DUMPSARENA

Salesforce Certified Marketing Cloud Engagement Developer

Salesforce MCE-Dev-201

Version Demo

Total Demo Questions: 20

Total Premium Questions: 208

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Data Modeling	16
Topic 2, Data Management	51
Topic 3, Content Creation and Delivery	6
Topic 4, Automation	3
Topic 5, Programming	70
Topic 6, API	50
Topic 7, Security	12
Total	208

QUESTION NO: 1

Certification Aid sends an email to a newly imported List with Subscribers who have no associated Subscriber Key. Which value will become the Contact Key? Choose 1.

A. ContactID

B. Email address

C. Subscriber ID

D. Unique random number

ANSWER: B

Explanation:

Email address is correct. In Marketing Cloud Engagement, the Subscriber Key is the identifier used to associate a subscriber record with a contact. For list-based imports, when no Subscriber Key value is supplied, Marketing Cloud Engagement uses the subscriber's email address as the Subscriber Key. Because the Subscriber Key is mapped to the Contact Key for email subscribers, that email-address value becomes the Contact Key for the newly created contact record.

This default behavior allows imported subscribers to be created and addressed even when the source file contains only email data. It also means that an email address used in this way must be treated as the identifying value for that subscriber across subsequent imports and sends, unless an organization deliberately supplies and consistently manages its own unique Subscriber Key values. Salesforce recommends using a stable, unique identifier as the Subscriber Key when available, particularly when an individual can have changing email addresses or multiple email addresses.

See Salesforce's documentation on Subscriber Keys and importing subscribers.

QUESTION NO: 2

Certification Aid wants to add new customers to a cross-channel welcome campaign when they register on the company website. Which API should be used for this? Choose 1.

A. Personalization Builder API

B. Event Notification API

C. Transactional Messaging API

D. Journey Builder API

ANSWER: D

Explanation:

Journey Builder API is the appropriate choice because it supports programmatically admitting an individual contact into a published Journey Builder journey through an API event. When registration occurs on the company website, the website or its backend can send an event payload to Marketing Cloud Engagement. The payload identifies the contact and supplies the event data configured for the journey's API entry source. This event injection starts that person's progression through the welcome journey immediately, allowing the journey to coordinate the planned sequence of email, SMS, push, advertising, or other configured channel activities. The implementation requires a published journey with an API event entry source and an event definition key that matches the key used in the API request. The registration system must authenticate to Marketing Cloud Engagement and submit the event data in the expected format. This design cleanly separates the website's registration transaction from the campaign orchestration logic while enabling a timely, personalized cross-channel onboarding experience. Salesforce documents the event endpoint and required request structure in its Fire an Event reference, and provides broader implementation guidance in the Journey Builder API documentation.

QUESTION NO: 3

An UpdateDE AMPscript function is used to update a column value in a data extension row when an email is sent. The emailis being sent to 250,000 subscribers, but the user decides to cancel the send during the sending process and only 400 emails are sent.

How many subscriber rows would be affected by the UpdateDE function?

A. No rows are updated

B. All 250,000 subscribers

C. 400 subscribers who were sent the email

D. Only subscribers who exist in All Subscribers

ANSWER: C

Explanation:

400 subscribers who were sent the email is correct. UpdateDE is an AMPscript data-extension function that runs in the context of a subscriber's email rendering during a send. It updates the specified row when that subscriber's message is processed, using the supplied search column/value pair to locate the row and the supplied column/value pair to make the change. It is not a batch operation that preemptively updates every row associated with the send audience. In this scenario, the send was cancelled after 400 messages had been sent. Those 400 subscribers' messages were processed and rendered, so the UpdateDE call was executed for those send contexts. The remaining targeted subscribers did not receive a processed email as part of the cancelled send and therefore did not execute the AMPscript function. Consequently, the data extension update affects 400 subscriber rows, assuming the function's lookup criteria identifies a row for each recipient. Salesforce documents UpdateDE as updating a data extension row from AMPscript: UpdateDE Function Reference. For related send-time AMPscript behavior, see Salesforce's AMPscript documentation.

QUESTION NO: 4

Which SSJSlibrary can be used in landing pages? Choose 1.

A. None

B. Core

C. Both

D. Platform

ANSWER: C

Explanation:

Both is correct because Salesforce Marketing Cloud Engagement supports both Server-Side JavaScript libraries in CloudPages landing pages: the Core library and the Platform library. The Core library supplies general-purpose objects and functions for common server-side tasks, including working with data extensions, rows, arrays, strings, dates, and HTTP requests. The Platform library provides platform-oriented functionality, including functions that interact with Marketing Cloud Engagement features and APIs. In a landing page, a developer can load and use either library according to the task at hand, and can use functionality from both in the same SSJS implementation when needed. This makes CloudPages a flexible SSJS execution context for processing request parameters, retrieving or updating subscriber-related data, personalizing page output, and integrating with Marketing Cloud resources. Salesforce documents the Core and Platform libraries as the two SSJS libraries, and its SSJS reference identifies CloudPages/landing pages as a supported context for their applicable functions. See the Salesforce SSJS overview and the SSJS landing pages documentation.

QUESTION NO: 5

A developer wants to create an AMPscript FOR loop that populates HTML table rows based on the number of rows and data in a target DE. Where should the developer place the FOR keyword to begin the loop?

A. Before the tag

B. Before the tag

C. Before the tag

D. Before the tag

ANSWER: D

Explanation:

Before the <tr> tag is correct because the AMPscript `FOR` block must wrap the complete HTML row that is to be repeated for each record returned from the target data extension. The developer can retrieve the data-extension rows into a rowset, use `RowCount()` as the loop boundary, and start the loop immediately before the opening table-row element. Within that row, functions such as `Row()` and `Field()` can retrieve the values for the current iteration and insert them into the appropriate table cells. The `NEXT` statement belongs immediately after the closing `</tr>` element, ensuring that each iteration outputs one valid, complete HTML table row. This structure keeps the table itself as a single container while AMPscript dynamically generates its repeated rows. AMPscript FOR loops use the form `FOR @index = start TO end DO`, followed by `NEXT @index`; placing those delimiters around the row is the appropriate pattern when the desired repeated unit is an HTML row. See Salesforce's AMPscript FOR loop documentation and its RowCount function reference.

QUESTION NO: 6

A developer is creating a CloudPage form that allows a customer to save a preferred store. The Preferences data extension uses SubscriberKey as its primary key. The record should be created when the subscriber has no existing preference and updated when one already exists.

Which AMPscript function should the developer use?

A. InsertData()

B. UpdateData()

C. UpsertData()

D. LookupRows()

ANSWER: C

Explanation:

`UpsertData()` is correct because it updates a matching data extension row when the supplied primary-key value exists and inserts a new row when it does not. In this scenario, using `SubscriberKey` as the search field ensures that each customer has one preference record. `InsertData()` would fail or create a duplicate when a matching primary key already exists, while `UpdateData()` cannot create a missing row.

QUESTION NO: 7

A developer initiated a batch delete of Contacts in Contact Builder due to an import error during implementation. There are over two million records that need to be deleted.

Which two factors should be considered when batch deleting large volumes of contacts?Choose 2 answers

A. Up to one million records can be deleted in each batch.

B. To more quickly remove contact information, use the suppression period length of 14.

C. The deletion process supersedes other automated account activities.

D. The suppression status does not show for individual contacts until the entire batch processes.

ANSWER: A D

Explanation:

“Up to one million records can be deleted in each batch.” is correct because Contact Builder contact deletion is designed to process a maximum of one million contacts in a single deletion request. A deletion involving more than two million records must therefore be planned as multiple batches. This limit is especially important during remediation of an erroneous import: the developer should divide the affected population into appropriately sized, identifiable segments and monitor each deletion request before proceeding with subsequent batches.

“The suppression status does not show for individual contacts until the entire batch processes.” is also correct. When a contact-deletion batch is submitted, Marketing Cloud Engagement processes the request asynchronously. Individual records do not immediately display their resulting suppression status while the job is still in progress. The status becomes visible only after processing of the full batch completes. This affects operational validation and downstream planning, since the team should allow the job to finish before relying on suppression status to confirm that the contacts are no longer eligible for messaging.

Salesforce documents the batch-size constraint and asynchronous processing behavior in its Contact Deletion documentation. Additional implementation guidance is available in the Contact Builder contact deletion help.

QUESTION NO: 8

Which statements are trueregarding the Marketing Cloud SOAP API? Choose 2.

A. More than 2000 SOAP calls can be performed per minute.

B. Most SOAP calls can be synchronous or asynchronous.

C. Uses XML in request and response body.

D. Uses JSON in request and response body.

ANSWER: B C

Explanation:

The statements **“Most SOAP calls can be synchronous or asynchronous.”** and **“Uses XML in request and response body.”** accurately describe the Marketing Cloud Engagement SOAP API. SOAP is an XML-based messaging protocol, so Marketing Cloud SOAP API requests are sent in XML SOAP envelopes and responses are returned as XML SOAP envelopes. Developers use the API’s WSDL to understand the available objects, operations, and XML message structure, then submit those messages to the appropriate SOAP endpoint.

The SOAP API also supports both synchronous and asynchronous processing patterns. A synchronous request returns the operation result directly in the response. For operations that need longer processing, an asynchronous pattern can return a request identifier; the client can then use that identifier to retrieve status or result information through subsequent requests. This enables integrations to work appropriately with both immediate CRUD-style operations and processing that cannot reliably finish within one request-response cycle. Salesforce’s SOAP API documentation describes SOAP messages and the available API operations, while its asynchronous API guidance documents the request-ID and follow-up retrieval pattern. See Salesforce SOAP Web Service API documentation and Salesforce asynchronous API calls documentation.

QUESTION NO: 9

A developer wants to set a variable to use a field from a Sendable Data Extension.

Which two options could be used in an AMPscript block to set the variable as a 'First Name" field from a Sendable Data Extension usedto send the email? Choose 2 answers

A. SET @firstName = [First Name]

B. SET @firstName = %%First Name%%

C. SET @firstName = AttributeValue("First Name")

D. SET @firstName = "First Name"]

ANSWER: B C

Explanation:

`SET @firstName = %%First Name%%` can assign the current send-context value of the `First Name` personalization field to an AMPscript variable. A sendable data extension supplies attribute values for each subscriber or contact during the send, and Marketing Cloud resolves the personalization string against that context. This is useful when the developer wants to reuse the recipient's first name later in the message logic or output.

`SET @firstName = AttributeValue("First Name")` is also valid AMPscript. The `AttributeValue()` function retrieves a value from the current subscriber context by attribute name, including an attribute supplied by the sendable data extension. Assigning that returned value to `@firstName` makes it available for conditional logic, string functions, and later output with `v(@firstName)`. The field name must match the attribute name available to the send context, including its spacing. Salesforce documents `AttributeValue()` as the AMPscript function for obtaining values associated with the current subscriber, while personalization strings provide direct field-value substitution in message content. See the AttributeValue() reference and the AMPscript personalization strings guide.

QUESTION NO: 10

What is the purpose of the IF statement below?

```
%%[
SET @rows = LookupRows('DEName', 'SubscriberKey', _SubscriberKey)
IF RowCount(@rows) > 0 THEN
   SET @row = row(@rows)
   SET @image = Field(@row, 'image')
]%%
   <img src="%%=v(@image)=%%">
%%[ ELSE ]%%
   <img src="image.jpg">
%%[ ENDIF ]%%
```

A. To handle when no row is returned by the LookupRows function

B. To handle when the subscriber is in a held status

C. To handle when images are broken

D. To handle when there are multiple records in the data extension for the subscriber

ANSWER: A

Explanation:

To handle when no row is returned by the LookupRows function is correct. In AMPscript, `LookupRows()` returns a rowset containing the records that match the supplied data extension and lookup criteria. The IF statement tests the number of rows in that returned rowset, generally by using `RowCount()`. This allows the email or landing page to safely branch according to whether matching data exists for the subscriber or other lookup value.

When at least one matching row is available, the code can retrieve the desired field value—such as an image URL—from the first returned row and use it for personalization. When the lookup produces no rows, the conditional logic assigns a standard or fallback image instead. This prevents the content from depending on a field value that does not exist and ensures that a usable image source is still rendered for recipients without a matching data-extension record.

This pattern is a standard defensive AMPscript approach for data-driven content: perform the lookup, evaluate the rowset count, then use personalized data only when a result is present. Salesforce documents `LookupRows()` as returning a rowset and `RowCount()` as returning the number of rows in a rowset. See LookupRows Function and RowCount Function.

QUESTION NO: 11

A developer, who is new to Marketing Cloud, needs to design a landing page for a new customer. They choose to use Server-Side JavaScript (SSJS) due to their extensive knowledge of JavaScript from previous projects.

Which two features would the developer be able to leverage in their Server-Side code? Choose 2 answers

A. Wrapping of AMPscript inSSJS code

B. Direct modification of the DOM

C. External Libraries to extend functionality

D. Include Try/Catch blocks within the code

ANSWER: A D

Explanation:

Wrapping of AMPscript in SSJS code is supported because Marketing Cloud can evaluate AMPscript contained in a string of content from SSJS. In particular, the SSJS `Platform.Function.TreatAsContent()` function processes the supplied content as Marketing Cloud content, enabling AMPscript expressions and personalization to be evaluated before the resulting output is returned. This allows a developer to combine SSJS control logic and data processing with AMPscript capabilities that are useful for Marketing Cloud personalization and content rendering. Salesforce documents this pattern in its TreatAsContent SSJS reference.

Include Try/Catch blocks within the code is also supported. SSJS provides JavaScript-style `try`/`catch` error handling, allowing server-side code to attempt an operation, capture a runtime exception, and respond in a controlled manner. On a CloudPage, this can be used to prevent an unhandled processing failure from ending the page request without useful handling, while allowing the developer to log or present an appropriate response. Salesforce's SSJS documentation includes examples of exception handling and describes SSJS as server-side JavaScript executed in Marketing Cloud contexts such as landing pages: Server-Side JavaScript documentation.

QUESTION NO: 12

A developer wants to create a JavaScript Web Token using a key from Key Management.

What function should the developer use?

A. ContentBlockByKey()

B. GetJWTByKeyName()

C. RegExMatch()

D. GeUWT()

ANSWER: B

Explanation:

`GetJWTByKeyName()` is the Server-Side JavaScript function designed to generate a JSON Web Token (JWT) using a key stored in Marketing Cloud Engagement Key Management. The developer supplies the name of the configured key along with the JWT payload, and the function uses that managed key to sign the resulting token. This avoids embedding a secret or private key directly in SSJS code, which supports safer credential handling and lets key material remain managed centrally in the account. The generated JWT can then be used in an integration flow that requires a signed token, provided the

payload and signing configuration meet the requirements of the receiving service. Salesforce documents this function in its SSJS utility-function reference, including its parameters and usage details. See the GetJWTByKeyName() reference and the Marketing Cloud Engagement Server-Side JavaScript documentation.

QUESTION NO: 13

Adeveloper is working on cross-channel campaign functions for the email team at Northern Trail Outfitters. They are reviewing available APIs for the different Marketing Cloud applications to determine the most appropriate solution for each.

Which application utilizes the REST API?

A. Automation Studio

B. Classic Content

C. Content Builder

ANSWER: C

Explanation:

Content Builder is the Marketing Cloud Engagement application designed to use REST API resources for content and asset management. Its REST endpoints enable a developer to work with the reusable content assets that support coordinated messaging across channels, including creating assets, retrieving asset details, updating existing content, organizing assets in categories, and deleting assets when appropriate. This makes REST a natural fit when campaign processes need to programmatically manage modern content rather than rely solely on the user interface.

The Content Builder REST API is centered on the `/asset/v1` resource family. Developers can authenticate through Marketing Cloud Engagement's OAuth 2.0 token endpoint, then submit authorized REST requests to the tenant-specific REST base URL supplied in the authentication response. Asset payloads identify the asset type and define the content and metadata needed for email and other supported experiences. Salesforce documents the available Content Builder asset operations in its REST API reference, including asset creation and retrieval patterns. See the Marketing Cloud Engagement REST API reference and the Content Builder Assets REST API reference for endpoint and request details.

QUESTION NO: 14

Which of the following statements are correct concerning Populations in Contact Builder? Choose 2.

A. Populations are used to create largesubgroups Contacts.

B. Populations need to be added to an Attribute Group.

C. No more than three Populations should be created.

D. Populations should be used for segmentation

ANSWER: A C

Explanation:

"Populations are used to create largesubgroups Contacts." is correct because a population in Contact Builder is intended to define a large subset of the overall contact model. It lets an organization distinguish a substantial group of contacts based on a shared characteristic or business purpose, such as a regional customer base, a product line, or a distinct organizational audience. This supports managing contact data at scale within Marketing Cloud Engagement.

"No more than three Populations should be created." is also correct. Salesforce recommends limiting an account to three populations. This limit is a platform design recommendation intended to keep the contact model manageable and to avoid unnecessary complexity when working with large contact datasets. A developer should therefore plan populations at a high level and reserve them for meaningful, broad divisions of contacts rather than creating many narrowly defined groups.

For Salesforce's population guidance and limits, see Salesforce Help: Populations. Additional context on organizing contact data in Contact Builder is available in Salesforce Trailhead: Create a Contact Model.

QUESTION NO: 15

In which three ways should a developer optimize a query activity if it is currently timing out? Choose 3

A. Use intrisic functions in the WHERE clause

B. Use SELECT * to include all fields

C. Use Primary key(s) on fields used in joins

D. Use intermediate tables to stage data

E. Only update records that have changed since last execution

ANSWER: C D E

Explanation:

Use Primary key(s) on fields used in joins, use intermediate tables to stage data, and only update records that have changed since last execution are effective approaches for reducing Query Activity execution time. Primary keys support efficient matching of records between data extensions, particularly when the query joins large tables. Structuring joins around key fields reduces the amount of work required to locate corresponding rows.

Intermediate staging data extensions allow a developer to split a large, complex operation into smaller queries. Each step can filter, aggregate, or prepare a narrower dataset before the next step runs, which makes the final query process fewer rows and reduces join complexity. This pattern is especially useful when source data extensions are large or when a query contains several joins and transformations.

Processing only records changed since the prior run implements an incremental-load pattern. Rather than repeatedly scanning and rewriting the entire target population, the query limits work to newly created or modified records, substantially reducing the volume processed during each automation run. Salesforce documents Query Activities and their use in Automation Studio at Salesforce Developer Documentation and provides guidance on data-extension keys at Salesforce Help.

QUESTION NO: 16

Certification Aid uses Marketing Cloud Connect and wants to create a lead capture form on a landing page. When a customer submits the form, a Lead record should be created in Salesforce. Which scripting language can be used for this? Choose 2.

A. AMPscript to create Salesforce record, SSJS for form handling.

B. SSJS to create Salesforce record, AMPscript for form handling.

C. AMPscript for whole functionality.

D. SSJS for whole functionality.

ANSWER: C D

Explanation:

Both "AMPscript for whole functionality." and "SSJS for whole functionality." are valid because Marketing Cloud Connect exposes Salesforce-object functions to both server-side scripting languages on a CloudPages landing page. An AMPscript-based form can obtain submitted field values with request functions such as `RequestParameter()` and then use `CreateSalesforceObject()` to create a Lead in the connected Salesforce organization. This supports handling the submitted values and creating the Salesforce record within the AMPscript implementation.

Server-Side JavaScript can likewise handle the complete workflow. SSJS can read values submitted to the CloudPage and invoke the Platform function `CreateSalesforceObject()` to create the Lead using those values. In either implementation, the connected Salesforce user and Marketing Cloud Connect integration must have the required access to create Lead records, and the submitted values must align with the Lead fields being populated. Salesforce documents the AMPscript Salesforce-object functions, including `CreateSalesforceObject()`, at Salesforce Developer Documentation. The corresponding SSJS Platform function is documented at SSJS Platform Functions.

QUESTION NO: 17

A developer must display the most recently created active coupon for the current subscriber. Multiple active coupon records can exist for the same SubscriberKey in the Coupons data extension.

Which two AMPscript statements should the developer use to identify the most recent coupon record? Choose 2 answers

A. SET @rows = LookupOrderedRows("Coupons", 1, "CreatedDate DESC", "SubscriberKey", @subKey, "IsActive", "1")

B. SET @coupon = Row(@rows, 1)

C. SET @rows = LookupRows("Coupons", "SubscriberKey", @subKey)

D. SET @coupon = Lookup("Coupons", "CouponCode", "SubscriberKey", @subKey)

ANSWER: A B

Explanation:

`LookupOrderedRows()` can return matching rows in a specified order. Setting the count to `1` and ordering by `CreatedDate DESC` returns a rowset containing only the most recently created matching coupon. `Row(@rows, 1)` then retrieves that first row from the returned rowset so its fields can be accessed with functions such as `Field()`.

`LookupRows()` does not provide ordering, so it cannot reliably select the most recent record. `Lookup()` returns a field value rather than an ordered rowset and is unsuitable when several matching records can exist.

QUESTION NO: 18

A company needs to retrieve a large number of rows from a data extension via the API.

Which two solutions would optimize the performance?

Choose 2 answers

A. Use the REST API instead of the SOAP API.

B. Use the AMPscript API functions on a CloudPage.

C. Use the ContinueRequest feature.

D. Use a SimpleFilterPart to retrieve small sets of relevant data.

ANSWER: C D

Explanation:

Using the ContinueRequest feature and using a SimpleFilterPart to retrieve small sets of relevant data optimize SOAP API retrieval of data extension rows. A SOAP RetrieveRequest returns data in pages rather than as one unrestricted response. When additional rows are available, Marketing Cloud returns a request identifier; supplying that identifier as `ContinueRequest` retrieves the next page. This makes it practical to process a large data extension incrementally, avoids attempting to handle the complete result set in one response, and provides a reliable paging pattern for batch processing. Salesforce documents this mechanism in its SOAP API guidance for ContinueRequest.

A `SimpleFilterPart` should be included when the integration needs only a relevant subset of rows. Filtering at retrieval time narrows the result set returned by Marketing Cloud, reducing data transfer, response processing, and the number of continuation calls needed. The filter can target a field and comparison value appropriate to the integration's selection criteria, allowing the application to retrieve and process manageable, meaningful groups of records. Salesforce's SOAP API reference describes the available SimpleFilterPart properties and operators. Together, server-side filtering and continuation-based paging are the appropriate approach for efficient large-volume data extension retrieval.

QUESTION NO: 19

A developer needs to add From Addresses to Marketing Cloud and wants to ensure they are verified before being used for sending.

Which tworoutes would allow this?

Choose 2 answers

A. POST /messaging/v1/domainverification

B. POST /messaging/v1/domainverification/bulk/insert

C. POST /messaging/v1/dataevents/domainverification

D. POST/messaging/v1/push/domain/verification

ANSWER: A B

Explanation:

`POST /messaging/v1/domainverification` and `POST /messaging/v1/domainverification/bulk/insert` are the appropriate REST API routes for initiating domain-verification requests in Marketing Cloud Engagement. Domain verification is part of establishing trusted sender identity for email. A verified sending domain supports the use of addresses on that domain as From Addresses and helps ensure the organization has completed the required ownership-validation process before sending. The single-domain route is appropriate when a developer needs to submit one domain for verification. The bulk-insert route supports submitting multiple domains in one request, which is useful when onboarding several brands, business units, or sender domains at the same time. Both routes belong to the Messaging REST API and use API version `v1`, not `vl`. After the verification request is created, the domain owner must complete the required DNS-based validation steps before the domain can be considered verified for sending. Salesforce's Messaging API reference documents these domain-verification resources and their request patterns. See the Marketing Cloud Engagement Messaging REST API reference and Salesforce guidance on sender authentication.

QUESTION NO: 20

A developer is creating a CloudPage which accepts secure parameters via an email link and will submit those parameters to another CloudPage for data upsert. The page currently captures an Appointment ID parameter passed into it and sets the value to the variable caapptld. The developer does NOT want the Appointment ID to be visible to anyone using the form.

What is the best method to ensure the parameters are passed successfully to the data upsert page?

A. apptId)»%%" readonly>

B.

C.

ANSWER: C

Explanation:

`<form action="%%=CloudPagesURL(123,'apptId',@apptId)=%%" method="post">` is the appropriate approach because `CloudPagesURL()` generates a URL to the destination CloudPage and encrypts the supplied name-value

parameter pair. The target CloudPage can retrieve the Appointment ID using AMPscript, such as `RequestParameter('apptId')`, and use it as part of its controlled upsert process. Encrypting the parameter in the destination URL is important when passing identifiers between CloudPages, because it prevents the raw Appointment ID from being directly exposed in the generated link or form action.

The form's `post` method then submits the form payload in the HTTP request body rather than adding submitted values to the browser's visible query string. Together, an encrypted CloudPages destination URL and a POST submission provide a reliable pattern for carrying the required identifier forward while keeping it out of a visible form field. The destination page ID must be replaced with the actual CloudPage ID, and the parameter name used by `CloudPagesURL()` must match the name retrieved on the upsert page. Salesforce documents this behavior in its CloudPagesURL AMPscript reference and its RequestParameter reference.