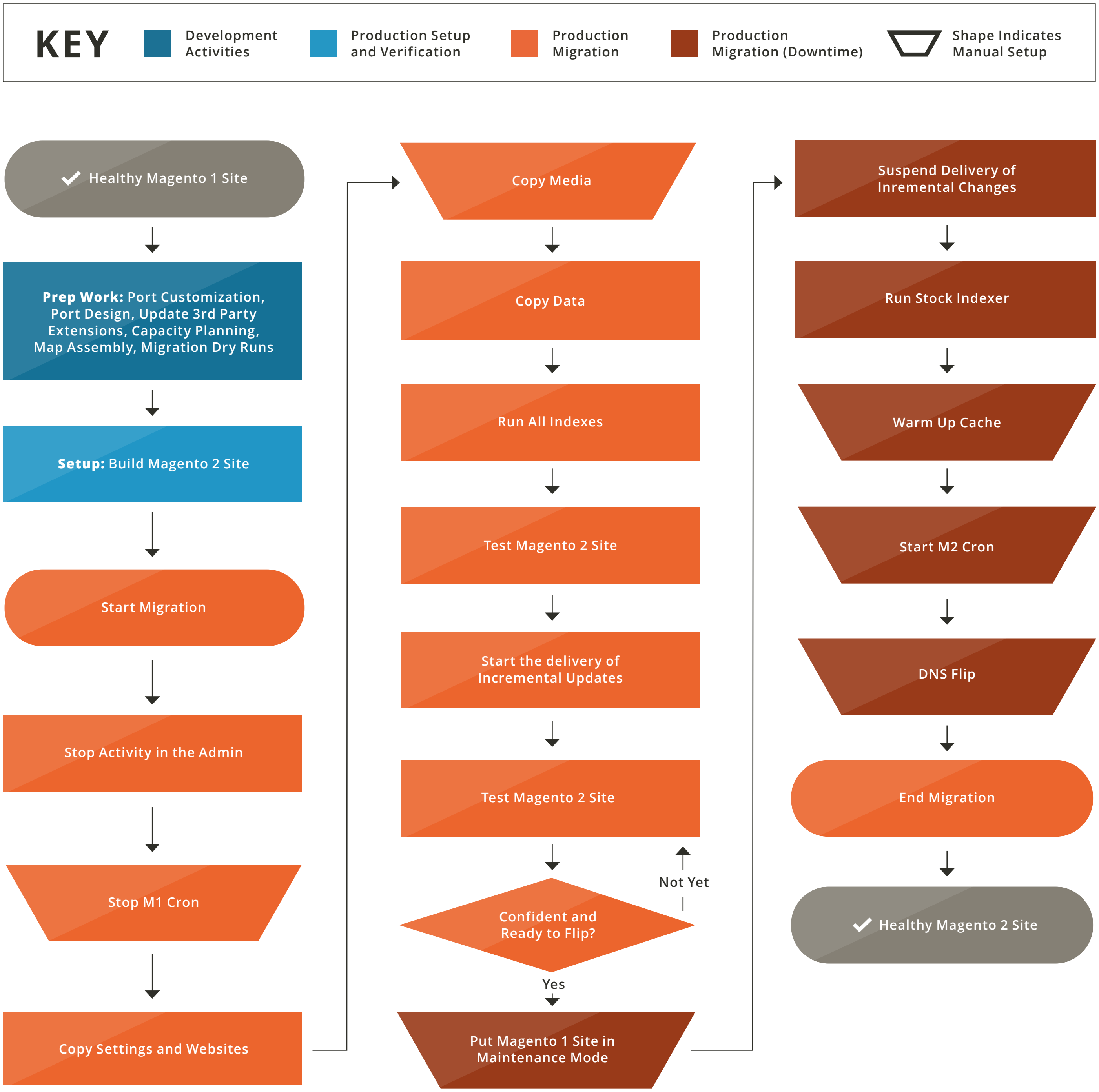
- What extensions have you installed?
- Have you identified if you need all these extensions in your new site? (There might be old ones you can safely drop.)
- Have you determined if Magento 2.0 versions your extensions exist? (Check with your extension providers to see if they have been ported yet.)
- What database assets from your extensions do you want to migrate?



Step 2: Capacity Planning

Consider whether the new site needs to be designed with more hardware or a more advanced topology with better caching tiers and so on. It's a good time to make more serious changes to get your site ready for your next level of growth.

The following figure shows the migration flow that is discussed in the remainder of this section.


KEY
Development Activities
Production Setup and Verification
Production Migration
Production Migration (Downtime)
Shape Indicates Manual Setup
✓ Healthy Magento 1 Site
Prep Work: Port Customization, Port Design, Update 3rd Party Extensions, Capacity Planning, Map Assembly, Migration Dry Runs
Setup: Build Magento 2 Site
Start Migration
Stop Activity in the Admin
Stop M1 Cron
Copy Settings and Websites
Copy Media
Copy Data
Run All Indexes
Test Magento 2 Site
Start the delivery of Incremental Updates
Test Magento 2 Site
Not Yet
Confident and Ready to Flip?
Yes
Put Magento 1 Site in Maintenance Mode
Suspend Delivery of Inremental Changes
Run Stock Indexer
Warm Up Cache
Start M2 Cron
DNS Flip
End Migration
✓ Healthy Magento 2 Site

Step 3: Build and Test Magento 2

To prepare for the migration, make sure you do all of the following:

- **Set up a Magento 2.0 system** that meets our **system requirements** and that uses a topology and design that at least matches your existing Magento 1.x system.
- **Reduce network latency** between your Magento 1.x and 2.0 systems as much as possible to improve migration performance.
- **To provide redundancy in the event of unexpected issues,** we advise you to replicate your Magento 1.x database. Use this replicated Magento 1.x database as the source data for your migration.

Step 4: Start Your Migration

To prepare for the migration, make sure you do all of the following:

1. **Stop all activity** in the Magento 1.x Admin Panel (except for order management and shipping). Activity cannot resume until your migration is complete.
2. **Stop all Magento 1.x** cron jobs.
3. **Use the migration tool** to migrate settings and websites.
4. **Copy your Magento 1.x media assets** to Magento 2.0. (You must copy these manually from **`<your Magento 1 install dir>/media`** directory to **`<your Magento 2 install dir>/pub/media`** directory.)
5. **Use the migration tool** to bulk copy your Magento 1.x database to your Magento 2.0 database. If some of your extensions have data you want to you migrate, you might need to use the mapping files provided with the migration tool.
6. **Reindex all Magento 2.0 indexers.** For details, see the Configuration Guide.
7. **Thoroughly test your Magento 2.0 site.** Make sure you can place orders using all configured payment processors.

Step 5: Incremental Updates

Now that you've migrated the bulk of your data, you must incrementally capture data updates (such as new or updated customers, inventory changes, and orders) from Magento 1.x to Magento 2.0.

- **Start the incremental migration;** updates run continually. You can stop the updates at any time by pressing Control+C.
- **Thoroughly test your Magento 2 site** during this time so you can catch any issues as soon as possible. If you find issues, press Control+C to stop incremental migration and start it again after issues are resolved.

Step 6: Go Live

Now that your Magento 2 site is up-to-date with Magento 1 and is functioning normally, do the following to cut over to the new site:

1. Put your Magento 1 system in maintenance mode ***(downtime starts)***.
2. Press Control+C in the migration tool command window to stop incremental updates.
3. Start your Magento 2 cron jobs.
4. In your Magento 2 system, reindex all indexers. For more information, see the **Configuration Guide**.
5. Using a tool of your choice, hit pages in your Magento 2 system to cache pages in advance of customers using your storefront.
6. Perform any final verification of your Magento 2 site.
7. Change DNS, load balancers, and so on to point to new production hardware ***(downtime ends)***
8. Resume activity in the Magento Admin.

How Migration Works

This section previews your migration experience by providing a high-level migration tool overview.

Terms Used

This document uses the following terminology to discuss the migration tool:

- **Step:** A unique migration task that must be executed in a prescribed order.
- **Map:** A set of rules that describes connections between Magento 1.x and Magento 2.0 data structures.
- **Mode:** Represented by a separate migration tool command, defines the basic mode of operation as:

    Settings: Migrates the system configuration and website-related settings.

    Data: Migrates database assets in bulk.

    Delta: Migrates incremental changes (for example, orders and inventory).

Disclaimer

Any specific information about the migration command-line tool is speculative. The tool is in development and some options could change.

Conceptual Overview

Each mode has several *steps*; steps might include data migration and integrity checking. Mapping files enable you to specify how custom database data is moved. (For example, if you have custom data you can move it as-is or you can rename tables and fields in the Magento 2 database). The declarative language used in the mapping files makes it as easy as possible for you to map your custom data.

All Magento 1.x core tables are automatically migrated to Magento 2.0 with no action required; you need to map only your custom database tables. In addition, any custom tables that do not change between Magento 1.x and Magento 2.0 are migrated automatically with no action required.

Mapping files are discussed in more detail **here**.

What Needs to be Manually Migrated

Extensions and custom code

To take advantage of the architectural advances in Magento 2.0, developers need to manually port custom code and extensions to Magento 2.0.

Media on the file system

Media assets include, for example, images for products, categories, WYSIWYG editor, and so on. You must copy these manually from **`<your Magento 1.x install dir>/media directory to<your Magento 2.0 install dir>/pub/media`** directory.

Storefront design

Cascading Stylesheets (CSS), JavaScript, templates, and XML layouts are implemented differently in Magento 2 and must be migrated manually.

Layouts

Layout updates implemented in Magento 1.x cannot be used in Magento 2.0. You must migrate the following manually:

- XML in CMS category pages in the Magento Admin
- Layout updates specified in widget instances

Web services credentials

You must manually create credentials for SOAP, XML-RPC and REST in Magento 2.0.

Data you should not migrate at all

For the reasons stated, you should not migrate the following data:

- **Indexed data:** You should perform a full reindex before you enable your Magento 2.0 on the production sever.
- **Google Shopping**: Google Shopping shipped in some older Magento 1.x versions but has since been removed.
- **Data that is not supported in Magento 2.0**: Poll, tag, staging modules, and recurring profiles are not currently supported in Magento 2.0.

Database Configuration and Mapping Files

The migration tool is controlled by a configuration file that defines data migration rules.

Using a configuration file and mapping files, you can:

- Automatically migrate tables and data for all tables that did not change between Magento 1.x and Magento 2.0.
- Migrate Magento 1.x tables that have different names in Magento 2.0 (for example, the Magento 1.x **`sales_flat_shipment_comment`** table migrates to the Magento 2.0 **`sales_shipment_comment`** table)
- Modify column names (for example, the **`cms_page->root_template`** Magento 1.x column migrates to **`cms_page->page_layout`** in Magento 2.0)
- Modify field values (for example, the Magento 1.x **`cms_page->root_template->two_columns_right`** field migrates to **`cms_page->page_layout->2columns-right`** in Magento 2.0)
- Ignore tables or columns that you don't want to migrate

Note: The preceding list uses examples of tables, columns, and fields provided with Magento 1.x and Magento 2.0. You can perform the same tasks for all of your custom database tables as well.

```
        string NumberStr = Html.
        return Convert.ToInt32(NumberStr);
    }
    return 0;
}
private string FindContributorName(string Html)
{
    string Pattern = "class=\"avatar \" alt=\"";
    int SectionStart = Html.IndexOf(Pattern);
    if (SectionStart != -1)
    {
        SectionStart += Pattern.Length;
        int NameStart = SectionStart;
        int NameEnd = Html.IndexOf("\"", SectionStart);
        if (NameStart != -1 && NameEnd != -1 && NameEnd > NameStart)
        {
            string Name = Html.Substring(NameStart, NameEnd - NameStart);
            return Name;
        }
        else
        {
            logger.AddLine("Contributor Name wasn't found");
        }
    }
    return "";
}
private string FindContributorWebsite(string Html)
{
    string Pattern = ">Business Website: <a href";
    int SectionStart = Html.IndexOf(Pattern);
    if (SectionStart != -1)
    {
```

```
    procedure ApplyClick(Sender: TObject);
    procedure CancelClick(Sender: TObject);
    procedure SearchClick(Sender: TObject);
    procedure OnNextIteration(Sender: TObject; ACount: Integer);
    procedure AddNode(ANode: TUFONode; ACount: Integer);

    procedure DeleteOldElement;
    procedure AddLink(AFrom, ATo: TUFOPortOnDiagram);
    procedure CreateLinks;
    procedure AddNewElement;
    procedure AddNewElementPos: TPoint;
    function CalcNewElementPlace(var APt: TPoint);
    procedure FindUFOElementPlace(AItem, AData: TObject; out

    procedure TryReconnectLink(AItem, AData: TObject; out
    procedure ReconnectLinks;
  protected
    procedure OnElementsMouseUp(Sender: TObject; Button: T
      Shift: TShiftState; X, Y: Integer);
  public
    constructor CreatePage(AOwner: TComponent; AFunction:
      ADiagramView: TUFODiagramView);
    destructor Destroy; override;
    procedure Init(APropEditor: TUFOPropEditor); override;

    procedure BeginAddPorts;
    procedure AddPort(APort: TUFOPortOnDiagram);
    procedure EndAddPorts;

    function OnEvent(ASender: TObject; ASource: TUFOEventS
      AEventCode: Integer; AData: TObject): Integer; overr
  end;
{$ENDIF}
```

Configuration File

The configuration file provides connection information about your Magento 1.x and Magento 2.0 databases.

Sample:

```
<config xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="../config.xsd">
    <steps mode="data">
        <step>Migration\Step\Eav</step>
    </steps>
    <steps mode="delta">
        <step>Migration\Step\Map</step>
        <step>Migration\Step\Log</step>
…
    </steps>
    <steps mode="settings">
        <step>Migration\Step\Settings</step>
        <step>Migration\Step\Stores</step>
    </steps>
    <source version="1.9.1">
        <database host="localhost" name="magento1" user="root"/>
    </source>
    <destination version="2.0.0.0">
        <database host="localhost" name="magento2" user="root"/>
    </destination>
    <options>
        <map_file>etc/ce-1.9.1/map.xml.dist</map_file>
        <eav_map_file>etc/ce-1.9.1/map-eav.xml.dist</eav_map_file>
…
        <deltalog_map_file>etc/ce-1.9.1/deltalog.xml.dist</deltalog_map_file>
        <class_map>etc/class_map.php</class_map>
        <bulk_size>100</bulk_size>
        <source_prefix />
        <dest_prefix />
        <auto_resolve_urlrewrite_duplicates>0</auto_resolve_urlrewrite_duplicates>
    </options>
</config>
```

Mapping Files

Following is a sample mapping file that:

- Transforms all Magento 1.x database tables and fields that have Magento 2.0 equivalents
- Renames fields from the Magento 1.x database in the Magento 2.0 database
- Ignores fields

```
<settings xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="../settings.xsd">
    <key>
        <ignore>
            <path>advanced/modules_disable_output/Mage_Connect</path>
        </ignore>
        <ignore>
            <path>advanced/modules_disable_output/Mage_Contacts</path>
        </ignore>
…
        <rename>
            <path>carriers/dhlint/gateway_url</path>
            <to>carriers/dhl/gateway_url</to>
        </rename>
        <rename>
            <path>carriers/dhlint/handling_action</path>
            <to>carriers/dhl/handling_action</to>
        </rename>
        <rename>
            <path>carriers/dhlint/handling_fee</path>
            <to>carriers/dhl/handling_fee</to>
        </rename>
        <rename>
…
    <value>
        <transform>
            <path>catalog/category/root_id</path>
            <handler class="\Migration\Handler\Settings\CategoryRootId"/>
        </transform>
        <transform>
            <path>catalog/seo/category_url_suffix</path>
            <handler class="\Migration\Handler\Settings\UrlSuffix"/>
        </transform>
        <transform>
            <path>catalog/seo/product_url_suffix</path>
            <handler class="\Migration\Handler\Settings\UrlSuffix"/>
        </transform>
    </value>
</settings>
```

Summary

We're excited you're ready to move to the latest in eCommerce technology, Magento 2.0! Data migration is a key component of the overall Magento 2.0 migration process, and the Magento 2.0 Migration Tool should give you confidence that your data will be transferred efficiently and reliably, with minimal downtime, so that your new site will be ready for action.

Don’t forget we have the following resources available for you to help you with your migration:

- **Magento Support**
- Magento Migration Guide (available at the same time as the migration tool)
- Documentation for creating Magento 2.0 **themes** and **layouts**
- **Magento 2 Extension Developer Guide**