# Celigo Builder Core Certification Practice Exam (Sample)

## Study Guide


**Everything you need from our exam experts!**

# Questions

**1. What kind of operations can JavaScript hooks perform in the context of data integration?**

**A. Only basic arithmetic calculations**

**B. Only logging and monitoring systems**

**C. Only manipulating data structures**

**D. Various operations including calculations and transformations**

**2. Which characteristic distinguishes preMap hooks from postMap hooks?**

**A. PreMap hooks can run calculations**

**B. PostMap hooks can manipulate data post-mapping**

**C. PreMap hooks validate data before it is sent**

**D. PostMap hooks establish the mapping rules**

**3. Which import operation type allows for an existing record to be updated?**

**A. Insert**

**B. Update**

**C. Delete**

**D. Copy**

**4. What does the connection borrowing concurrency allow?**

**A. To prioritize all existing connections**

**B. To exceed its concurrency when unused**

**C. To deny all other concurrent connections**

**D. To limit the number of connections**

**5. What is the primary purpose of the Playground tool in the Celigo platform?**

**A. To save integration data permanently**

**B. To experiment outside of an integration**

**C. To track API calls**

**D. To generate reports**

**6. In token authentication, what is typically issued after verifying a server request?**

**A. A database connection string**

**B. An API key and password**

**C. A temporary credential token**

**D. A signed JWT**

**7. Which statement accurately describes a condition for importing data through MyAPI?**

**A. The request body must include user permissions.**

**B. The request must only contain a confirmation message.**

**C. The import ID must be specified in the request body.**

**D. The endpoint can be any default path.**

**8. HMAC verification is utilized primarily for what purpose in data transmission?**

**A. To issue API keys for server access**

**B. To set encryption and encoding standards**

**C. To store temporary credentials securely**

**D. To facilitate universal connector functionality**

**9. What is the main function of digest authentication in server communication?**

**A. To encrypt data with a secret key**

**B. To verify the identity of users**

**C. To send data back to the server encrypted with credentials**

**D. To facilitate short-lived token exchanges**

**10. When dynamically pulling data from a nested array, which tool can be useful?**

**A. Code compiler**

**B. Handbars helpers**

**C. Integrated development environment (IDE)**

**D. Text editor**

# Answers

1. D
2. B
3. B
4. B
5. B
6. B
7. C
8. B
9. C
10. B

# Explanations

## 1. What kind of operations can JavaScript hooks perform in the context of data integration?

**A. Only basic arithmetic calculations**

**B. Only logging and monitoring systems**

**C. Only manipulating data structures**

**D. Various operations including calculations and transformations**

**JavaScript hooks in the context of data integration are powerful tools that allow for various operations beyond the limitations of basic functions. They can perform calculations, enabling developers to manipulate and compute data as needed. Additionally, they facilitate transformations, allowing for modifications of data structures and formats to fit specific requirements during data integration processes. This versatility makes JavaScript hooks essential for handling complex data workflows, as they can adapt data dynamically based on business logic, perform validations, and format data before passing it to other systems. The focus on "various operations" highlights the comprehensive capabilities of JavaScript hooks in integrating and processing data effectively within different platforms and applications. This flexibility is crucial for developers when managing diverse datasets and performing intricate data operations, ensuring smooth and efficient integrations.**

## 2. Which characteristic distinguishes preMap hooks from postMap hooks?

**A. PreMap hooks can run calculations**

**B. PostMap hooks can manipulate data post-mapping**

**C. PreMap hooks validate data before it is sent**

**D. PostMap hooks establish the mapping rules**

**The characteristic that distinguishes preMap hooks from postMap hooks is that postMap hooks can manipulate data after the mapping process has occurred. PostMap hooks are specifically designed to operate on the data once it has been mapped, allowing users to apply additional transformations or adjustments to the data before it is finally processed or sent to the destination system. This manipulation can include tasks such as cleaning data, formatting it for specific uses, or enriching it with additional information. PreMap hooks, on the other hand, are typically utilized before the mapping takes place. They focus on tasks such as validating incoming data or performing calculations that are necessary to prepare the data for mapping. Thus, postMap hooks have a distinct purpose related to post-processing, while preMap hooks are oriented toward preparing or checking data beforehand.**

## 3. Which import operation type allows for an existing record to be updated?

**A. Insert**

**B. Update**

**C. Delete**

**D. Copy**

**The import operation type that allows for an existing record to be updated is the "Update" operation. This type of operation is specifically designed to modify pre-existing records within a data source. When you perform an update, you typically provide an identifier for the record you want to change, along with the new values for the fields that need to be updated. Using the update operation ensures that only the specified fields in the existing record are changed, while all other data remains intact. This is crucial for maintaining data integrity, as it allows you to refresh or correct information without creating duplicate entries or losing any additional data that is still relevant. In contrast, other operation types such as insert serve to add new records rather than modify existing ones, delete removes records without updating them, and copy typically refers to creating duplicate records rather than altering existing data. Understanding these distinctions is essential for effective data management and integration within your systems.**

## 4. What does the connection borrowing concurrency allow?

**A. To prioritize all existing connections**

**B. To exceed its concurrency when unused**

**C. To deny all other concurrent connections**

**D. To limit the number of connections**

**The concept of connection borrowing concurrency pertains to how systems manage concurrent connections to optimize resource usage. When a system utilizes connection borrowing, it allows for the concurrency limit to be exceeded temporarily when there are unused connections. This means that if certain connections are idle and are not being used, the system can borrow from that pool of available connections to handle additional load or requests that exceed the typical concurrency threshold. This feature is particularly useful in scenarios where there may be spikes in demand or when there are periods of inactivity, allowing the system to maximize efficiency and maintain performance without permanently increasing the number of concurrent connections. It promotes flexibility and optimal resource management by making sure that available connections are utilized effectively rather than remaining idle. In this context, the other options do not accurately reflect the functionality of connection borrowing concurrency. For instance, prioritizing all existing connections does not align with the concept of borrowing; rather, it focuses on managing current connections without allowing for surplus handling. Dismissing all concurrent connections is counterproductive to the goal of scalability, while limiting the number of connections runs counter to the principle of maximizing resources by borrowing from those that are underutilized. Hence, the accuracy of the chosen answer centered on the ability to exceed standard limits when connections are not actively**

## 5. What is the primary purpose of the Playground tool in the Celigo platform?

**A. To save integration data permanently**

**B. To experiment outside of an integration**

**C. To track API calls**

**D. To generate reports**

**The Playground tool in the Celigo platform is primarily designed for experimentation outside of established integrations. It serves as a safe environment where users can test different scenarios, run simulations, and develop new functionalities without affecting live data or existing integrations. This space allows integration developers and users to explore various configurations, validate business logic, and troubleshoot issues in a controlled setting. The other options do not accurately reflect the true purpose of the Playground tool. For instance, while saving integration data and tracking API calls are important aspects of integration development, these functionalities are typically managed within the main integration settings rather than in the Playground. Generating reports is also not a primary function of the Playground; instead, reporting tools and dashboards on the platform are suited for that purpose. Thus, the correct answer highlights the Playground's role as a sandbox for safe testing and experimentation.**

## 6. In token authentication, what is typically issued after verifying a server request?

**A. A database connection string**

**B. An API key and password**

**C. A temporary credential token**

**D. A signed JWT**

**In token authentication, after a server request is successfully verified, the typical outcome is the issuance of a temporary credential token. This token serves as a means of verifying the identity of the user in subsequent requests without having to constantly send their credentials. This approach enhances security by limiting the exposure of sensitive information and allowing for temporary access, which can be revoked or expired after a certain duration. While an API key and password can also be used for authentication, they are typically credentials that grant access on a more permanent basis, rather than a dynamic token that can expire or be invalidated. In contrast, the use of a temporary credential token is essential in scenarios where applications need to maintain session-based authentication or where frequent revalidation of user identity is required without exposing long-term credentials. Options like a database connection string are not relevant in the context of token authentication, as they pertain to establishing communication between an application and a database rather than user authentication. Similarly, while a signed JWT (JSON Web Token) represents a form of token, the phrasing of the answer is specific to the generic concept of a temporary credential token in token authentication. Thus, the issuance of such a temporary token after request verification aligns perfectly with the standard practices of token-based authentication systems.**

## 7. Which statement accurately describes a condition for importing data through MyAPI?

**A. The request body must include user permissions.**

**B. The request must only contain a confirmation message.**

**C. The import ID must be specified in the request body.**

**D. The endpoint can be any default path.**

**The correct choice highlights the importance of specifying the import ID in the request body when importing data through MyAPI. This import ID serves as a unique identifier for the data being processed, ensuring that the system can correctly relate the incoming data to the appropriate import settings and mechanisms already established within MyAPI. Including this import ID is crucial for maintaining data integrity and consistency during the import process. It allows the API to function correctly by linking the data to a predefined structure and processing logic. In contrast, other options do not capture the key requirements for a successful data import through MyAPI. User permissions do not typically need to be included directly in the request body, as these are usually handled by the API's authentication and authorization processes. A request that solely contains a confirmation message would not fulfill the necessary requirements for data importation, as there should be specific data payloads involved. Lastly, the endpoint for data import is not flexible or arbitrary; it must adhere to specific paths defined by the API. Using any default path would be improper and could lead to errors or undefined behaviors in the import process.**

## 8. HMAC verification is utilized primarily for what purpose in data transmission?

**A. To issue API keys for server access**

**B. To set encryption and encoding standards**

**C. To store temporary credentials securely**

**D. To facilitate universal connector functionality**

**HMAC verification is primarily used for ensuring the integrity and authenticity of data during transmission. It involves creating a hash-based message authentication code that combines data with a secret key, allowing the recipient to confirm that the message has not been altered and is indeed from a legitimate source. This mechanism is essential in scenarios where data integrity and security are paramount, such as in API communications. While setting encryption and encoding standards is important in securing data, the specific purpose of HMAC is more focused on verifying that the data has not been tampered with and is authentic. The underlying mechanism of HMAC ensures that even if the data is intercepted, any changes made would be detectable by the verification process, thus maintaining trust and security in the communication. The other choices, while they relate to security and data handling, do not directly pertain to the specific purpose of HMAC verification, which is about data integrity and authenticity, rather than issuing keys, storing credentials, or facilitating connector functionality.**

## 9. What is the main function of digest authentication in server communication?

**A. To encrypt data with a secret key**

**B. To verify the identity of users**

**C. To send data back to the server encrypted with credentials**

**D. To facilitate short-lived token exchanges**

**The main function of digest authentication in server communication is to verify the identity of users by ensuring that the credentials sent during the authentication process are not transmitted in clear text and are not easily compromised. Digest authentication enhances security by creating a hash of the user's credentials and the request information, which minimizes the risk of credential exposure during transmission. In this authentication method, a unique nonce (a number used once) is generated by the server and included in the authentication challenge. The client then combines this nonce with the user's credentials and submits the hashed result back to the server. This allows the server to confirm the user's identity without requiring the actual password to be sent over the network. While the option of sending data encrypted with credentials touches on aspects of how authentication might be perceived, it does not capture the essence of what digest authentication specifically accomplishes in its primary function of user identity verification. Encryption of data and tokens pertains to broader terms in secure communication, but digest authentication focuses squarely on confirming who the user is at the time of the request.**

## 10. When dynamically pulling data from a nested array, which tool can be useful?

**A. Code compiler**

**B. Handbars helpers**

**C. Integrated development environment (IDE)**

**D. Text editor**

**Using Handlebars helpers is particularly beneficial when dynamically pulling data from a nested array because Handlebars is a templating engine designed to make it easier to work with data, especially when rendering content based on complex data structures like nested arrays. Helpers in Handlebars allow for custom logic and manipulation of data during the template rendering process. For example, if you have a nested array where items are grouped in sub-arrays or objects, Handlebars helpers can be used to iterate over these structures and access specific properties or manipulate the data within the templates. This makes it straightforward to display the necessary information from a potentially complex dataset without cluttering the template with logic. While other tools like a code compiler, IDEs, or text editors serve important functions in coding and development workflows, they do not provide the specific capabilities for manipulating and rendering nested data structures in templates as effectively as Handlebars helpers.**