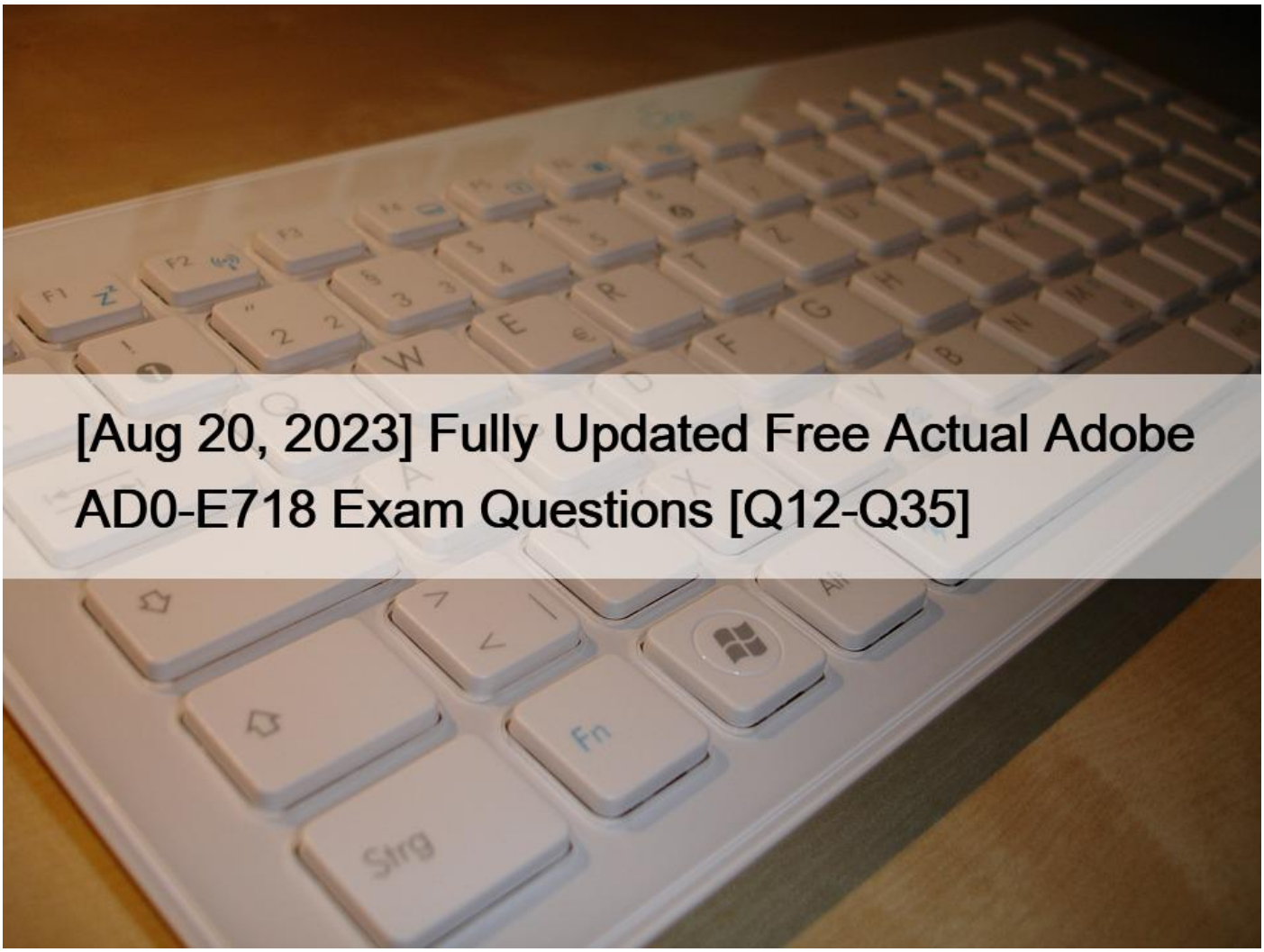
[Aug 20, 2023 Fully Updated Free Actual Adobe AD0-E718 Exam Questions [Q12-Q35



[Aug 20, 2023 Fully Updated Free Actual Adobe AD0-E718 Exam Questions Free AD0-E718 Questions for Adobe AD0-E718 Exam [Aug-2023 QUESTION 12

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?
* Change Session Cookie attribute to “SameSite=None”
* Disable Session Validation for “HTTP_X_FORWARDED_FOR” header
* Set Cookie Domain for both websites to “.example.com”
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.

QUESTION 13

An Adobe Commerce Architect is reviewing api-functional test code. Some tests send errors to indicate that the customer address does not exist.

The test codes show the following:

```
/**
 * @magentoDataFixture Magento/Customer/_files/customer_one_address.php`
 * ...
 */
public function testMyUseCasTestForCartAddress(): void
```

Which steps should the Architect take to fix the test errors?

A)

Update the annotation to specify address_id `@magentoDataFixture Magento/Customer/_files/customer_one_address.php with:`
`{"address_id":"$address.id$"}`

B)

Change the annotation to use `@magentoApiDataFixture Magento/Customer/_files/customer_one_address.php`
instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

C)

Set the annotation to use `@magentoPersistDataFixture Magento/Customer/_files/customer_one_address.php`
instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

* Option A
* Option B
* Option C

The test errors are caused by using the wrong customer ID and address ID in the request. The correct customer ID and address ID should be obtained from the response of the previous request to create a customer and an address. The test code should use $this->customer->getId() and $this->address->getId() instead of hard-coded values. Reference: https://devdocs.magento.com/guides/v2.4/get-started/web-api-functional-testing.html

QUESTION 14

An Architect is investigating a merchant’s Adobe Commerce production environment where all customer session data is

randomly being lost. Customer session data has been configured to be persisted using Redis, as are all caches (except full page cache, which is handled via Varnish).

After an initial review, the Architect is able to replicate the loss of customer session data by flushing the Magento cache storage, either via the Adobe Commerce Admin Panel or running bin/iuagento cache: flush on the command line. Refreshing all the caches in the Adobe Commerce Admin Panel or running bin/magento cache: clean on the command line does not cause session data to be lost.

What should be the next step?
* Educate the merchant to not flush cache storage and only refresh the caches in future.
* Set the ‘Stores > Configuration’ option for “Store Session Data Separately’ to ‘Yes’ in the Adobe Commerce Admin Panel.
* Check app/etc/evn.php and make sure that the Redis configuration for caches and session data use different database numbers.
Checking app/etc/env.php and making sure that the Redis configuration for caches and session data use different database numbers is the next step. This is because using the same database number for both caches and session data can cause session data to be lost when flushing the cache storage. By using different database numbers, the session data can be isolated from the cache data and avoid being overwritten. See Use Redis for session storage in the Adobe Commerce Help Center1. Reference: https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/cache/redis/redis-session.html?lang=en1

QUESTION 15

An Adobe Commerce Architect designs a data flow that contains a new product type with its own custom pricing logic to meet a merchant requirement.

Which three developments are valid when reviewing the implementation? (Choose three.)
* Content of the etc/product_types.xml file
* Hydrator for attributes belonging to the new product type
* Custom type model extended from the abstract Product Type model
* A new class with custom pricing logic, extending the abstract Product model class
* Data patch to register the new product type
* New price model extending MagentoCatalogModelProductTypePrice
Explanation

According to some tutorials45, creating a custom product type in Adobe Commerce involves several steps, such as:

* Creating a product_types.xml file in etc folder to declare the new product type

* Creating a custom type model that extends from an abstract product type model

* Creating a custom price model that extends from an abstract price model

* Creating a layout file for the new product type

* Creating a data patch to register the new product type

Based on these steps, I would say that three possible developments that are valid when reviewing the implementation are:

* A. Content of the etc/product_types.xml file

* C. Custom type model extended from the abstract Product Type model

* F. New price model extending MagentoCatalogModelProductTypePrice These developments would allow creating a new product type with its own custom pricing logic and attributes.

QUESTION 16

An Adobe Commerce Architect needs to customize the workflow of a monthly installments payment extension. The extension is from a partner that is contracted with the default website PSR which has its own legacy extension (a module using deprecated payment method).

The installment payment partner manages only initializing a payment, and then hands the capture to be executed by the PSP. Once the amount is successfully captured, the PSP notifies the website through an IPN. The goal of the IPN is only to create an “invoice” and save the ‘capture information’ to be used later for refund requests through the PSP itself.

The Architect needs the most simple solution to capture the requested behavior without side effects.

Which solution should the Architect implement?
* Add a plugin before the $invoice-> () and changes its input to prevent the call of the $payment-> capture()
* Change the can_ capture attribute for the payment method under config.xml to be <can_capture>0</can_capture>
* Declare a capture command with type MagentopaymentGatewayCommandNullCommand for the payment method CommandPool in di.zm1

The best solution for the Adobe Commerce Architect to implement in order to capture the requested behavior without side effects is to declare a capture command with type MagentopaymentGatewayCommandNullCommand for the payment method CommandPool in di.xml. This will allow the partner to initialize the payment and then hand the capture over to the PSP, while also preventing the website from calling the $payment->capture() method. It will also allow the PSP to notify the website through an IPN, which will create an “invoice” and save the ‘capture information’ to be used later for refund requests through the PSP itself.

The Architect should implement the solution of declaring a capture command with type MagentoPaymentGatewayCommandNullCommand for the payment method CommandPool in di.xml. This command will do nothing when the capture method is called on the payment method, which is the desired behavior since the capture is handled by the PSP. The NullCommand class implements MagentoPaymentGatewayCommandInterface and overrides the execute() method to return null. Option A is incorrect because adding a plugin before the $invoice->capture() method and changing its input will not prevent the call of the $payment->capture() method, but rather change the invoice object that is passed to it. Option B is incorrect because changing the can_capture attribute for the payment method under config.xml to be <can_capture>0</can_capture> will not prevent the capture method from being called, but rather disable the capture option in the Admin panel. Reference: https://devdocs.magento.com/guides/v2.4/payments-integrations/base-integration/facade-configuration.html

QUESTION 17

An Architect is configuring the preload.keys for Redis on an Adobe Commerce on-premise instance.

The Architect discovers that the following cache keys are loaded on each frontend request: eav_entity_types, GLOBAL_PLUGIN_LIST, DB_IS_UP_TO_DATE , SYSTEM_DEFAULT.

* The id_prefix of the frontend =>page_cache is set to 061_.

* The id_prefix of frontend => default: is not set.

* The Architect has enabled and configured Redis L2 caching.

How should the preload.keys be configured?

*
```
'preload_keys' => [
    '061_EAV_ENTITY_TYPES:hash',
    '061_GLOBAL_PLUGIN_LIST:hash',
    '061_DB_IS_UP_TO_DATE:hash',
    '061_SYSTEM_DEFAULT:hash',
],
```

*
```
'preload_keys' => [
    'EAV_ENTITY_TYPES:hash',
    'GLOBAL_PLUGIN_LIST:hash',
    'DB_IS_UP_TO_DATE:hash',
    'SYSTEM_DEFAULT:hash',
],
```

*
```
'preload_keys' => [
    'EAV_ENTITY_TYPES',
    'GLOBAL_PLUGIN_LIST',
    'DB_IS_UP_TO_DATE',
    'SYSTEM_DEFAULT',
],
```

*
```
'preload_keys' => [
    '061_EAV_ENTITY_TYPES',
    '061_GLOBAL_PLUGIN_LIST',
    '061_DB_IS_UP_TO_DATE',
    '061_SYSTEM_DEFAULT',
],
```

QUESTION 18

An Adobe Commerce Architect is working on a sales campaign to present a new product on the site that allows the purchase of a pre-defined set of products with a discount. Each product in the set should have a separate stock and tax class.

One requirement is to use a third-party system to build reports with REST API to fetch the following data:

* SKU

* Qty

* Original price

* Sales price

* Tax amount

Which solution should the Architect use to meet these requirements?
* * Create Fixed Bundle Product for gathering simple products;

* Manage price for every selected option;

* Add extension attribute original_simple_price for MagentoSalesApiDataOrderItemExtensionInterface and populate value with price of simple product;
* * Create Dynamic Bundle Product for gathering simple products;

* Utilize Content Staging to manage special prices for bundle products on time for the campaign;

* Expose required data via Adobe Commerce Order API;
* * Create Grouped Product and Create after plugin on MagentoGroupedProductModelProductTypeGrouped:preparedForCarrAdvanced for bunch products ordering;

* Utilize Content Staging to manage special prices on time for the campaign for simple products;

* Expose required data via Adobe Commerce Order API;
To meet the requirements, the Architect should use the following solution: * Create Fixed Bundle Product for gathering simple products. This will allow the purchase of a pre-defined set of products with a discount and separate stock and tax class for each product. * Manage price for every selected option. This will allow setting the original and sales price for each product in the bundle. * Add extension attribute original_simple_price for MagentoSalesApiDataOrderItemExtensionInterface and populate value with price of simple product. This will allow exposing the original price of the simple product via REST API along with other required data. Option B is incorrect because Dynamic Bundle Product will not allow setting a pre-defined set of products with a discount. Option C is incorrect because Grouped Product will not allow setting a discount for the whole set of products. Reference: https://docs.magento.com/user-guide/catalog/product-create-bundle.html
https://devdocs.magento.com/guides/v2.4/extension-dev-guide/extension_attributes/adding-attributes.html

QUESTION 19

While reviewing a newly developed pull request that refactors multiple custom payment methods, the Architect notices multiple classes that depend on MagentoFrameworkEncryptionEncryptorInterf ace to decrypt credentials for sensitive dat a. 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?
* Replace all VendorPaymentModuleGatewayConfigConfig Classes With virtualTyp- Of MagentoPayxer.tGatewayConflgConfig and Set <user_secret backend_Model=”MagentoConfigModelConfigBackendEncrypted” /> under ccnfig.xml
* 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?
* Create a common config service class vendorPaymentGatewayconfigConfig under Vendor.Payment and use it as a parent class for all of the Vender EaymentModuleGatewayConfigConfig Classes and remove $sccpeConfig and $encryptor dependencies
To avoid redundant dependency and duplicate code among the methods, the Architect should recommend replacing all VendorPaymentModuleGatewayConfigConfig classes with virtualType of MagentoPaymentGatewayConfigConfig and setting <user_secret backend_model=”MagentoConfigModelConfigBackendEncrypted” /> 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.
Reference: 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 20

A company has an Adobe Commerce store. An attribute named “my.attribute” (type “text”) is created to save each product’s global ID that is shared between multiple systems.

Several months after going live, the values of “my.attribute” are all integer. This causes a problem for the other systems when those systems receive this data.

An Adobe Commerce Architect needs to recommend a solution to change the type of “my.attribute” from textXo int Which two steps should the Architect take to achieve this? (Choose two.)
* Migrate data from table “catalog_product_entity_text” to “catalog.producLentityjnt” for the attribute.id
* Go to Admin > Stores > Attributes > Product, edit “my.attribute” and update type from “text’ to “inf
* Write a plugin for MagentoEavModelEntityAttrlbuteBackendAbstractBackend::afterLoad() and load data from “catalog_product_entity_int”
* Create a Data Patch and update ‘my.attribute’ type from “text” to “inf
* Run the Command bin/magentc indexer: reset catalog_product_attribute
Explanation

Option A is correct because it will migrate data from one table to another based on the attribute id, which is required when changing the attribute type . Option D is correct because it will create a data patch that will update ‘my.attribute’ type from “text” to “int”, which is a recommended way of changing the attribute type programmatically4.

QUESTION 21

An Adobe Commerce Architect notices that queue consumers close TCP connections too often on Adobe Commerce Cloud server leading to delays in processing messages.

The Architect needs to make sure that consumers do not terminate after processing available messages in the queue when CRON job is running these consumers.

How should the Architect meet this requirement?
* Increase multiple_process limit to spawn more processes for each consumer.
* Set CONSUMER_WAIT_FOR_MAX_MESSAGES variable true in deployment stage.
* Change max_messages from 10,000 to 1,000 for CRON_CONSUMER_RUNNERvariable.

The best way to meet this requirement is to set the CONSUMERWAITFORMAXMESSAGES variable to true in the deployment stage. This variable will ensure that the consumer will not terminate when there are no more messages in the queue and will instead wait until a new message is available, preventing it from closing the connection prematurely. Additionally, the multiple_processes limit can be increased to spawn more processes for each consumer, which will help ensure that messages can be processed faster.

Setting CONSUMER_WAIT_FOR_MAX_MESSAGES variable true in deployment stage will make sure that consumers do not terminate after processing available messages in the queue when CRON job is running these consumers. This will prevent consumers from closing TCP connections too often and improve performance. See Start message queue consumers in the Adobe Commerce Help Center1. Reference:
https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/message-queues/consumers.html?lang=en2
https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/cli/start-message-queues.html1

QUESTION 22

An Architect needs to integrate an Adobe Commerce store with a new Shipping Carrier. Cart data is sent to the Shipping Carrier’s API to retrieve the price and display to the customer. After the feature is implemented on the store, the API hits its quota and returns the error “Too many requests”. The Shipping Carrier warns the store about sending too many requests with the same content to the API.

In the carrier model, what should the Architect change to fix the problem?
* Implement _setCachedQuotes () and_getCachedQuotes() return the data if the request matches.
* In _doShipmentRequest (), call canCollectRates() before sending request to the API
* Override getResponse (), save the response to a variable, check if the response exists, then return.

Explanation

Implementing setCachedQuotes () andgetCachedQuotes() in the carrier model can allow the store to store the cart data in a cache, so that repeated requests with the same content can be retrieved from the cache instead of sending a new request to the API. This can reduce the number of requests and avoid hitting the quota limit.

References:

[1] https://docs.adobe.com/content/help/en/experience-manager-65/commerce/commerce-payment-shipping-mod

QUESTION 23

An Adobe Commerce Architect needs to ensure zero downtime during the deployment process of Adobe Commerce on-premises. Which two steps should the Architect follow? (Choose two.)
* Run bin/magento setup:upgrade –keep-generated to Upgrade database

* Run bin/magento setup:upgrade -dry-run=true to upgrade database
* Rim bin/magento setup:upgrade –convert-old-scripts=true to Upgrade database
* Enable config flag under developer/zere _down_time/enabled
* Enable config flag under deployment/blue_ green/enabled

To ensure zero downtime during the deployment process of Magento 2 on-premises, the Architect should follow two steps:

Run bin/magento setup:upgrade –keep-generated to upgrade database. This will skip the regeneration of static content and code files during the upgrade process, which can take a long time and cause downtime. The static content and code files should be generated separately before or after the upgrade process.

Enable config flag under deployment/blue_green/enabled. This will enable the blue-green deployment strategy, which creates a copy of the current production environment (blue) and deploys the new code to it (green). Then, it switches the traffic from the blue environment to the green environment without any downtime. This option can be enabled by adding a line like deployment/blue_green/enabled: true to the .magento.env.yaml file.

QUESTION 24

An Adobe Commerce Architect is working on a sales campaign to present a new product on the site that allows the purchase of a pre-defined set of products with a discount. Each product in the set should have a separate stock and tax class.

One requirement is to use a third-party system to build reports with REST API to fetch the following data:

* SKU

* Qty

* Original price

* Sales price

* Tax amount

Which solution should the Architect use to meet these requirements?
* * Create Fixed Bundle Product for gathering simple products;

* Manage price for every selected option;

* Add extension attribute original_simple_price for

MagentoSalesApiDataOrderItemExtensionInterface and populate value with price of simple product;
* * Create Dynamic Bundle Product for gathering simple products;

* Utilize Content Staging to manage special prices for bundle products on time for the campaign;

* Expose required data via Adobe Commerce Order API;
* * Create Grouped Product and Create after plugin on

MagentoGroupedProductModelProductTypeGrouped:preparedForCarrAdvanced for bunch products ordering;

* Utilize Content Staging to manage special prices on time for the campaign for simple products;

* Expose required data via Adobe Commerce Order API;
Explanation

A bundle product is a customizable product that consists of several options, each based on a simple or virtual product. A grouped product is a collection of simple products that are presented as a group.

According to some tutorials , creating a bundle product in Adobe Commerce involves several steps, such as:

* Choosing the bundle product template and attribute set

* Completing the required settings, such as name, SKU, price, and weight

* Configuring the basic settings, such as status, visibility, and categories

* Adding the bundle options and associated products

* Adding optional product information, such as images and meta data

* Posting the product

Content staging is a feature that allows creating, previewing, and scheduling content updates for your store directly from the Admin . You can use content staging to create campaigns that include changes to products, categories, pages, blocks, widgets, price rules, and more.

Based on these steps and features, I would say that one possible solution that the Architect should use to meet these requirements is:

* B. Create Dynamic Bundle Product for gathering simple products; Utilize Content Staging to manage special prices for bundle products on time for the campaign; Expose required data via Adobe Commerce Order API; This solution would allow creating a new product that allows the purchase of a pre-defined set of products with a discount. Each product in the set would have a separate stock and tax class. The special prices for bundle products could be managed using content staging. The required data could be exposed via Adobe Commerce Order API.

QUESTION 25

An Architect needs to review a custom product feed export module that a developer created for a merchant.

During final testing before the solution is deployed, the product feed output is verified as correct. All unit and integration tests for code pass.

However, once the solution is deployed to production, the product price values in the feed are incorrect for several products. The products with incorrect data are all currently part of a content staging campaign where their prices have been reduced.

What did the developer do incorrectly that caused the feed output to be incorrect for products in the content staging campaign?
* The developer forgot to use the getContentStagingValue() method to retrieve the active campaign value of the product data
* The developer retrieved product data directly from the database using the entity_id column rather than a collection or repository.
* The developer did not check for an active content staging campaign and emulates the campaign state when retrieving product data.
Explanation

According to the Adobe Commerce documentation, when retrieving product data, developers must use the getContentStagingValue() method to ensure that the active campaign value for theproduct is retrieved. This method takes into account any content staging campaigns that might be active, and returns the appropriate value from the content staging campaign rather than the default value from the database. Failing to use this method can result in incorrect data being returned in the product feed.

QUESTION 26

An Adobe Commerce store owner sets up a custom customer attribute “my.attribute” (type int).

An Architect needs to display customer-specific content on the home page to Customers with “my.attribute” greater than 3. The website is running Full Page Cache.

Using best practices, which two steps should the Architect take to implement these requirements? (Choose two.)
* Use customer-data JS library to retrieve “my.attribute” value
* Add a new context value of “my.attribute” to MagentoFrameworkAppHttpContext
* Add a custom block and a phtml template with the content to the cmsjndexjndex.xml layout
* Create a Customer Segment and use “my.attribute” in the conditions
* Add a dynamic block with the content to the Home Page

Explanation

https://docs.magento.com/user-guide/v2.3/stores/attributes-customer.html displaying custom customer attributes on cached pages using best practices involves several steps, such as:

* Creating a custom block and a phtml template with the content to display

* Adding the custom block to the layout file of the page where it should appear

* Creating a section.xml file to declare a new section for the custom attribute

* Creating a plugin for MagentoCustomerCustomerDataSectionPoolInterface to add the custom attribute value to the section data

* Using customer-data JS library to retrieve and display the custom attribute value in the phtml template

QUESTION 27

An Architect agrees to improve company coding standards and discourage using Helper classes in the code by introducing a new check with PHPCS.

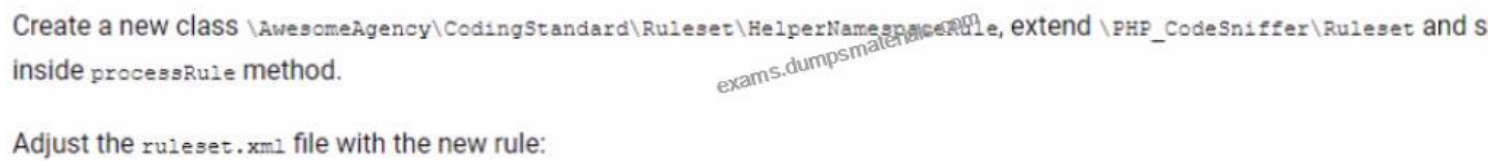
The Architect creates the following:

* A new composer package under the AwesomeAgencyCodingStandard namespace

* The ruleset. xml file extending the Magento 2 Coding Standard

What should the Architect do to implement the new code rule?

*



*
```
<rule ref="Magento2.Namespaces.ForbiddenNamespaces">
    <include-pattern>AwesomeAgency\*\Helper\*</include-pattern>
</rule>
```

* Implement `\PHP_CodeSniffer\Sniffs\Sniff` under your `\AwesomeAgency\CodingStandard\Sniff\HelperNamespaceSniff`. Prov implementation in `process` method.

Option B is the correct way to implement the new code rule. The Architect should create a new class that extends the PHP_CodeSnifferSniffsSniff abstract class and implements the register() and process() methods. The register() method should return an array of tokens that the rule applies to, such as T_CLASS. The process() method should check if the class name contains Helper and add a warning or an error if it does. The Architect should also reference the new class in the ruleset.xml file using the <rule ref> tag. Reference: https://devdocs.magento.com/guides/v2.4/coding-standards/technical-guidelines.html#14-code-style https://github.com/squizlabs/PHP_CodeSniffer/wiki/Coding-Standard-Tutorial

QUESTION 28

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?
* 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.
* Use OAuth-based authentication to provide access to system resources. Integration will be registered by the merchant in the panel an OAuth handshake during activation. The third-party system should follow OAuth protocol to authorize.
* 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.
To create an application that will integrate the Adobe Commerce system to get orders data for reporting using the get /v1/orders endpoint, you should use OAuth-based authentication to provide access to system resources. OAuth is a token-passing mechanism that allows a system to control which third-party applications have access to internal data without revealing or storing any user IDs or passwords. The integration will be registered by the merchant in the admin panel and will perform an OAuth handshake during activation. The third-party system should follow OAuth protocol to authorize. The merchant will have the ability to restrict or extend access to resources as well as to revoke the access using Admin Panel. Reference: 1

1: https://devdocs.magento.com/guides/v2.3/get-started/authentication/gs-authentication-oauth.html

QUESTION 29

An external system integrates functionality of a product catalog search using Adobe Commerce GraphQL API. The Architect creates a new attribute my_attribute in the admin panel with frontend type select.

Later, the Architect sees that Productinterface already has the field my_atcribute, but returns an mc value. The Architect wants this field to be a new type that contains both option id and label.

To meet this requirement, an Adobe Commerce Architect creates a new module and file etc/schema.graphqls that declares as

follows:

```
interface ProductInterface {
    my_attribute: SelectableOption @resolver(class:"Vendor\\CatalogGraphQl\\Model\\Resolver\\SelectableOption")
}
```

After calling command setup:upgrade, the introspection of Productlnterface field xy_attribute remains int. What prevented the value type of field my_attribute from changing?
* The fields of Productlnterface are checked during processing schema.graphqls files. If they have a corresponding attribute, then the backendjype of product attribute is set for field type.
* The interface Productlnterface is already declared in Magento.CatalogGraphQI module. Extending requires use of the keyword -xceni before a new declaration of Productlnterface.
* The Magento.CatalogGraphQI module occurs later in sequence than the Magento.GraphQI module and merging output of dynamic attributes schema reader overrides types declared in schema.graphqls
products query is a GraphQL query that returns information about products that match specified search criteria. It also shows how to use ProductInterface fields to retrieve product data.

https://devdocs.magento.com/guides/v2.3/graphql/queries/products.html

The Magento.CatalogGraphQI module occurs later in sequence than the Magento.GraphQI module and merging output of dynamic attributes schema reader overrides types declared in schema.graphqls. The dynamic attributes schema reader is responsible for adding product attributes to the Productinterface based on the backend type of the attribute. Since the attribute my_attribute has a backend type of int, the field my_attribute in the Productinterface will also have a type of int, regardless of the custom type declared in the schema.graphqls file. To avoid this, the Architect should either change the backend type of the attribute to match the custom type, or use a different name for the field that does not conflict with the attribute name. Reference:
https://devdocs.magento.com/guides/v2.4/graphql/develop/create-graphqls-file.html

QUESTION 30

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 deployment should have been done on Production to initialize Fastly caching.
* The site URLs in the Production Database are the URLs of the Staging Instance and must be updated.
* The Fastly credentials in the Production Database are incorrect.
Explanation

The likely cause of the issue is that a deployment should have been done on Production to initialize Fastly caching. This is because when the database is migrated from Staging to Production, any changes made to the Staging Database will not be reflected in the Production environment until a deployment is made. This includes any changes made to the Varnish Cache, which needs to be cleared in order for the new products to appear on the frontend.

QUESTION 31

An Adobe Commerce Architect gets a request to change existing payment gateway functionality by allowing voided transactions only for a certain range of paid amounts.

In the vendor module file etc/config.xml,payment method has an option can,_voidsetto 1.

How should this customization be done?
* Extend MagentoPaymentModelMethodAdapter and reimplement method void. Use this new class as a new type of payment method facade configuration overriding virtualType type for adapter.
* Declare a new plugin for class MagentoPayment GatewayConfigConfigValueHandler and using the afterHandle method, change the result for Subject can_void.
* Add new handler with name can_void to virtualType based on typeMagento paymentGatewayconfigValueHandlerPool In payment method facade configuration.
Explanation

payment facade is an instance of Payment Adapter configured with virtual types and allows to process payment actions between Magento Sales Management and payment processor. It also says that you can add dependency injection (DI) configuration for payment method facade in your %Vendor_Module%/etc/di.xml file.

https://devdocs.magento.com/guides/v2.3/payments-integrations/base-integration/facade-configuration.html

QUESTION 32

An external system integrates functionality of a product catalog search using Adobe Commerce GraphQL API.

The Architect creates a new attribute my_attribute in the admin panel with frontend type select.

Later, the Architect sees that Productinterface already has the field my_atcribute, but returns an mc value. The Architect wants this field to be a new type that contains both option id and label.

To meet this requirement, an Adobe Commerce Architect creates a new module and file etc/schema.graphqls that declares as follows:

```
interface ProductInterface {
    my_attribute: SelectableOption @resolver(class:"Vendor\\CatalogGraphQl\\Model\\Resolver\\SelectableOption")
}
```

After calling command setup:upgrade, the introspection of ProductInterface field xy_attribute remains int.

What prevented the value type of field my_attribute from changing?
* The fields of ProductInterface are checked during processing schema.graphqls files. If they have a corresponding attribute, then the backendjype of product attribute is set for field type.
* The interface ProductInterface is already declared in Magento.CatalogGraphQI module. Extending requires use of the keyword -xceni before a new declaration of ProductInterface.
* The Magento.CatalogGraphQI module occurs later in sequence than the Magento.GraphQI module and merging output of dynamic attributes schema reader overrides types declared in schema.graphqls
Explanation

products query is a GraphQL query that returns information about products that match specified search criteria. It also shows how to use ProductInterface fields to retrieve product data.

https://devdocs.magento.com/guides/v2.3/graphql/queries/products.html

QUESTION 33

An Adobe Commerce Architect is reviewing api-functional test code. Some tests send errors to indicate that the customer address does not exist.

The test codes show the following:

```
/**
 * @magentoDataFixture Magento/Customer/_files/customer_one_address.php`
 * ...
 */
public function testMyUseCasTestForCartAddress(): void
```

Which steps should the Architect take to fix the test errors?

* Update the annotation to specify address_id `@magentoDataFixture Magento/Customer/_files/customer_one_address.php with: {"address_id":"$address.id$"}`

* Change the annotation to use `@magentoApiDataFixture Magento/Customer/_files/customer_one_address.php` instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

* Set the annotation to use `@magentoPersistDataFixture Magento/Customer/_files/customer_one_address.php` instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

The test errors are caused by using the wrong customer ID and address ID in the request. The correct customer ID and address ID should be obtained from the response of the previous request to create a customer and an address. The test code should use $this->customer->getId() and $this->address->getId() instead of hard-coded values. Reference: https://devdocs.magento.com/guides/v2.4/get-started/web-api-functional-testing.html

QUESTION 34

An Adobe Commerce Architect is supporting deployment and building tools for on-premises Adobe Commerce projects. The tool is executing build scripts on a centralized server and using an SSH connection to deploy to project servers.

A client reports that users cannot work with Admin Panel because the site breaks every time they change interface locale.

Considering maintainability, which solution should the Architect implement?

* Edit project env.php file, configure ‘admin_locales_for.build’ value, and specify all required locales
* Adjust the tool’s build script and specify required locales during ‘setup:static-content:deploy’ command
* Modify project config.php file, configure ‘admin_locales_for_deploy’ value, and specify all required locales

Explanation

To ensure that the Admin Panel works correctly and is maintainable, the Architect should adjust the tool’s build script so that it specifies all of the required locales when executing the ‘setup:static-content:deploy’ command. This will ensure that the project is correctly configured for all supported locales and will also make sure that the build script does not need to be modified each time a new locale is added.

QUESTION 35

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?
* 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.
* Use OAuth-based authentication to provide access to system resources. Integration will be registered by the merchant in the panel an OAuth handshake during activation. The third-party system should follow OAuth protocol to authorize.
* 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.
Explanation

According to the documentation1, token-based authentication is a simple way to access resources using an access token that is generated when an integration is activated. OAuth-based authentication is a more secure way to access resources using a consumer key, consumer secret, access token, and access token secret that are generated when an integration is registered and authorized.

Based on these definitions, I would say that the type of authentication that should be used and implemented in a third-party system for this integration is:

* B. Use OAuth-based authentication to provide access to system resources. Integration will be registered

* by the merchant in the panel an OAuth handshake during activation. The third-party system should follow OAuth protocol to authorize.