IBM

C1000-138 Exam

IBM API Connect v10.0.3 Solution Implementation

Questions & Answers
Demo

Version: 4.0

Question: 1

In which case is the customization of a native OAuth provider not immediately ready to be used?

A. One or more "Transform" policies are used in the assembly.
B. Grant types are added.
C. A logic operator is used in the assembly.
D. A policy with a reference to a TLS profile is added.

Answer: D

Explanation:

OAuth Provider Customization: When customizing a native OAuth provider in IBM API Connect, certain configurations and policies can affect its readiness for immediate use.
TLS Profile Reference: Adding a policy that references a TLS (Transport Layer Security) profile introduces additional security configurations that need to be validated and applied. This process can delay the immediate readiness of the OAuth provider.
Impact of TLS Profiles: TLS profiles are used to secure communications and ensure data integrity and confidentiality. When a policy with a TLS profile reference is added, the system must ensure that the TLS settings are correctly configured and operational, which can take additional time.
Other Options:
Transform Policies: These are used to modify the request or response messages and do not inherently delay the readiness of the OAuth provider.
Grant Types: Adding grant types involves configuring the OAuth provider to support different methods of obtaining access tokens, which is a standard part of customization and does not delay readiness.
Logic Operators: Using logic operators in the assembly is related to the flow control of the API assembly and does not directly impact the readiness of the OAuth provider.
Reference:
IBM API Connect documentation and best practices for OAuth provider customization.
General principles of TLS and its impact on API security configurations.

Question: 2

Given the API Endpoint, "https://cor.client.rtc/savings/annual/", which statement is correct with default Catalog settings?

A. "annual" is the name of the Catalog the API is deployed to.

B. "savings" is the name of the gateway cluster the Catalog is set to.
C. "savings" is the name of the Catalog the API is deployed to.
D. "annual" is the name of the API being called.

Answer: C

Explanation:

API Endpoint Structure: In IBM API Connect, the structure of an API endpoint URL typically includes the base URL, followed by the catalog name, and then the API name.
Catalog Name: The segment of the URL immediately following the base URL (in this case, “savings”) is generally the name of the catalog to which the API is deployed. This is a default setting in IBM API Connect.
API Name: The last segment of the URL (in this case, “annual”) is usually the name of the API being called.
Default Catalog Settings: With default catalog settings, the catalog name is included in the URL to distinguish between different catalogs. This helps in organizing and managing APIs across different environments or stages (e.g., development, testing, production).
Reference:
IBM API Connect documentation on API endpoint structure and catalog settings.
General principles of API management and deployment in IBM API Connect.

Question: 3

Which of the following is true for Products?

A. A Product can be published to selected communities.
B. A Product must be published before it is staged.
C. APIs become accessible when a Product is staged on the Developer Portal.
D. When a Product is staged or published, it is visible for all communities.

Answer: A

Explanation:

Product Publication: In IBM API Connect, a Product is a collection of APIs and Plans that can be published to a Developer Portal. When publishing a Product, you have the option to select specific communities to which the Product will be available.
Selected Communities: This feature allows API providers to control access to their APIs by making them available only to certain groups or communities. This is useful for managing access based on different criteria such as user roles, subscription levels, or organizational units.
Staging vs. Publishing:
Staging: When a Product is staged, it is deployed to a staging environment where it can be tested and validated before being made publicly available.
Publishing: After successful staging, the Product can be published to the Developer Portal, making it accessible to the selected communities.
Accessibility of APIs: APIs within a Product become accessible to developers when the Product is published to the Developer Portal, not just when it is staged.

Visibility: The visibility of a Product is controlled by the API provider. It can be restricted to specific communities or made available to all, depending on the publication settings.
Reference:
IBM API Connect documentation on Product management and publication.
Best practices for managing API access and visibility in IBM API Connect.

Question: 4

Which statement is true about the use of $ref?

A. Gatewayscript can use $ref instead of included files.
B. $ref can only be defined in YAML files.
C. $ref must be defined at the root level of the YAML file.
D. When an API is published, the $ref is replaced with the contents of the referenced file.

Answer: D

Explanation:

$ref Usage: The $ref keyword is used in OpenAPI (formerly Swagger) specifications to reference reusable components, such as schemas, parameters, and responses, defined elsewhere in the document or in external files.
Replacement Mechanism: When an API is published, the $ref is processed and replaced with the actual contents of the referenced file or component. This allows for modular and maintainable API definitions, where common elements can be defined once and reused across multiple APIs.
YAML and JSON: The $ref keyword can be used in both YAML and JSON files, which are common formats for defining OpenAPI specifications. It is not limited to YAML files.
Location Flexibility: $ref can be defined at various levels within the YAML or JSON file, not necessarily at the root level. It can be used within objects, arrays, and other structures as needed.
Gatewayscript: While Gatewayscript can include external files, it does not use the $ref keyword in the same way as OpenAPI specifications. Gatewayscript has its own mechanisms for including and referencing external scripts.
Reference:
IBM API Connect documentation on OpenAPI specifications and the use of $ref.
General principles of API design and modularization using OpenAPI.

Question: 5

Which statement is true regarding API Analytics Dashboards?

A. Data from multiple Catalogs cannot be visualized on a single dashboard.
B. A visualization should be created and saved after creating a custom dashboard.
C. A single dashboard may be created that includes data from multiple Catalogs.
D. All users have the ability to create custom dashboards.

Answer: C

Explanation:

API Analytics Dashboards: In IBM API Connect, analytics dashboards provide insights into API usage, performance, and trends. These dashboards are essential for monitoring and optimizing API strategies.
Multiple Catalogs: IBM API Connect allows the creation of a single dashboard that can aggregate and visualize data from multiple catalogs. This feature is particularly useful for organizations that manage APIs across different environments or stages (e.g., development, testing, production).
Visualization Creation: While visualizations are an integral part of dashboards, they can be created and saved at any time, not necessarily after creating a custom dashboard.
User Permissions: The ability to create custom dashboards may be restricted based on user roles and permissions. Not all users may have the necessary permissions to create custom dashboards.
Data Aggregation: By including data from multiple catalogs in a single dashboard, API providers can gain a comprehensive view of their API ecosystem, making it easier to identify patterns, detect anomalies, and make informed decisions.
Reference:
IBM API Connect documentation on analytics and dashboard creation.
Best practices for API management and monitoring in IBM API Connect.