# Data Migration from Magento 1 to Magento 2

## Including ParadoxLabs Authorize.Net CIM Plugin

Last Updated 2018-11-30

*This guide was contributed by a community developer for your benefit.*

## Background

Magento 2 includes a lot of new functionality and enhancements, but these new features required massive changes under the hood to both the application codebase as well as the database structure used by the store. In order to make migrating from Magento 1.x (hereafter M1) to Magento 2 (hereafter M2) easier, a data migration tool was developed that will migrate existing store data (sales, customers, products, etc) from M1 to M2. This migration tool is set up by default to migrate a vanilla M1 store, but if extra plugins have been installed that altered the database in any way, that plugin data will not migrate correctly, and in fact will cause the migration tool to fail before it migrates any data at all.

The purpose of this guide is to explain how to set up the migration tool to specifically account for database changes introduced by the ParadoxLabs Authorize.Net CIM plugin.

## Prerequisites and Assumptions

It is assumed that the upgrade for this process will be starting from a version of M1 that is supported by the migration tool. At the time of this writing that includes community edition versions 1.6-1.9. The specific examples provided in this document are upgrading from CE 1.7.0.2 to CE 2.0.0.

It is further assumed that the data migration tool has already been downloaded and installed into a fresh M2 installation. Further information on the migration tool, including complete instructions for how to download, install, and set it up can be found in the Magento Developer Documentation here: http://devdocs.magento.com/guides/v2.0/migration/bk-migration-guide.html

Finally, it is assumed the data migration will be taking place on a Linux host, with command-line access, and that the database for both M1 and M2 is MySQL and both the M1 and M2 databases are in the same MySQL instance.

## Process Overview

In a nutshell, we'll be doing the following:

- Set up a clean M2 instance, install the migration tool
- Prepare the M2 database
- Configure migration mapping for the copied table and the new fields for the migration tool.
- Cross our fingers and run the migration tool.

## Getting Started

The M1 to M2 data migration tool relies heavily on xml configuration files for its operation. There are separate configurations for mapping database tables and columns, EAV entity mapping, EAV document mapping, EAV attribute groups, log conversions, store settings, customer maps, etc. The migration tool comes with default versions of all these xml files, most of which you can use directly. Exactly which configuration files will be used is specified in one master configuration file, which is specified as an argument to the data migration tool on the command line. There are different master configuration files for each version of M1 that will be migrated, and they are found here:

```
[path to M2]/vendor/magento/data-migration-tool/etc/ce-to-ce/[version number]/config.xml.dist
```

In our case, the master configuration file is here:

```
[path to M2]/vendor/magento/data-migration-tool/etc/ce-to-ce/1.7.0.2/config.dist.xml
```

It is a best practice to make a copy of the distribution version of this config file to modify for the migration process. We'll name our copy config.xml, and keep it in the same directory.

There are only a few things we need to update in the config.xml file: We need to tell it where our source M1 database is, and where the M2 database is, as well as how to map fields from one to the other. Opening up the config.xml file and scrolling to near the bottom, we find where the "source" and "destination" sections are set up, as well as where the mapping begins:

```
<source>
  <database host="localhost" name="magento1" user="root"/>
</source>
<destination>
  <database host="localhost" name="magento2" user="root"/>
</destination>
<options>
  <map_file>etc/ce-to-ce/1.7.0.2/map.xml.dist</map_file>
```

Fill in correct credentials for the M1 and M2 databases (it is recommended that you do the migration on a copy of your M1 database).

Next turn our attention to the "options" section. This section will be used to specify the specific xml file we want to use for each of the different kinds of mapping the migration tool can accomplish. Note that there are more than a dozen xml files that can be specified here for various migration activities, but for our purposes we only care about the setting for map_file. Make a copy of the distribution map.xml.dist file, and update the setting in the config.xml file to match the new map.xml file name. In our case, we'll just copy map.xml.dist to map.xml, and update the config.xml file to be:

```
<options>
    <map_file>etc/ce-to-ce/1.7.0.2/map.xml</map_file>
```

## Updating the M2 Database

The migration tool does not create or rename tables or columns in either the M1 or M2 database - it only moves data from M1 into M2 using the defined mappings in the various XML files. This means that before we can migrate any data from M1 to M2, we must alter the M2 schema to have the destinations already in place for the M1 data. The simplest way to do this is to install the ParadoxLabs Authorize.Net CIM extension on M2 before continuing.

Purchase and download the M2 version of the extension, and follow the installation directions in the User Manual.

## Setting Up Field Mappings

Open the map.xml file. It defines mappings for both the **source** M1 database and the **destination** M2 database. You can define mappings at the table or field level. Tables are referred to as a "document" in the xml file. Options for mappings include:

- Ignore - document or field will not be migrated from M1 to M2
- rename - used to rename a document or field in M1 to a new document or field name in M2.
- move - used to move a specific database field from a table in M1 to a different table in M2.
- transform - used to migrate fields following behavior described in a handler
- handler - describes a transformation behavior for fields.

More information on the mapping types can be found here:
http://devdocs.magento.com/guides/v2.0/migration/migration-tool-internal-spec.html#configuration

For the CIM module, there are only a few fields and tables we need to worry about:

- customer attribute authnetcim_profile_id
- customer attribute authnetcim_profile_version
- column sales_flat_quote_payment.tokenbase_id
- column sales_flat_order_payment.tokenbase_id
- table paradoxlabs_stored_card

The customer attributes will already be converted correctly by the built-in EAV migration, as they follow the standard M1 EAV rules. The paradoxlabs_stored_card table has already been manually copied. So that leaves us with 3 items to map: tokenbase_id in both the sales and quote table, and the paradoxlabs_stored_card table. NOTE: the M1 sales_flat* tables have been reorganized and moved for M2. The new tables are named quote_payment and sales_order_payment.

Here are the mappings that need to be added to the source section of the map.xml file.

First, enter these mappings under source/field_rules:

```
<move>
  <field>sales_flat_quote_payment.tokenbase_id</field>
  <to>quote_payment.tokenbase_id</to>
```

```
</move>
<move>
   <field>sales_flat_order_payment.tokenbase_id</field>
   <to>sales_order_payment.tokenbase_id</to>
</move>
```

Second, add these transformations, again under source/field_rules: These instruct the migration tool to convert card addresses and additional data from serialized format to JSON.

```
<transform>
   <field>paradoxlabs_stored_card.address</field>
   <handler class="Migration\Handler\SerializeToJson"/>
</transform>
<transform>
   <field>paradoxlabs_stored_card.additional</field>
   <handler class="Migration\Handler\SerializeToJson"/>
</transform>
```

With these changes complete, we are ready to attempt to migrate our data. Log onto the server as the web server user, and change to the directory where M2 is installed. Then fire off this command:

```
./bin/magento migrate:data /[path to M2]/vendor/magento/data-migration-tool/etc/ce-to-ce/1.7.0.2/config.xml
```

Depending on the size of the source M1 schema, this may take anywhere from a few minutes to a few hours to complete. You can add a -vvv to the command to turn on debugging and very verbose logging in the migration tool, though this may cause it to fail erroneously, as it is more strict. Once the migration is complete, your M1 data should all be available in the M2 schema, including the data that is needed for the ParadoxLabs CIM module.

## Troubleshooting Note

At the time of this writing, there is a bug in the migration tool that does not do a correct comparison on the order of some fields in some tables, and even though your settings and mappings are all correct,


you'll get an error like this one when you try to run the migrator:

If you get an error similar to this, you likely need to apply the patch described here:
https://github.com/magento/data-migration-tool-ce/pull/25/files

(i.e., open src/Migration/Step/UrlRewrite/Version191to2000.php and replace lines 148-151 with the new lines from the commit listed at the link above).

This issue may be resolved in the version of the migrator you try to run.

## Questions? Confusion? Frustrations?

This process is not really an easy one to understand, and you will most likely run into problems. Magento has provided excellent documentation for their Migration Tool here:
http://devdocs.magento.com/guides/v2.0/migration/bk-migration-guide.html