

# MasterControl Certified Integration Developer Exam

[August 15 meeting notes](#)

[August 28 meeting notes](#)

[Untitled](#)

[Wednesday, October 9, 2024](#)


---

## Course outlines

[Integrations Setup \(lock next courses\)](#)

Pre Exam? Maybe include before setup? Have Andy send? Basically, want to make sure people are prepared for the class.

 [Course #1—Integrations: JavaScript Basics](#)


 [Course #2—Integrations: Build an Integration \(Virtual Lab\) \[writing\]](#)

 [Course #3—Integrations: Parameters, Configuration Values, and Session Items](#)

 [Course #4—Integrations: Use Variables \(Virtual Lab\)](#)


 [Course #5—Integrations: HTTPS Requests](#)


 [Course #6—Integrations: Convert JSON Between String and Object](#)

 [Course #7: Call an API \(Virtual Lab\)](#)

 [Course #8: Set and Add Values for an Mx Integration \(Virtual Lab\)](#)

 [Course #9: Working with Files \(Virtual Lab\)](#)

 [Course #10: Use .csvs \[Virtual Lab\] \[writing\]](#)

 [Get Certified! \(locked/pre-reqs\).](#)

---

## Exam questions

### ▼ Course #1 - Exam Questions

**Q1: Which of the following most accurately defines a variable?**

- A) A variable is a fixed value that cannot be changed once assigned.
  - B) A variable is a label used to store and manage data, allowing you to assign and change values as needed.\*
  - C) A variable is a function that performs calculations based on its label.
  - D) A variable is a container that stores numbers and cannot be updated.
- 

**Q2: True or False: In JavaScript, you must specify a variable type (integer, string, etc.) when declaring variables.**

False

---

**Q3: Which of the following keywords can be used to declare variables in JavaScript? Select all that apply.**

- A) var\*
  - B) let\*
  - C) set
  - D) define
  - E) new
- 

**Q4: Which of the following is an example of *declaring* a variable in JavaScript?**

- A) let myVar;\*
- B) myVar = 4;

- C) myVar++;
  - D) myVar == 4;
  - E) myVar 4 + 2;
- 

**Q5: Which of the following is an example of *assigning a value to a variable* in JavaScript?**

- A) let myVar;
  - B) myVar = 4;\*
  - C) myVar++;
  - D) myVar == 4;
  - E) myVar 4 + 2;
- 

**Q6: Which of the following is an example of *declaring a variable with an initial value* in JavaScript?**

- A) let myVar = 4;\*
  - B) myVar = 4;
  - C) myVar++;
  - D) myVar == 4;
  - E) myVar 4 + 2;
- 

**Q7: Declaring a variable \_\_\_\_\_, while assigning a value to a variable \_\_\_\_\_.**

- A) creates it in memory; stores data in that variable\*
  - B) stores data in that variable; creates it in memory
  - C) names the variable; removes it from memory
  - D) adds a value to the variable; clears the memory
  - E) assigns a function to the variable; creates an object
-

### Q8: Which of the following most accurately describes reserved words in JavaScript?

A) Reserved words are keywords that cannot be used as names for variables, functions, or other elements because they serve a special purpose in the JavaScript syntax.\*

*This is the correct answer.*

B) Reserved words are words that can only be used for naming variables, not functions or objects.

*This is incorrect because reserved words cannot be used for naming at all.*

C) Reserved words are terms that can be used interchangeably with variable names in JavaScript.

*This is incorrect because reserved words cannot be used as variable names.*

D) Reserved words are optional keywords used for styling and commenting in JavaScript.

*This is incorrect because reserved words are core parts of JavaScript syntax, not for styling or comments.*

E) Reserved words are case-sensitive and can be used as long as they are written in lowercase.

*This is incorrect because reserved words cannot be used at all, regardless of case.*

---

### Q9: The following code is throwing an error. What is the cause of the error?

```
let for = 10;  
console.log(for);
```

A) The variable name "for" is a reserved word in JavaScript and cannot be used as a variable name.\*

*Correct answer: "for" is a reserved word and cannot be used as a variable.*

B) The variable "for" is not initialized properly.

*Incorrect: The issue is with the reserved word, not initialization.*

C) There is a syntax error because the variable should be declared with `const` instead of `let`.

*Incorrect: The problem is not the use of `let`, but the reserved word.*

D) The `console.log()` method is not correctly printing the variable.

*Incorrect: The error occurs before `console.log()` is even executed.*

E) The number "10" cannot be assigned to a variable of type "for".

*Incorrect: The error is about using the reserved word, not the type of data.*

---

### Q10: What is the purpose of conditional statements?

A) They perform different actions based on different conditions.

*Correct: Conditional statements allow your code to decide what actions to perform based on whether certain conditions are true or false.*

B) They repeat actions a set number of times.

*Incorrect: This describes loops, not conditional statements.*

C) They store values that can be accessed and modified later.

*Incorrect: This describes variables, not conditional statements.*

D) They organize code into reusable blocks.

*Incorrect: This describes functions, not conditional statements.*

E) They handle errors in the program.

*Incorrect: This describes error-handling mechanisms like `try` and `catch`, not conditional statements.*

---

### Q11: Which of the following code follows the correct syntax for a conditional statement?

*Note to person building exam: Perhaps include screenshots of the code shown below.*

A)

```

if (aname == 'mike') {
    MC.log('name is mike');
} else if (aname == 'kim') {
    MC.log('name is kim');
} else {
    MC.log('name is not mike or kim');
}

```

Correct: This is the correct syntax for an `if-else if-else` conditional statement.

B)

```

if aname = 'mike' {
    MC.log('name is mike');
} elseif aname = 'kim' {
    MC.log('name is kim');
} else {
    MC.log('name is not mike or kim');
}

```

Incorrect: `=` is used for assignment, not comparison. The `if` and `else if` blocks also lack parentheses.

C)

```

if (aname === 'mike')
    MC.log('name is mike');
else (aname === 'kim') {
    MC.log('name is kim');
} else {
    MC.log('name is not mike or kim');
}

```

Incorrect: The `else` block has an incorrect condition after it; `else` should not have a condition.

## Q12: The following code is an example of what?

```
let documentBody = {  
  username: "user1234",  
  id: "6B29FC40-CA47-1067-B31D-00DD010662DA",  
  person: { fullname: "mike smith" }  
};
```

A) Static object\*

*Correct: This is an example of a static object, where the properties and values are defined at the time of creation.*

B) Dynamic object

*Incorrect: Dynamic objects are created and modified during runtime.*

C) Function

*Incorrect: This is an object, not a function.*

D) Array

*Incorrect: This is an object, not an array.*

E) Event listener

*Incorrect: There are no event listeners in this code.*

---

## Q13: The following code is an example of what?

```
let documentBody = {};  
docbody.username = "user1234";  
docbody.id = "6B29FC40-CA47-1067-B31D-00DD010662DA";  
docbody.person = {};  
docbody.person.fullname = "mike smith";
```

A) Static object

*Incorrect. The properties and values for static objects are defined at the time of creation.*

B) Dynamic object\*

*Correct. Dynamic objects are created and modified during runtime.*

C) Function

*Incorrect: This is an object, not a function.*

D) Array

*Incorrect: This is an object, not an array.*

E) Event listener

*Incorrect: There are no event listeners in this code.*

---

**Q14: True or False: With dynamic objects, object properties and values are added one by one during runtime.**

True

---

**Q15: Which of the following is an example of correct key-value pair syntax when creating an object?**

A)

```
let user = { name: "John", age: 30 };
```

*Correct: This is a valid object with key-value pairs, where `name` and `age` are keys, and `"John"` and `30` are their respective values.*

B)

```
let user = { "John", 30 };
```

*Incorrect: This is missing the key-value structure. Each value should be paired with a key.*

C)

```
let user = { name: "John": age: 30 };
```

*Incorrect: The syntax uses colons incorrectly. Each key-value pair should be separated by a comma.*



D)

```
let user = { name = "John", age = 30 };
```

*Incorrect: Keys and values in an object should be assigned using colons (:), not equal signs (=).*

---

### Q16: What kind of circumstance would warrant using a loop in your code?

A) When you must automate repetitive tasks and process large amounts of data.\*

*Correct: Loops are used to repeat actions multiple times or iterate through data sets.*

B) When you want to execute code once based on a specific condition.

*Incorrect: A conditional statement (e.g., if statement) is used to execute code once based on a condition, not a loop.*

C) When you must define the structure of an object.

*Incorrect: Object structures are defined with key-value pairs, not with loops.*

D) When you must stop your code from executing.

*Incorrect: To stop or break out of code, you use control statements like return or break, not loops.*

E) When you want to trigger a function at a specific time.

*Incorrect: This would involve event listeners or time-based methods like setTimeout() or setInterval(), not loops.*

---

### Q17: Which loop executes a block of code as long as the specified condition remains true?

A) While\*

B) For

C) Switch

- D) Continuous
  - E) If
- 

**Q18: Which loop is used when the number of iterations is known or can be determined ahead of time?**

- A) While
  - B) For\*
  - C) Switch
  - D) Continuous
  - E) If
- 

**Q19: What situation would warrant using a for loop in your integration?**

A) When you need to iterate through a list of items, such as processing an array of data.\*

*Correct.* A `for` loop is ideal for iterating through lists, arrays, or other collections, allowing you to repeat actions for each item.

B) When you need to check if a condition is true only once and execute code accordingly.

*Incorrect.* This scenario calls for a simple `if` statement, not a loop.

C) When you need to execute code only after an error occurs.

*Incorrect.* This is a situation for error handling, often using `try-catch` blocks, not a `for` loop.

D) When you need to handle a specific case based on different conditions.

*Incorrect.* This is suited for a `switch` or `if-else` statement, not a loop.

E) When you need to break out of a function immediately, regardless of the logic inside it.

*Incorrect.* This describes the use of a `return` statement, not a loop.

---

▼ Course #2 - Exam Questions

**Q20: Which of the following is a correct syntax example of JSON?**

A) `{ "name": "John", "age": 30, "city": "New York" }`

*Correct.* This is the proper JSON syntax, with key-value pairs in double quotes and no trailing commas.

B) `{ name: "John", age: 30, city: "New York" }`

*Incorrect.* In JSON, keys must be in double quotes. This looks more like JavaScript object syntax, not valid JSON.

C) `{ "name": "John", "age": 30 "city": "New York" }`

*Incorrect.* This example is missing a comma between `"age": 30` and `"city": "New York"`, which is required in JSON formatting.

D) `[ "name": "John", "age": 30, "city": "New York" ]`

*Incorrect.* This is an array, not a JSON object. JSON objects use curly braces `{ }`, not square brackets `[ ]`.

E) `{ "name": "John"; "age": 30; "city": "New York" }`

*Incorrect.* Semicolons ( `;` ) are not allowed in JSON. Commas must be used to separate key-value pairs.

---

**Q21: All integrations must include (select all that apply):**

A) `integration.json` \*

*Correct.* Every integration has an `integration.json` file, which stores configuration data for the integration.

B) `main.js` \*

*Correct.* The `main.js` file contains the core logic and script that runs the integration.

C) `README.md`

*Incorrect.* A `README.md` file is common in many projects, but it's not required for integrations.

D) `settings.cfg`

*Incorrect.* There is no default `settings.cfg` file in standard integrations.

E) `error.log`

*Incorrect.* An `error.log` file may be generated if there are errors, but it is not an inherent part of the integration structure.

---

## Q22: Scheduled integrations must include what section?

A) a cron section

*Correct.* Scheduled integrations use a `cron` section to define the schedule for when the integration runs.

B) a time zone section

*Incorrect.* While the time zone may be relevant, it's not a required section in scheduled integrations.

C) an execution block

*Incorrect.* An execution block defines the actions taken, but it's not specific to scheduling.

D) a `schedule.json` file

*Incorrect.* There is no `schedule.json` file in the standard structure of scheduled integrations.

E) an event trigger

*Incorrect.* Event triggers may be used in event-based integrations, but they're not specific to scheduled ones.

---

## Q23: Which of the following is NOT a required element in your `integration.json` configuration file?

A) `mainfile`

B) `type`

C) `runtime`

D) `modules`

E) version\*

---

**Q24: Which of the following most accurately describes the difference between scheduled and triggered integrations?**

A) Scheduled integrations run automatically at specified intervals, while triggered integrations execute in response to specific events.\*

*Correct.* This accurately defines the difference: scheduled integrations run based on a timer (e.g., using cron jobs), while triggered integrations respond to real-time events.

B) Scheduled integrations require user input to start, while triggered integrations run automatically.

*Incorrect.* Scheduled integrations run automatically based on a schedule, not user input.

C) Triggered integrations run at regular intervals, while scheduled integrations run in response to events.

*Incorrect.* This reverses the definitions—triggered integrations respond to events, and scheduled integrations run at set intervals.

D) Both scheduled and triggered integrations only run when manually started.

*Incorrect.* Both scheduled and triggered integrations can run automatically, either by schedule or in response to events.

E) Scheduled integrations are only for reporting, and triggered integrations are only for data processing.

*Incorrect.* Both scheduled and triggered integrations can handle a variety of tasks, including reporting and data processing.

---

▼ Course #3 - Exam Questions

**Q25: Which keyword best matches the following description: values passed in at runtime from an external system, allowing the integration to adjust its behavior dynamically?**

A) Parameters \*

*Correct.* Parameters are values passed in at runtime that influence how the integration operates.

B) Configuration items

*Incorrect.* Configuration items are typically set in advance and do not change during runtime.

C) Session values

*Incorrect.* Session values are specific to the current session but are not passed in from an external system at runtime.

---

**Q26: Which keyword best matches the following description: stored values that remain constant across executions and are used to define settings and credentials necessary for the integration?**

A) **Parameters**

*Incorrect.* Parameters are passed in at runtime and can vary with each execution.

B) **Configuration items \***

*Correct.* Configuration items are constant values like settings and credentials that do not change between executions.

C) **Session values**

*Incorrect.* Session values are specific to a single execution and can change across different runs.

---

**Q27: Which keyword best matches the following description: values that typically change with each execution and need to be remembered between runs; these values help track state or progress across different runs of the integration?**

A) Parameters

*Incorrect.* Parameters are passed at runtime and can change with each execution, but they are not typically used to track state or progress.

B) Configuration items

*Incorrect.* Configuration items are constant values and do not change between executions, nor are they used for tracking progress.

C) Session values \*

*Correct.* Session values are used to track the state or progress across executions, allowing the integration to remember specific data between runs.

---

## **Q28: Why is it important to encrypt sensitive values before passing them in an integration?**

A) To prevent unauthorized access to sensitive data during transmission.\*

*Correct.* Encrypting sensitive values, such as API keys or credentials, ensures that they are protected during transmission and cannot be easily intercepted or accessed by unauthorized parties.

B) To make sure the data is compressed for faster transmission.

*Incorrect.* Encryption secures data, but it is not related to data compression, which deals with reducing file size for faster transmission.

C) To avoid errors caused by incorrect data formatting.

*Incorrect.* Encryption is not a solution for data formatting errors; it only ensures that the data is unreadable without proper decryption.

D) To ensure the data remains in plain text for easier debugging.

*Incorrect.* Leaving data in plain text would expose it to unauthorized access, making debugging easier but compromising security.

**E) To ensure sensitive data is stored securely in the integration's log files.**

*Incorrect.* Encryption protects data during transmission, but secure logging requires a separate process to ensure sensitive information isn't written to log files in plaintext.

---

## **Q29: You can access variables from your code using which of the following?**

A) EventExecution class\*

*Correct.* The `EventExecution` class is used to access variables, such as parameters, session values, and configuration items within an integration.

B) Log class

*Incorrect.* The `Log` class is used for logging messages, not for accessing variables.

C) IntegrationManager class

*Incorrect.* There is no `IntegrationManager` class designed for accessing variables directly within your code.

D) APIRequest class

*Incorrect.* The `APIRequest` class is typically used for handling HTTP requests, not for accessing internal variables of an integration.

E) VariableManager class

*Incorrect.* No such class exists for managing variables in this context.

---

### **Q30: Which special character instructs an integration to decrypt specific values?**

A) \*

*Correct.* The asterisk ( `*` ) is used to indicate that a value should be decrypted by the system.

B) \$

*Incorrect.* The dollar sign ( `$` ) is often used in some programming languages or systems to reference variables, but it is not used for decryption in this context.

C) #

*Incorrect.* The hash symbol ( `#` ) may be used as a comment marker or for other purposes, but it does not indicate decryption.

D) @

*Incorrect.* The at symbol ( `@` ) may be used in email addresses or annotations in some languages, but it is not related to decrypting values.

E) &



*Incorrect.* The ampersand ( & ) is used in some contexts to denote concatenation or references, but not for decryption.

▼ Course #4 - Exam Questions

N/A

▼ Course #5 - Exam Questions

### **Q31: What does HTTP stand for?**

#### **A) Hypertext Transfer Protocol**

Why it's correct: This is the full and correct name for HTTP. It refers to the protocol used for transferring hypertext documents between web browsers and servers on the internet.

#### **B) Hypertext Transport Process**

Why it's incorrect: While "Hypertext" is correct, "Transport" and "Process" are incorrect. The correct term is "Transfer," and HTTP is a "Protocol," not a process.

#### **C) Hyperlink Text Transmission Protocol**

Why it's incorrect: This option incorrectly uses "Hyperlink" instead of "Hypertext" and "Transmission" instead of "Transfer." HTTP governs the transfer of hypertext documents, not just hyperlinks, and "Transfer" is the key term.

#### **D) Hyperlink Transfer Procedure**

Why it's incorrect: "Hyperlink" is the wrong term; HTTP refers to "Hypertext." Also, "Procedure" is not correct, as HTTP is a "Protocol," not a procedure.

#### **E) Hypertext Transfer Procedure**

Why it's incorrect: This option correctly uses "Hypertext" and "Transfer," but "Procedure" is not the correct term—HTTP is a protocol, not a procedure.

---

### **Q32: What are the three methods covered in the course on HTTP requests? Select all that apply.**

- A) GET\*
  - B) POST\*
  - C) PUT\*
  - D) NOTE
  - E) MARK
- 

### Q33: What is the purpose of HTTP requests?

**A) They encrypt data to ensure secure communication between systems.**

*Correct.* HTTPS requests encrypt the data exchanged between systems, ensuring that sensitive information is protected during transmission and reducing the risk of unauthorized access or data breaches.

**B) They compress data for faster transmission.**

*Incorrect.* HTTPS ensures security, not data compression. Compression is related to performance optimization, not securing the data.

**C) They convert data into JSON format.**

*Incorrect.* HTTPS is a communication protocol for secure data transfer. Converting data into JSON format is a separate process used when handling data structures, unrelated to HTTPS itself.

**D) They ensure that data is sent only over local networks.**

*Incorrect.* HTTPS works over the internet and does not restrict communication to local networks. Its purpose is to secure the data transfer, regardless of the network type.

**E) They bypass authentication steps for faster processing.**

*Incorrect.* HTTPS does not bypass authentication. In fact, it often works with authentication mechanisms like SSL certificates to ensure the security of communication.

A) HTTPS requests ensure secure communication between your integration and external services by encrypting the data exchanged. This protects sensitive information and helps prevent unauthorized access or data breaches during transmission.

---

### Q34: What is an `HTTPResponse`?

A) The server's reply after processing a request, containing a status code, headers, and body.\*

*Correct.*

An `HTTPResponse` is the response sent by the server after handling an HTTP request. It includes a status code (e.g., 200 for success), headers (metadata), and the body (the actual data or content).

B) The server's error message when a request fails.

*Incorrect.*

An `HTTPResponse` is not just for errors—it contains both successful and failed responses, depending on the status code.

C) The original request that was sent to the server.

*Incorrect.*

The `HTTPResponse` is the reply to the request, not the request itself.

D) A JavaScript function used to send requests to servers.

*Incorrect.*

This describes a function for making requests, such as `HTTP.request()`, not the response received from the server.

E) A list of all the servers your request passed through.

*Incorrect.*

An `HTTPResponse` doesn't track the servers involved; it only contains the server's reply to the request.

#### ▼ Course #6 - Exam Questions

### Q35: What is the purpose of `JSON.stringify()` ?

A) To convert a JavaScript object into a JSON string.

*Correct.*

`JSON.stringify()` is used to transform a JavaScript object into its JSON string representation, which can then be easily stored, transmitted, or logged.

B) To parse a JSON string into a JavaScript object.

*Incorrect.*

This describes the function `JSON.parse()`, which is the reverse operation of `JSON.stringify()`.

C) To encode JavaScript objects into base64 format.

*Incorrect.*

`JSON.stringify()` does not encode objects into base64; it simply converts them into a JSON string.

D) To store JavaScript variables in local storage.

*Incorrect.*

While you might use `JSON.stringify()` before saving an object to local storage, the function itself does not store variables.

E) To execute JavaScript objects as JSON code.

*Incorrect.*

`JSON.stringify()` does not execute code; it only converts an object into a JSON-formatted string.

---

Q36: What is the purpose of `JSON.parse()`?

**A) To convert a JSON string into a JavaScript object**

**Why it's correct:** The purpose of `JSON.parse()` is to take a well-formed JSON string and convert it into a corresponding JavaScript object. This allows developers to work with JSON data in the context of JavaScript.

**Incorrect Answers:**

**A) To convert a JavaScript object into a JSON string**

**Why it's incorrect:** This describes the purpose of `JSON.stringify()`, not `JSON.parse()`. `JSON.stringify()` converts JavaScript objects into JSON strings, whereas `JSON.parse()` does the opposite.

**B) To parse a JavaScript function into a JSON string**

**Why it's incorrect:** JSON is strictly a data format, and it doesn't support functions. Therefore, `JSON.parse()` cannot parse JavaScript functions, only data in JSON string format

**C) To validate whether a string is a valid JSON format**

**Why it's incorrect:** While `JSON.parse()` may throw an error if the JSON string is invalid, its main purpose is not validation but to convert the string into an object. Validating JSON format can be an indirect result, but it's not the primary purpose.

**To format a JavaScript object for easy reading in JSON format**

**Why it's incorrect:** Formatting or pretty-printing JSON data is not the function of `JSON.parse()`. This task is better handled by other methods like `JSON.stringify()` with formatting options. `JSON.parse()` simply converts JSON strings into objects.

B)

C)

D)

E)

- ▼ Course #7 - Exam Questions
- ▼ Course #8 - Exam Questions
- ▼ Course #9 - Exam Questions
- ▼ Course #10 - Exam Questions