**Magento Open Known Issues**

**1) Magento 2 REST API - Issues with Cart Operations (Issue #34392)**

* **Check the Magento version:** php bin/magento --version
* **Review API Documentation:** Ensure you are following the correct structure in your API calls for adding, updating, or retrieving cart information.
* **Verify the REST API Call:** The common endpoints involved in cart operations are:

**Create a Cart for Guest Users**: /V1/guest-carts

**Add a Product to the Cart**: /V1/guest-carts/{cartId}/items

**Update a Product in the Cart**: /V1/carts/{cartId}/items

* Ensure that your payload structure matches Magento’s requirements.

{

"cartItem": {

"quote\_id": "{cartId}",

"sku": "product\_sku",

"qty": 1

}

}

* **Error:** Could not add item to cart
* **Debugging Common** **Errors :** php bin/magento deploy:mode:set developer  
  php bin/magento setup:config:set --enable-debug-logging  
  tail -f var/log/debug.log
* **Fix Issues with Guest Cart Session:** Magento requires a valid cart session for both logged-in users and guests. If you're using guest carts, ensure you are generating a cart before adding items.  
  POST /V1/guest-carts
* **Fix the Quote Merge Issue for Logged-in Users**  
   For logged-in users, issues can occur when merging guest and customer quotes. To address this:Clear the existing cart data to avoid any conflict:  
   $this->cart->truncate();

$quoteId = $this->customerCartManagement->getCartForCustomer($customerId);

$this->quoteRepository->get($quoteId)->merge($guestQuote);

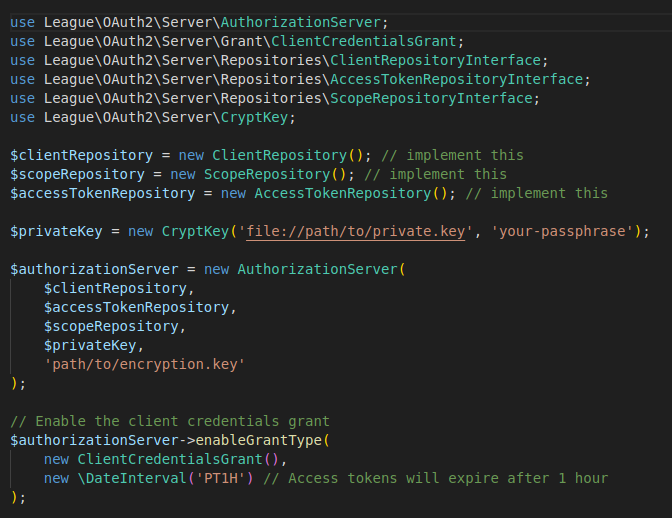
* **Clear Cache and Reindex**

php bin/magento cache:clean

php bin/magento indexer:reindex

* By following these steps, you can resolve the cart operation issues using the Magento 2 REST API.
* The most common issues involve quote generation, session handling, and payload structure.
* Debugging and checking logs provide detailed insight into why cart operations might fail.

**2) OAuth 2.0 and OpenID support (Issue #34780)**

* Magento 2 does not provide native support for **OAuth 2.0 and OpenID** (GitHub Issue #34780), as it uses the older **OAuth 1.0a** protocol for API authentication.
* you can integrate **OAuth 2.0 and OpenID** support in Magento 2 using third-party libraries or custom modules.
* Install a PHP OAuth 2.0 Library: You can start by installing a library that supports OAuth 2.0. **League’s OAuth 2.0 server** is a popular option.
* composer require league/oauth2-server  
    
  
* **Integrate OpenID Connect (OIDC) :** OpenID Connect is built on top of OAuth 2.0. You can integrate OIDC by handling ID tokens and user authentication.
* Add OIDC endpoints such as /authorize and /token in your module's routes (routes.xml), and ensure you implement proper authentication and token generation.
* **Handle User Authentication (Login with OpenID):** Use an existing identity provider (IdP) that supports OpenID Connect, such as **Google** or **Auth0**, to authenticate users. Redirect users to the IdP for login, and upon success, they will be redirected back to Magento with a valid token.
* Example of a flow:Redirect the user to the IdP's authorization URL.The IdP will redirect back with an authorization code.Exchange the code for an access token via Magento’s /token endpoint.
* **Store and Validate Access Tokens:** Store access tokens securely in your database using Magento’s models and repositories. Ensure tokens are validated during every API request.
* **Add Middleware to Protect API Endpoints:** To ensure that API requests are only allowed for authenticated users, create middleware that checks the validity of the OAuth 2.0 tokens.
* 
* Use **Postman** or **OAuth 2.0** libraries to test your implementation. You should be able to: Register clients, Request access tokens. Access protected resources using the tokens.
* After implementing the OAuth 2.0 and OpenID support, clear Magento’s cache and reindex:

php bin/magento cache:clean ,php bin/magento indexer:reindex

* **Error** If there are issues with the implementation, common errors include:

**Invalid Grant Type**: Ensure you enable the correct OAuth 2.0 grants (e.g., authorization code, client credentials).

**Token Mismatch**: This could happen if the client and server encryption keys do not match or if tokens expire.

**Invalid Scopes**: Ensure that the correct scopes are requested in the authorization phase.

* Integrating OAuth 2.0 and OpenID Connect into Magento 2 involves setting up an OAuth 2.0 server, handling client authentication, and integrating OIDC for secure login and authorization.
* By following these steps, you should be able to enhance your Magento 2 instance with modern authentication mechanisms.

**3) Magento 2 Product Import Performance Issue (Issue #33034)**

* To resolve the **Product Import Performance Issue in Magento 2 (GitHub Issue #33034)**, the problem generally arises due to the large number of products being imported, inefficient indexing, database bottlenecks, or lack of optimization in import processes.
* Here is a step-by-step guide to improving Magento 2 product import performance.
* Update Magento to the Latest Version: Magento frequently optimizes performance in newer versions. If you are using an older version, it is recommended to update Magento 2 to the latest stable version.  
   composer require magento/product-community-edition 2.x.x --no-update

composer update

php bin/magento setup:upgrade

php bin/magento cache:flush

* **Increase PHP Memory Limit:** Magento imports can require significant memory. To improve import performance, increase the memory\_limit in php.ini.
* Open your php.ini file: sudo nano /etc/php/7.x/cli/php.ini
* Find the memory\_limit directive and increase it: memory\_limit = 2G
* sudo systemctl restart php7.x-fpm
* Optimize MySQL Configuration: Since product imports involve heavy database interactions, optimizing your MySQL settings can significantly improve performance. Adjust the following MySQL settings in my.cnf or my.ini:  
   **Increase InnoDB Buffer Pool Size:** This allows MySQL to load more data into memory.  
   innodb\_buffer\_pool\_size = 2G  
    
   Increase the Write Buffer Size: innodb\_log\_buffer\_size = 512M  
    
   Enable Query Cache (for large imports): query\_cache\_size = 128M

query\_cache\_type = 1  
 sudo systemctl restart mysql  
 php bin/magento indexer:reindex

sudo systemctl restart mysql

* **Use Asynchronous Bulk Product Import (Fast Method):** Magento provides a better, more scalable product import mechanism using the Asynchronous Import module.
* **Install the Asynchronous Import/Export module:** If you are using the Magento Commerce edition, it supports asynchronous imports. Install and configure this if not already enabled.
* **Use Bulk API Import for Product Imports:** You can use the bulk import functionality via the REST API for better performance. Here's an example of how to upload the import file:

Create a CSV file for your products.

Use Magento's Bulk Import REST API to import products asynchronously. The API reduces load by processing tasks in the background.

* **Enable Flat Catalog for Products**
* Enabling flat catalog structures in Magento can reduce the complexity of database queries during product imports.
* In the Magento Admin panel, go to: Stores > Configuration > Catalog > Catalog > Storefront
* Set **Use Flat Catalog Product** to **Yes**.
* **Enable Redis and Varnish Cache**
* Magento 2 relies heavily on caching. By using Redis for cache and session storage and Varnish for full-page caching, you can speed up operations, including imports.
* **Enable Redis and Varnish Cache**
* Magento 2 relies heavily on caching. By using Redis for cache and session storage and Varnish for full-page caching, you can speed up operations, including imports.
* **Install and configure Redis for Magento: sudo apt-get install redis-server**
* Configure Redis in app/etc/env.php:   
  'cache' => [

'frontend' => [

'default' => [

'backend' => 'Cm\_Cache\_Backend\_Redis',

'backend\_options' => [

'server' => '127.0.0.1',

'port' => '6379',

'persistent' => '',

'database' => '0',

'password' => '',

'force\_standalone' => '0',

'connect\_retries' => '1',

'read\_timeout' => '10',

'automatic\_cleaning\_factor' => '0',

'compress\_data' => '1',

'compress\_tags' => '1',

'compress\_threshold' => '20480',

'compression\_lib' => 'gzip',

],

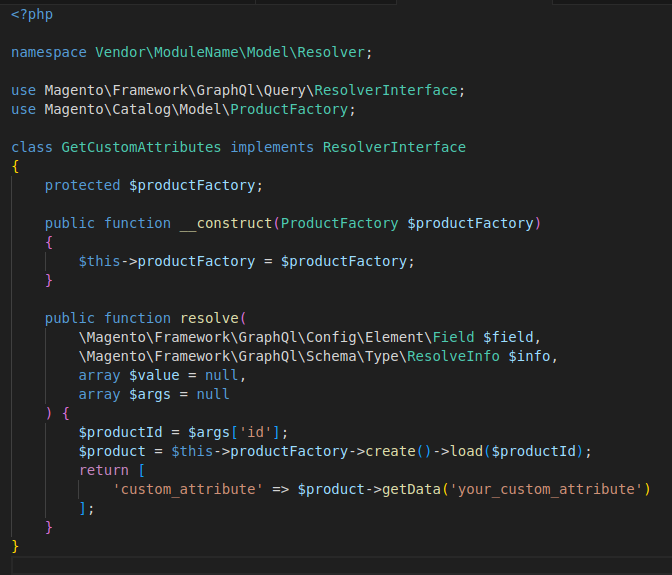
],

],

],

* **Use Third-Party Import Extensions**
* There are third-party extensions designed specifically for faster imports in Magento 2. Extensions like **Improved Import & Export** by FireBear Studio can significantly speed up imports by optimizing processes and offering features like multi-threading.
* Clear Cache After this step
* **Error Lines**If you face errors during the process, the following common areas can be problematic:**Memory Limit Exceeded**: PHP logs will show an error about exceeding memory.**Database Bottlenecks**: Slow query logs in MySQL can highlight problematic queries during imports.**Invalid CSV Format**: If there are format issues, Magento will log errors regarding CSV structure in var/log.
* Improving the performance of Magento 2 product imports involves both optimizing the server environment (PHP, MySQL) and utilizing Magento’s asynchronous import features.

**4) GraphQL Issues with Custom Product Attributes (Issue #33740)**

* To resolve the **GraphQL Issues with Custom Product Attributes (GitHub Issue #33740)** in Magento 2, follow the steps below. This issue is related to the inability to retrieve or query custom product attributes through GraphQL APIs.
* Verify Custom Attributes in Admin:Ensure that the custom product attributes are set up correctly in the Magento admin:
* Go to **Stores > Attributes > Product** in the Magento Admin panel.
* Search for the custom attribute that’s not working with GraphQL.
* Make sure the attribute has the following configurations:
  + **Scope**: Set to **Global** or the appropriate scope based on your store.
  + **Frontend Input Type**: Ensure the input type is valid (e.g., text, dropdown, etc.).
  + **Used in Product Listing**: Set to **Yes** if you want to expose this attribute in product listings.
  + **Used in Search**: If needed, set to **Yes** for search functionality.
* Extend GraphQL Schema
* Magento 2 GraphQL only exposes attributes that are part of its schema. To make a custom product attribute accessible via GraphQL, you need to extend the GraphQL schema.
* **Create or Modify the GraphQL Module**: If you already have a custom module, you can modify it. If not, create a new custom module.  
  app/code/Vendor/ModuleName/
* **Define the GraphQL Resolver**: In the custom module, create a resolver class to define how your custom attribute will be handled in GraphQL.
* 
* **Define GraphQL Schema for Custom Attributes**You need to add your custom attribute to the GraphQL schema.Create a **schema.graphqls** file in your custom module at:
* app/code/Vendor/ModuleName/etc/graphql/schema.graphqls
* Add the following schema definition to expose the custom attribute:

type Product {

custom\_attribute: String

}

extend type Query {

productCustomAttributes(id: Int!): Product

}

* Register the Resolver : To connect your resolver with the custom GraphQL query, register the resolver in di.xml.

Create or update the **di.xml** file in the following location:

<type name="Magento\Framework\GraphQl\Schema\Type\ResolveInfo">

<arguments>

<argument name="resolvers" xsi:type="array">

<item name="Query/productCustomAttributes" xsi:type="object">Vendor\ModuleName\Model\Resolver\GetCustomAttributes</item>

</argument>

</arguments>

</type>

* Flush magento cache
* Error   
  {

"errors": [

{

"message": "Cannot query field 'custom\_attribute' on type 'ProductInterface'.",

"extensions": {

"category": "graphql"

}

}

]

}

* Improving the performance of Magento 2 product imports involves both optimizing the server environment (PHP, MySQL) and utilizing Magento’s asynchronous import features.

5) Email Notification Issues in Magento 2.4.x (Issue #35200)

* This can occur due to misconfiguration in the email system, cron job issues, or a failure to properly connect with the SMTP server. Let’s break down how to solve the issue step-by-step, the potential error lines, and what kind of errors may appear when this issue arises.
* Error Messages:

"Email sending failed: Unable to send the email."

"Unable to connect to SMTP host."

* Check Magento Email Configuration

Admin Panel: Go to Stores > Configuration > Sales > Sales Emails.

Make sure **Enabled** is set to **Yes** for the relevant email types (e.g., order, invoice, shipment).

Sender Email: Ensure that the correct **email sender** and **email template** are selected.

Save the configuration.

* Set Up and Verify SMTP Server :Magento 2.4.x does not have an out-of-the-box SMTP server setup. If you're using an SMTP provider, ensure it's properly configured:

Install a third-party SMTP module (e.g., **Mageplaza SMTP** or **Amasty SMTP**).

Configure the SMTP settings in **Stores > Configuration > Advanced > System > Mail Sending Settings**:

* + Set **Host** to your SMTP provider's server (e.g., smtp.gmail.com).
  + Set the correct **port** (e.g., 465 for SSL or 587 for TLS).
  + Enter the **username** and **password** of your SMTP account.

Test the connection to make sure the SMTP provider works.

* Ensure Cron Jobs are Set Up :Magento uses cron jobs to send emails in the background. If your cron job is not set up correctly, email notifications may not work:Check if cron jobs are configured by running this command:

crontab –l

php bin/magento cron:install

php bin/magento cron:run

php bin/magento queue:consumers:start async.operations.all

php bin/magento queue:consumers:start async.operations.all

* Check Magento Email Logs: Magento keeps logs for system errors, including email issues. Check the logs to identify what’s going wrong:

Go to the **var/log** directory: cd var/log

Open **system.log** or **exception.log** to find any relevant email issues:

tail -f system.log

tail -f exception.log

* Check File and Folder Permissions: Incorrect file permissions can also prevent Magento from sending emails:

Ensure the proper permissions are set for the **var** and **generated** folders

* chmod -R 775 var generated
* php bin/magento cache:clean
* you should be able to resolve the email notification issues in Magento 2.4.x. If the issue persists, recheck the SMTP configuration, email templates, and logs for more detailed troubleshooting.