# TestHorse

Certified IT practice exam authority

Accurate study guides, High passing rate!
Testhorse provides update free of charge in one year!



http://www.testhorse.com

**Exam** : **AD0-E722**

**Title** : Adobe Commerce Architect Master

**Version** : DEMO

1.A company wants to build an Adobe Commerce website to sell their products to customers in their country. The taxes in their country are highly complex and require customization to Adobe Commerce. An Architect is trying to solve this problem by creating a custom tax calculator that will handle the calculation of taxes for all orders in Adobe Commerce.
Following best practices, how should the Architect add the taxes for all orders?
A. Add a new observer to the event sales.quote.collecLtotals.before" and add the custom tax to the quote
B. Write a before plugin to \Magento\Quote\Model\QuoteManagement::placeOrder() and add the custom tax to the quote
C. Declare a new total collector in "etc/sales.xmr in a custom module
**Answer:** C
**Explanation:**
According to the Adobe Commerce documentation, the best way to add a custom tax calculation to all orders is to declare a new total collector in the “etc/sales.xml” file of a custom module. This way, the custom tax logic can be implemented in a separate class that extends the \Magento\Quote\Model\Quote\Address\Total\AbstractTotal class and overrides the collect() and fetch() methods. The collect() method is responsible for calculating the tax amount and adding it to the quote address, while the fetch() method is responsible for displaying the tax amount in the cart and checkout pages. The new total collector can be assigned to any area of the order totals, such as before or after the subtotal, shipping, or grand total.
Reference: Customizing order totals
How to add custom fee or discount to order totals in Magento 2

2.An Adobe Commerce Architect is creating a new GraphQL API mutation to alter the process of adding configurable products to the cart. The mutation accepts configurable product ID. If the given product has only one variant, then the mutation should add this variant to the cart and return not nullable Cart type. If the configurable product has more variants, then the mutation should return not
nullable Conf igurableProduct type.
The mutation declaration looks as follows:

```
type Mutation {
    addConfigurableToCart(product_id: Int!): AddToCartOutput!
        @resolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddConfigurableToCart")
}
```

How should the Adobe Commerce Architect declare output of this mutation?
A)

```
union AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver")
= ConfigurableProduct | Cart
```

B)

```
interface AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver") {
}

type ConfigurableProduct implements AddToCartOutput {
}

type Cart implements AddToCartOutput {
}
```

C)

```
type AddToCartOutput {
    product: ConfigurableProduct
    cart: Cart
}
```

A. Option A
B. Option B
C. Option C
**Answer:** B
**Explanation:**
According to the Adobe Commerce documentation, the output of a GraphQL mutation is declared by specifying the type of the data returned by the mutation. The type can be either a scalar type (such as String, Int, Boolean, etc.), an object type (such as Cart, Product, Customer, etc.), or a union type (such as SearchResult, which can be either Product or Category). A union type is used when the mutation can return more than one possible type of data, depending on the input or the logic of the mutation. In this case, the mutation can return either a Cart type or a ConfigurableProduct type, depending on the number of variants of the configurable product. Therefore, the output of the mutation should be declared as a union type that includes both Cart and ConfigurableProduct types. Option B is the only option that correctly declares a union type using the pipe symbol (|) to separate the possible types. Option A and Option C are incorrect because they use brackets ([ ]) and curly braces ({ }), which are used for declaring list types and input object types, respectively.
Reference: GraphQL API - Adobe Inc.
Schema language with GraphQL | Adobe Commerce

3.A third-party company needs to create an application that will integrate the Adobe Commerce system to get orders data for reporting. The integration needs access to the GET /VI/orders endpoint. It will call this endpoint automatically every hour around the clock. The merchant wants the ability to restrict or extend access to resources as well as to revoke the access using Admin Panel.
Which type of authentication available in Adobe Commerce should be used and implemented in a third-party system for this integration?
A. Use token-based authentication to obtain the Admin Token. The third-party system will utilize the REST endpoint using the admin username and password to get the Admin Token, which will be used as the Bearer Token to authorize.
B. Use token-based authentication to obtain an integration Token, integration will be created and activated in the admin panel using default integration token settings to get access to the token, which will be used as the Bearer Token to authorize.
C. Use OAuth-based authentication to provide access to system resources. Integration will be registered by the merchant in the admin panel with an OAuth handshake during activation. The third-party system should follow OAuth protocol to authorize.
**Answer:** C
**Explanation:**
According to the Adobe Commerce documentation, OAuth-based authentication is the recommended method for integrations that need access to system resources, such as orders, customers, products, etc. OAuth-based authentication allows the merchant to control the access level and scope of the integration,

as well as to revoke the access at any time using the admin panel. OAuth-based authentication also requires an OAuth handshake between the integration and the Adobe Commerce system during activation, which ensures a secure exchange of tokens and keys. The third-party system should follow the OAuth protocol to obtain and refresh the access token, which will be used as the Bearer Token to authorize the REST API calls.
Reference: Authentication | Adobe Commerce Developer Guide OAuth-based authentication | Adobe Commerce Developer Guide

4.In a custom module, an Architect wants to define a new xml configuration file. The module should be able to read all the xml configuration files declared in the system, merge them together, and use their values in PHP class.
Which two steps should the Architect make to meet this requirement? (Choose two.)
A. Inject a "reader" dependency for "Magento\Framework\Config\Data" in di.xml
B. Write a plugin for \Magento\Framework\Config\Data::get() and read the custom xml files
C. Create a Data class that implements "\Magento\Framework\Config\Data"
D. Append the custom xml file name in "Magento\Config\Model\Config\Structure\Reader" in di.xml
E. Make a Reader class that implements '\Magento\Framework\Config\Reader\Filesystem"
**Answer:** CE
**Explanation:**
According to the Adobe Commerce documentation, to create a new xml configuration file, the Architect needs to create a Data class and a Reader class for the custom module. The Data class is responsible for storing and retrieving the configuration data from the cache or the Reader class. The Data class should implement the “\Magento\Framework\Config\Data” interface or extend the “\Magento\Framework\Config\Data” class. The Reader class is responsible for reading and validating the xml configuration files from the file system. The Reader class should implement the '\Magento\Framework\Config\Reader\Filesystem" interface or extend the '\Magento\Framework\Config\Reader\Filesystem" class. The Architect also needs to declare the Data class and the Reader class in the di.xml file of the custom module, and specify the name of the xml configuration file, the converter class, and the schema locator class for the Reader class.
Reference: Configuration types | Adobe Commerce - Experience League Create configuration types | Adobe Commerce - Experience League

5.An Adobe Commerce Architect creates a stopword for the Italian locale named stopwordsjtJT.csv and changes the stopword directory to the following:
<magento_root>/app/code/Custo«vendor/Elasticsearch/etc/stopwords/
What is the correct approach to change the stopwords directory inside the custom module?
A. Add stopwords to the stopwordsDirectory and CustomerVendor_Elasticsearch to the stopword sModule parameter Of the \Magento\Elasticsearch\SearchAdapter\Query\Preprocessor\Stopwords ClflSS Via di.xml
B. Add a new ClaSS implementing \Magento\Framework\Setup\Patch\PatchInterface to modify the default Value Of elasticsearch\customer\stopwordspath in core.conf ig_data table.
C. Add stopwords to the stopwordsDirectory parameter of the\Hagento\Elasticsearch\Model\Adapter\Document\DirectoryBuilder ClaSS Via stopwords/it.xml and Adobe Commerce will automatically detect the current module.

**Answer:** A

**Explanation:**

According to the Adobe Commerce documentation, the correct approach to change the stopwords directory inside a custom module is to use dependency injection to override the default values of the stopwordsDirectory and stopwordsModule parameters of the \Magento\Elasticsearch\SearchAdapter\Query\Preprocessor\Stopwords class. The stopwordsDirectory parameter specifies the relative path of the stopwords directory from the module directory, while the stopwordsModule parameter specifies the name of the module that contains the stopwords directory. By adding these parameters to the di.xml file of the custom module, the Architect can change the location of the stopwords files without modifying the core code or database.

Reference: To change the directory from your module

Configure Elasticsearch stopwords