DUMPSARENA

# Adobe Commerce Architect Master

## Adobe AD0-E718

## Version Demo

**Total Demo Questions: 10**

**Total Premium Questions: 50**

**Buy Premium PDF**

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

## QUESTION NO: 1

An Adobe Commerce Architect needs to set up two websites on a single Adobe Commerce instance with base URLs: example.com and website2.example.com.

How should the Architect configure this project so that both websites can use the same customer base?

**A.** Change Session Cookie attribute to "SameSite=None"

**B.** Disable Session Validation for "HTTP_X_FORWARDED_FOR" header

**C.** Set Cookie Domain for both websites to ".example.com"

## ANSWER: C

**Explanation:**

By setting the same cookie domain for both websites, the customer base can be shared between both websites, as the customer will be authenticated by the same cookie across both sites. This will ensure that customers don't have to log in twice when switching between the two sites.

Setting Cookie Domain for both websites to ".example.com" will allow both websites to use the same customer base. This is because the cookie domain determines which websites can access the customer information stored in the cookie. By using a common domain, both websites can share the same customer cookie. See Multiple websites or stores in the Adobe Commerce Help Center1. References: https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/multi-sites/ms-overview.html?lang=en1

## QUESTION NO: 2

Since the last production deployment, customers can not complete checkout. The error logs show the following message multiple times:

```
main.CRITICAL: Report ID: webapi-61b9fe83f0c3e; Message: Infinite loop detected, review the
trace for the looping path
```

The Architect finds a deployed feature that should limit delivery for some specific postcodes.

The Architect sees the following code deployed in/webapi_rest \di .xml and etc\frontend\di xml

```
<type name="Magento\Shipping\Model\Rate\Result">
    <plugin name="RestrictDeliveryMethods" type="Vendor\RestrictDeliveryMethods\Plugin\Shipping\LimitRates"/>
</type>
```

LimitRates.php:

```
public function __construct(
    \Magento\Checkout\Model\Session $session,
    ResultProvider $resultProvider
) {
    $this->session = $session;
    $this->resultProvider = $resultProvider;
}

public function afterGetAllRates(\Magento\Shipping\Model\Rate\Result $subject, array $result): array
{
    return $this->resultProvider->getLimitedRates($this->session->getQuote(), $result);
}
```

Which step should the Architect perform to solve the issue?

**A.** Inject an instance of \Magentro\Quote\API\CartRepostoryInterface and receive cart instance via$this->cartRepository->get($this-session->getQucteId())

**B.** Replace the injected dependency \Magento\Checkout\Model\Session\With\Magento\Framework\Session\SessionManagerInterface

**C.** Change 'after' plugin with 'around' plugin. The issue is being caused by calling the result provider code after the code of the original method.

**ANSWER: C**

**Explanation:**

The ‘after’ plugin is not suitable for modifying the arguments or return value of the original method. The ‘around’ plugin allows the plugin method to wrap around the original method and modify its behavior and output. References: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/plugins.html#around-methods

## QUESTION NO: 3

An Adobe Commerce Architect is asked by a merchant using B2B features to help with a configuration issue.

The Architect creates a test Company Account and wants to create Approval Rules for orders. The Approval Rules tab does not appear in the Company section in the Customer Account Menu when the Architect logs in using the Company Administrator account.

Which two steps must be taken to fix this issue? (Choose two.)

**A.** Set 'Enable Purchase Orders' in the B2B Admin to TRUE

**B.** Merchant needs to log out of frontend and then log back in to load new permissions

**C.** Set Enable Purchase Orders' on the Company Record to TRUE

**D.** Make sure that the 'Purchase Order' payment method is active

**E.** Set 'Enable B2B Quote" in the B2B Admin to TRUE

**ANSWER: A C**

**Explanation:**

Enabling Purchase Orders at both the B2B Admin and the Company Record levels is necessary for Approval Rules to appear in the Company section of the Customer Account Menu. When 'Enable Purchase Orders' is set to TRUE, the system assumes that the company will be making purchases using purchase orders, and the Approval Rules tab becomes visible.

To create Approval Rules for orders, the Architect needs to enable Purchase Orders both in the B2B Admin and on the Company Record. This will allow the Company Administrator to access the Approval Rules tab in the Customer Account Menu and create rules based on various criteria. The Purchase Order payment method and the B2B Quote feature are not required for this functionality. References: https://docs.magento.com/user-guide/customers/account-dashboard-approval-rules.html

## QUESTION NO: 4

An Architect wants to create an Integration Test that does the following:

• Adds a product using a data fixture

• Executes $this->someLogic->execute($product) on the product

• Checks if the result is true.

Sthis->someLogic has the correct object assigned in the setup () method-Product creation and the tested logic must be executed in the context of two different store views with IDs of 3 and 4, which have been created and are available for the test.

How should the Architect meet these requirements?

**A.** Create one test class with one test method. Use the \Magento\testFramework\ store\Executionstorecontext class once in the fixture and another time in the test.

**B.** Create two test Classes With one test method each. Use the @magentoExecuteInStoreContext 3 and @magentoExecuteInStoreContext 4 annotations on the class level.

**C.** Create one test class with two test methods. Use the @magentoStoreContext 3 annotation in one method and @magentoStoreContext 4 in the other one.

## ANSWER: C

**Explanation:**

The @magentoStoreContext annotation allows the test to run in the context of a specific store view. This annotation can be used on the method level to specify different store views for different test methods. This way, the product creation and the tested logic will be executed in the context of the same store view for each test method. References: https://devdocs.magento.com/guides/v2.4/test/integration/annotations/magento-store-context.html

## QUESTION NO: 5

A developer needs to uninstall two custom modules as well as the database data and schemas. The developer uses the following command:

bin/magento module:uninstall Vendor_SampleMinimal Vendor_SampleModifyContent

When the command is run from CLI, the developer fails to remove the database schema and data defined in the module Uninstall class.

Which three requirements should the Architect recommend be checked to troubleshoot this issue? (Choose three.)

**A.** remove-schema and --remove-data options are specified as arguments for the CLI command

**B.** bin/magento maintenance: enable command should be run in CLI before

**C.** composer.json file is present and defines the module as a composer package

**D.** Invoke uninstallData() and uninstallSchema () are defined in the Uninstall class

**E.** --remove-data option is specified as an argument for the CLI command

**F.** invoked uninstall () method is implemented in the Uninstall class

**ANSWER: A D E**

**Explanation:**

To troubleshoot the issue of failing to remove the database schema and data using the bin/magento module:uninstall command, the following requirements should be checked: A. Check if the remove-schema and --remove-data options are specified as arguments for the CLI command. These options will remove the schema and data for the specified module. D. Confirm that the uninstallData() and uninstallSchema() methods exist in the Uninstall class. These methods are responsible for removing the database schema and data. E. Check if the --remove-data option is specified as an argument for the CLI command. This option will remove the data for the specified module.

## QUESTION NO: 6

An Adobe Commerce system is configured to run in a multi-tier architecture that includes:

• A cache server with Varnish installed

• A backend web server with Adobe Commerce installed

• A database server with MySQL installed

When an Adobe Commerce Architect tries to clean the cache from the Store Admin by using the "Flush Magento Cache" in Cache Management, the Full Page Cache does not clear.

Which two steps should the Architect take to make the Full Page Cache work properly? (Choose two.)

**A.** Set the backend destination host to the frontend server's address in the Store Admin
Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Host

**B.** Set the backend port destination to the frontend server's Varnish port in the Store Admin
Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Port

**C.** Set the cache type to "Varnish Caching" in the Store Admin.
Stores > Configuration > Advanced > System > Full Page Cache > Caching Application

**D.** Use "Flush Cache Storage" instead of "Flush Magento Cache"

**E.** Set the cache destination host using magento CLI bin/magento setup:config:set --http-cache-hosts:

**ANSWER: C E**

**Explanation:**

To use Varnish as the full page cache, the cache type must be set to “Varnish Caching” in the Store Admin. This will enable the “Flush Magento Cache” button to send a purge request to Varnish. Additionally, the cache destination host must be specified using the magento CLI command to tell Magento which Varnish servers to purge. References: https://devdocs.magento.com/guides/v2.4/config-guide/varnish/config-varnish.html

## QUESTION NO: 7

An Adobe Commerce Architect is working on a scanner that will pull prices from multiple external product feeds. The Architect has a list of vendors and decides to create new config file marketplacejeeds.xml.

Which three steps can the Architect take to ensure validation of the configuration files with unique validation rules for the individual and merged files? (Choose three.)

**A.** Implement validation rules in the Converter class for the Config Reader

**B.** Add the Uniform Resource Name to the XSD file in the config XML file.

**C.** Provide schema to validate a merged file.

**D.** Provide schema to validate an individual file.

**E.** Create a class that implements \Magento\Framework\Config\DataInterface.

**F.** Create validation rules in marketplace.schema.xsd.

**ANSWER: B C E**

**Explanation:**

To ensure validation of the configuration files with unique validation rules for the individual and merged files, the Architect can take the following steps: Add the Uniform Resource Name to the XSD file in the config XML file. This will allow the configuration file to reference the schema that defines its structure and validation rules. Provide schema to validate a merged file. This will allow the configuration object to validate the merged configuration data from all modules. Provide schema to validate an individual file. This will allow the configuration object to validate each module’s configuration data before merging. Create a class that implements \Magento\Framework\Config\DataInterface. This will allow the configuration object to read and process the configuration data from XML files. See Module configuration files in the Adobe Commerce Help Center1. References: https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/files/module-files.html?lang=en1

## QUESTION NO: 8

An Adobe Commerce Architect is troubleshooting an issue on an Adobe Commerce Cloud project that is not yet live.

The developers migrate the Staging Database to Production in readiness to Go Live. However, when the developers test their Product Import feature, the new products do not appear on the frontend.

The developers suspect the Varnish Cache is not being cleared. Staging seems to work as expected. Production was working before the database migration.

What is the likely cause?

**A.** The Fastly credentials in the Production Database are incorrect.

**B.** A deployment should have been done on Production to initialize Fastly caching.

**C.** The site URLs in the Production Database are the URLs of the Staging Instance and must be updated.

**ANSWER: A**

**Explanation:**

The Fastly credentials in the Production Database are incorrect. This means that the Varnish cache cannot be cleared by Commerce when new products are imported. The Fastly credentials should be updated to match the Production environment. See Configure Fastly credentials in the Adobe Commerce Help Center. References: https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/cache/use-varnish-cache.html?lang=en1 https://support.magento.com/hc/en-us/articles/360006008192-Configure-Fastly-credentials

## QUESTION NO: 9

While reviewing a newly developed pull request that refactors multiple custom payment methods, the Architect notices multiple classes that depend on \Magento\Framework\Encryption\EncryptorInterf ace to decrypt credentials for sensitive data. The code that is commonly repeated is as follows:

```
namespace Vendor\PaymentModule\Gateway\Config;

class Config extends \Magento\Payment\Gateway\Config\Config
{
    ...
    public function __construct(
        ...
        ScopeConfigInterface $scopeConfig,
        EncryptorInterface $encryptor,
        ...
    ) {
        parent::__construct($scopeConfig, $methodCode, $pathPattern);
        $this->scopeConfig = $scopeConfig;
        $this->encryptor = $encryptor;
    }

    public function getUserSecret(): string
    {
        return $this->encryptor->decrypt(
            $this->scopeConfig->getValue('payment/method_code/user_secret')
        );
    }
```

In each module, the user_secret config is declared as follows:

```
<field id="user_secret" translate="label" type="obscure" sortOrder="20" showInDefault="1" >
        <label>Secret Key</label>
        <backend_model>Magento\Config\Model\Config\Backend\Encrypted</backend_model>
</field>
```

The Architect needs to recommend an optimal solution to avoid redundant dependency and duplicate code among the methods. Which solution should the Architect recommend?

**A.** Replace all Vendor\PaymentModule\Gateway\Config\Config Classes With virtualTyp- Of Magento\Payxer.t\Gateway\Conflg\Config and Set under ccnfig.xml

**B.** Add a plugin after the getvalue method of $sccpeConfig, remove the $encryptor from dependency and use it in the plugin to decrypt the value if the config name is 'user.secret?

**C.** Create a common config service class vendor\Payment\Gateway\config\Config under Vendor.Payment and use it as a parent class for all of the Vender \EaymentModule\Gateway\Config\Config Classes and remove $sccpeConfig and $encryptor dependencies

**ANSWER: A**

**Explanation:**

To avoid redundant dependency and duplicate code among the methods, the Architect should recommend replacing all Vendor\PaymentModule\Gateway\Config\Config classes with virtualType of Magento\Payment\Gateway\Config\Config and setting under config.xml. This will allow using the core config class that already has the scopeConfig dependency and the logic to get the value from the config. The backend_model attribute will ensure that the user_secret value is encrypted and decrypted automatically by the core EncryptorInterface class. Option B is incorrect because it will add unnecessary complexity and overhead to the scopeConfig object. Option C is incorrect because it will still require creating a custom config service class and injecting the encryptor dependency. References: https://devdocs.magento.com/guides/v2.4/payments-integrations/base-integration/integration-intro.html https://devdocs.magento.com/guides/v2.4/config-guide/prod/config-reference-encryptor.html

## QUESTION NO: 10

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.** Write a plugin for \Magento\Framework\Config\Data::get() and read the custom xml files

**B.** Append the custom xml file name in "Magento\Config\Model\Config\Structure\Reader" in di.xml

**C.** Create a Data class that implements "\Magento\Framework\Config\Data'

**D.** Inject a "reader" dependency for "Magento\Framework\Config\Data" in di.xml

**E.** Make a Reader class that implements "\Magento\Framework\Config\Reader\Filesystem"

**ANSWER: C E**

**Explanation:**

Based on web searches, it seems that Magento uses different classes and interfaces to interact with configuration files, such as Data, Reader, and Converter12.

According to the documentation1, Data is a class that provides access to configuration data using a scope. Reader is an interface that reads configuration data from XML files. Converter is an interface that converts XML data into an array representation.

Based on these definitions, I would say that two possible steps that the Architect should make to meet the requirement are:

These steps would allow the custom module to read all the XML configuration files declared in the system, merge them together, and use their values in PHP class.

The Architect should make two steps to meet this requirement: C) Create a Data class that implements "\Magento\Framework\Config\Data". This class will be responsible for reading and merging the custom xml configuration files and providing access to their values. The Data class should extend \Magento\Framework\Config\Data and use the constructor to inject the Reader class and the cache type. E) Make a Reader class that implements "\Magento\Framework\Config\Reader\Filesystem". This class will be responsible for loading and validating the custom xml configuration files from different modules. The Reader class should extend \Magento\Framework\Config\Reader\Filesystem and use the constructor to specify the file name, schema file, and validation state of the custom xml configuration files. Option A is incorrect because writing a plugin for \Magento\Framework\Config\Data::get() will not define a new xml configuration file, but rather modify the existing one. Option B is incorrect because appending the custom xml file name in "Magento\Config\Model\Config\Structure\Reader" in di.xml will not define a new xml configuration file, but rather add it to the system configuration structure. Option D is incorrect because injecting a "reader" dependency for "Magento\Framework\Config\Data" in di.xml will not define a new xml configuration file, but rather use an existing one. References: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/build/XSD-XML-validation.html