DUMPSBOSS.

IBM App Connect Enterprise V11 Solution Development

IBM C1000-056

Version Demo

Total Demo Questions: 9

Total Premium Questions: 92

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Solution Design	4
Topic 2, Application Development	48
Topic 3, Application Configuration	8
Topic 4, Application Testing	2
Topic 5, Administration	17
Topic 6, Security	11
Topic 7, Problem Determination	2
Total	92

QUESTION NO: 1

Which two external security providers can be configured in the security profiles in IBM App

Connect Enterprise?

A. RADIUS

B. LDAP

C. OAuth 2.0

D. Linux PAM

E. WS-Trustv1.3STS

ANSWER: B C E

Explanation:

IBM App Connect Enterprise security profiles can reference an LDAP server for directory-based authentication and authorization functions, an OAuth 2.0 authorization server for OAuth token-based security, and a WS-Trust v1.3 Security Token Service for obtaining or validating security tokens in supported message-security scenarios. These are external providers because App Connect Enterprise delegates the relevant identity, token, or authorization operation to a separately configured service rather than storing and managing those security artifacts in the message flow itself.

Consequently, the question is not accurately limited to two answers: all three of these listed provider types are supported in the applicable security-profile and security-configuration capabilities. LDAP is commonly used where enterprise user identities and groups are held in a directory. OAuth 2.0 supports protected APIs and token flows, while a WS-Trust v1.3 STS supports SOAP-oriented token exchange and trust scenarios. IBM's security-profile documentation describes configuring security behavior and its use of external security services; its OAuth documentation covers the authorization-server integration. See IBM App Connect Enterprise security profiles and IBM App Connect Enterprise OAuth 2.0 security.

QUESTION NO: 2

Which two nodes can be used together to implement a SOAP web service in a message flow?

A. SOAPInput

B. SOAPReply

C. SOAPRequest

D. HTTPInput

E. MQReply

ANSWER: A B

Explanation:

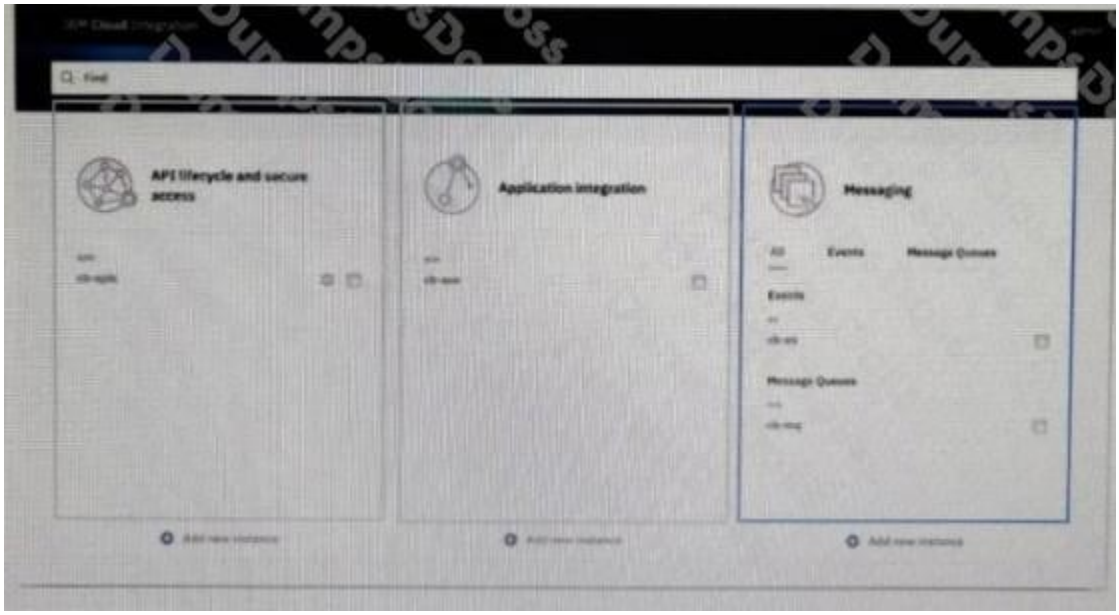
SOAPInput and **SOAPReply** can be used together to expose a SOAP web service. The SOAPInput node receives a SOAP request from a client and initiates processing in the message flow. The SOAPReply node sends the SOAP response back to that client when processing is complete.

A SOAPRequest node is used when a flow acts as a SOAP client and invokes an external SOAP web service. It is not used to receive an inbound service request. See the IBM App Connect Enterprise Version 11 documentation for SOAP node configuration details.

QUESTION NO: 3 - (HOTSPOT)

HOTSPOT

Which section in the Cloud Integration Platform Navigator should be clicked in order to install IBM MQ?



ANSWER:

Explanation:

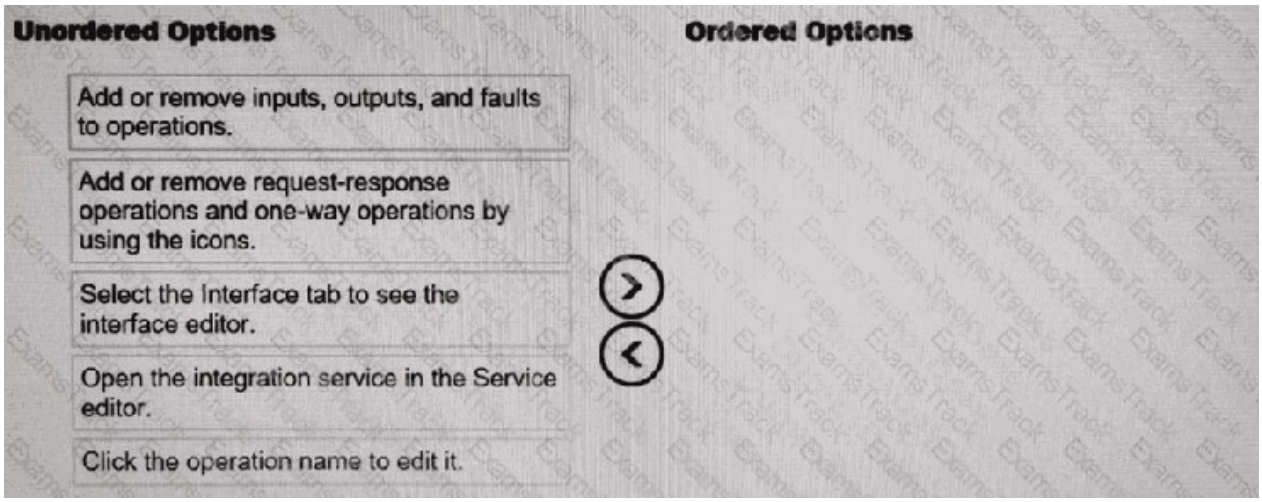
IBM MQ belongs in the **Messaging** section of the Cloud Integration Platform Navigator because its core purpose is to provide reliable, asynchronous message delivery between applications. IBM MQ manages messages through queues, supports applications that send and receive data independently, and helps ensure that messages are retained until they can be processed. In the Navigator, the Messaging tile groups the messaging-related capabilities, including IBM MQ, Message Queues, and event-oriented functions. Selecting this section is therefore the correct starting point for installing IBM MQ.

The Navigator is organized by integration function so that products are installed from the area that matches their role in the integration platform. IBM MQ is not merely an API-management or application-flow capability; it is the foundational messaging component used to connect distributed systems using queues and dependable message transport. Its placement under Messaging reflects that role. Once the Messaging section is selected, the user can access the product instance installation workflow for IBM MQ and provide the configuration required for a deployment. IBM describes IBM MQ as

software that enables applications to communicate and exchange data reliably using messaging and queues. See the IBM MQ product page and the IBM Cloud Pak for Integration documentation for product and platform documentation.

QUESTION NO: 4 - (SIMULATION)

What are the correct sequence of steps to define the operations in a service interface?



ANSWER: See the explanation for the answer

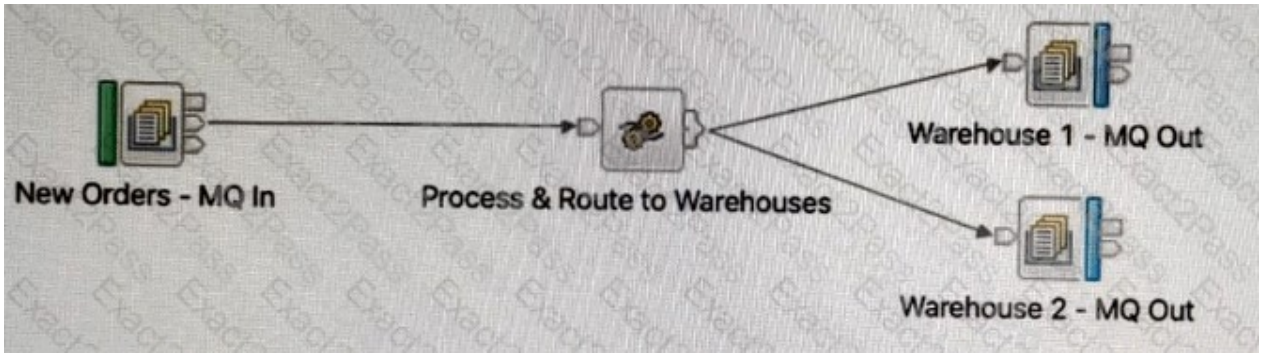
Explanation:

Correct sequence:

This sequence first opens the service and its interface editor, then selects/edits the default operation before defining additional operations and their input, output, and fault details.

QUESTION NO: 5

The following flow receives order messages on an MQInput node then uses a Compute node to do some processing and sends MQ messages to two warehouses on two different queues.



The compute node routes the message to two MQOutput nodes connected through terminals Out1 and Out using the following ESQL expressions.

PROPAGATE TO TERMINAL ' out1; RETURN TRUE;

After deploying the flow, the solution developer noticed that the second queue always receives an empty message. To fix this issue, which ESQL expression lines should be used in the Compute node instead?

A. PROPAGATE TO TERMINAL ' out; PROPAGATE TO TERMINAL ' out ' ;

B. PROPAGATE TO TERMINAL 'out1' DELETE NONE; RETURN TRUE;

C. RETURN TRUE DELETE NONE; RETURN TRUE;

D. PROPAGATE TO TERMINAL ' out1; PROPAGATE; RETURN FALSE;

ANSWER: B

Explanation:

`PROPAGATE TO TERMINAL 'out1' DELETE NONE; RETURN TRUE;` is correct because a Compute node can explicitly propagate its current message assembly to a named output terminal and then allow normal processing to continue. The important part is `DELETE NONE`. By default, a `PROPAGATE` statement deletes the message tree after it has propagated the message. Therefore, when the node subsequently performs its normal automatic propagation caused by `RETURN TRUE`, the message tree would no longer be present. Specifying `DELETE NONE` retains the message assembly after the explicit propagation to `out1`. The explicit propagation sends the populated order message to the first warehouse, and `RETURN TRUE` then causes the same populated message to be propagated through the Compute node's default `Out` terminal to the second warehouse. This is the standard fan-out pattern when a Compute node must send one message to multiple downstream paths while preserving its content. IBM documents both the default deletion behavior of `PROPAGATE` and the use of the `DELETE NONE` clause in the PROPAGATE statement reference. See also IBM's Compute node documentation for its ESQL processing behavior.

QUESTION NO: 6

When referencing a policy within a node configuration, what is the format of the reference?

A. {PolicyProjectName}:PolicyName

B. PolicyProjectName:PolicyName

C. Policy:{PolicyName}

D. ApplicationName:PolicyName

ANSWER: A

Explanation:

`{PolicyProjectName}:PolicyName` is the required policy-reference format when a message flow node configuration refers to a policy in IBM App Connect Enterprise. The portion enclosed in braces identifies the policy project that contains the policy, and the name following the colon identifies the individual policy within that project. For example, a policy named `MyFTPPolicy` in a policy project named `IntegrationPolicies` is referenced as `{IntegrationPolicies}:MyFTPPolicy`.

This qualified syntax lets App Connect Enterprise resolve the policy resource correctly when the application or integration service is deployed. Policies are deployable configuration artifacts that can externalize connection and runtime settings from message flows. A node property that supports a policy reference therefore uses the project-qualified reference rather than an application-qualified resource name. The braces are significant because they delimit the policy project name as part of the reference syntax.

IBM documents policy projects as the containers for policy definitions and describes configuring message flow nodes to use policies through policy references. See the IBM App Connect Enterprise policy projects documentation and the IBM App Connect Enterprise 11.0 documentation.

QUESTION NO: 7

Which file needs to exist in the directory where the command docker build run?

A. Dockerfile

B. docker.yaml

C. Configfile

D. config.yaml

ANSWER: A

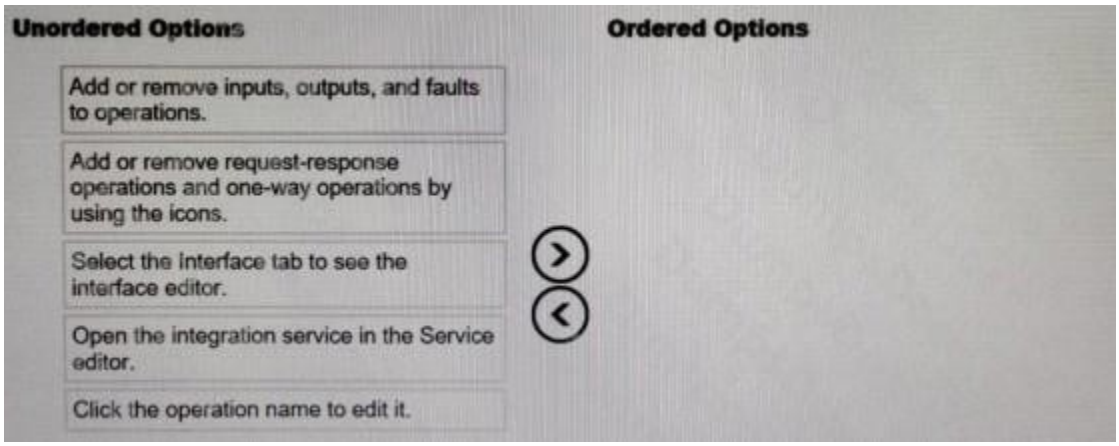
Explanation:

The correct answer is **Dockerfile**. When `docker build` is run without the `-f` or `--file` option, Docker uses the supplied build context (commonly the current directory, represented by `.`) and looks there for a file named exactly `Dockerfile`. This file contains the ordered build instructions Docker follows to create an image, such as selecting a base image with `FROM`, copying application artifacts with `COPY`, installing dependencies with `RUN`, and specifying the startup command with `CMD` or `ENTRYPOINT`. For example, running `docker build -t my-image .` sends the current directory as the build context and, by default, reads `./Dockerfile`. Docker supports a differently named or differently located build-definition file only when it is explicitly identified, for example: `docker build -f path/to/custom-file -t my-image .`. Therefore, for the normal command usage described, a file named `Dockerfile` must be available in the directory used as the build context. Docker's build documentation describes the default Dockerfile lookup behavior and the `-f` override: Docker build file option. The Dockerfile reference also defines the instructions used in that file: Dockerfile reference.

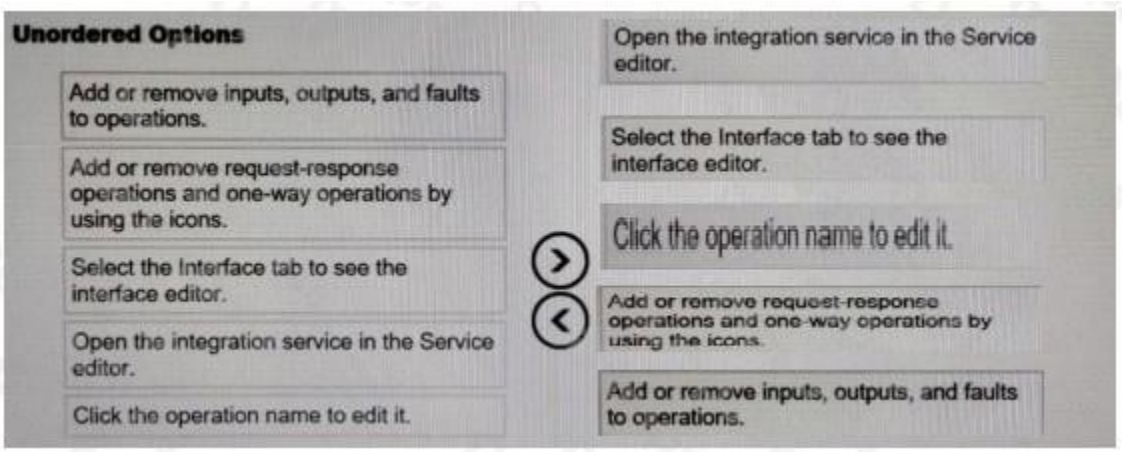
QUESTION NO: 8 - (DRAG DROP)

DRAG DROP

What are the correct sequence of steps to define the operations in a service interface?



ANSWER:



Explanation:

Defining operations in a service interface follows the natural editing flow of the Service editor. First, open the integration service in the Service editor, because the service description is the artifact that owns the interface definition. Next, select the Interface tab, which exposes the interface editor and the operations that the service makes available. A newly created integration service includes a default request-response operation, so clicking that operation name is the appropriate next step when defining or editing its details.

Once the existing operation is available for editing, use the interface editor controls to add or remove request-response and one-way operations. This establishes the set of callable interactions that the service supports. After the required operations have been established, add or remove the inputs, outputs, and faults for each operation. These elements define the messages accepted by the operation, the response returned for request-response interactions, and the fault information that can be produced. Their names and types can then be refined in the Properties view, including creation of complex types where needed.

This sequence keeps the work properly scoped: open the service, navigate to its interface, select the default operation, shape the operation set, and finally define each operation's message contract. IBM documentation for service development and service interfaces is available through the IBM App Connect Enterprise integration service development documentation and the Service Description editor documentation.

QUESTION NO: 9

Which two ESQL statements can be used to define reusable routines?

A. CREATE FUNCTION

B. CREATE PROCEDURE

C. SET

D. PROPAGATE

E. DECLARE

ANSWER: A B

Explanation:

CREATE FUNCTION and **CREATE PROCEDURE** are used to define reusable routines in ESQL. A function returns a value and can be used in an expression. A procedure performs a sequence of actions and can use input, output, and input-output parameters.

Functions and procedures can be defined in ESQL modules and invoked from other ESQL code, helping developers organize common processing logic. A SET statement assigns a value to a field or variable, but it does not define a reusable routine. See the IBM App Connect Enterprise Version 11 documentation for ESQL language information.