DUMPSBOSS.

Salesforce Certified OmniStudio Consultant

Salesforce OmniStudio-Consultant

Version Demo

Total Demo Questions: 25

Total Premium Questions: 251

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, OmniStudio Fundamentals	11
Topic 2, FlexCards	71
Topic 3, OmniScripts	86
Topic 4, Data Tools	40
Topic 5, Integration Procedures	35
Topic 6, Deployment and Testing	3
Topic 7, Mix Questions	5
Total	251

QUESTION NO: 1

A Salesforce Omnistudio Consultant is validating an Omniscript they have developed for a client. The client has specific business requirements that the

script must meet.

What is the best way for the consultant to ensure that the Omniscript meets the client ' s business requirements?

A. Implement the Omniscript without comparing it with the client ' s business requirements.

B. Assume that the Omniscript meets the client's business requirements without conducting a review.

C. Conduct a thorough review of the Omniscript, comparing each step with the client ' s business requirements.

D. Ask the client to review the Omniscript with their internal team.

ANSWER: C

Explanation:

Conduct a thorough review of the Omniscript, comparing each step with the client ' s business requirements. is the best approach because validation must establish traceability between what stakeholders requested and what the implemented guided process actually does. The consultant should walk through each OmniScript step and assess whether its inputs, displayed content, branching logic, validations, calculations, data mappings, actions, and integrations support the documented acceptance criteria. This review should use representative test scenarios, including expected user paths and relevant data conditions, to confirm that the experience produces the required outcomes and handles the business process correctly. Any differences discovered during the review can then be resolved before the solution is presented for stakeholder acceptance or released. This is consistent with Salesforce's guidance to test OmniScripts during development and to use the OmniScript Debugger to inspect execution, data JSON, element behavior, and errors. Comparing the completed configuration directly to requirements also helps ensure that the script is complete rather than merely functional from a technical perspective. After the consultant's internal validation, client review and user acceptance testing can provide additional confirmation that the delivered experience satisfies business needs. See Salesforce's OmniScript debugging documentation and OmniScript Basics on Trailhead.

QUESTION NO: 2

An OmniStudio Consultant receives a request to build a FlexCard that displays the following information:

• Campaign name

• Members of the campaign

• Campaign expiration date

• Expected revenue

A popup window will display additional information about the campaign, such as details about the members. A button must initiate a guided process to add new leads to the campaign.

Which two FlexCard features should the consultant recommend to meet these requirements?

A. Block element

B. OmniScript action

C. DataTable

D. Flyout action

ANSWER: B D

Explanation:

Flyout action and **OmniScript action** meet the two required user interactions. A Flyout action lets a user open contextual content from a FlexCard without leaving the current page. It can present extra campaign information in an overlay-style flyout, including a child FlexCard or other configured content. This supports the need to expose member details and other campaign information in a popup experience while retaining the campaign summary on the original card.

An OmniScript action is designed to launch an OmniScript from a FlexCard. OmniScripts provide a guided, step-based interaction for collecting information, applying business rules, and performing updates through configured actions and integrations. The FlexCard button can therefore start an OmniScript that guides the user through selecting or entering lead information and adding those leads to the campaign. Combining the flyout for on-demand detail viewing with the OmniScript launch action for the lead-addition workflow provides both required behaviors in a reusable FlexCard design. Salesforce's OmniStudio documentation describes FlexCards as configurable UI components that can use actions to invoke interactions, and identifies OmniScripts as guided processes: Salesforce Help: FlexCards and Salesforce Help: OmniScripts.

QUESTION NO: 3

A Data Mapper Load is used to create a Contact. If the LeadSource is not provided in the input, it should default to " Web " .

How should an OmniStudio Consultant implement this default value rule within the Data Mapper?

A. Set the default value using a required validation rule in the Data Mapper.

B. Configure the field-level default value within the Salesforce Object Manager.

C. Enable the fallback value property in the Data Mapper Options tab.

D. Define the value in the Default Value column of the mapping tab.

ANSWER: D

Explanation:

Define the value in the Default Value column of the mapping tab. In a Data Mapper Load, the mapping configuration controls how incoming data is assigned to Salesforce record fields. For the Contact `LeadSource` field mapping, entering `Web` in the mapping row's Default Value column establishes the value that Data Mapper Load uses when the corresponding source value is absent from the input payload. This keeps the defaulting behavior in the OmniStudio Data Mapper itself, exactly where the requirement places it, rather than relying on logic outside the integration procedure or load configuration. The field can still accept a supplied LeadSource value when one is present, while the configured default ensures that records created without that input receive a consistent value. Data Mapper Load is specifically intended to map input data to Salesforce object fields for record creation and updates, and its field-mapping settings support defining constants and default values as part of that transformation. See Salesforce's Data Mapper Load documentation and the Data Mapper overview for Data Mapper configuration concepts.

QUESTION NO: 4

A company wants to create a new digital interaction process that allows customers to request a quote for service from a local retail energy supplier. The process requires the following actions:

• Allow the user to select one or more energy products from a list

• Get current energy usage data from an external system via an API

• Save the data back to Salesforce as a lead

Which three OmniScript elements should the consultant recommend to meet these requirements?

Choose 3 answers

A. DataRaptor Post Action

B. Multi-select Input

C. Post to Object Action

D. Radio Input

E. HTTP Action

ANSWER: A B E

Explanation:

Multi-select Input is appropriate because the customer must be able to choose more than one energy product during the same interaction. The selected values become part of the OmniScript data JSON and can be used later in the flow. **HTTP Action** is the OmniScript action for making an HTTP callout to an external REST-style API, enabling the script to retrieve the customer's current energy-usage information from the external system. **DataRaptor Post Action** is the recommended persistence mechanism when the collected OmniScript data must be mapped and written to Salesforce. A DataRaptor Load configuration can map the OmniScript JSON, including product selections and retrieved usage details, to Lead fields and create or update the Lead record. This approach keeps field mappings declarative and reusable, which is especially useful when the quote-request process grows or the target data model changes. These elements therefore cover the required user input, external integration, and Salesforce record persistence stages of the digital interaction. Salesforce's OmniScript and DataRaptor learning material is available at Trailhead: OmniScripts and DataRaptors, and OmniStudio documentation can be accessed through Salesforce Help: OmniScripts.

QUESTION NO: 5

A business needs a 360° view of their accounts, including a FlexCard to display all of the products sold to the account. The business identified 20 different data elements and 10 actions that users would need when view the product information. Once all of the elements are collected together on the FlexCard, it looks cluttered.

Which two FlexCard features should the consultant recommend to address this issue?

Choose 2 answers

A. Use a table or chart to display data

B. Use a flyout action to display additional information

C. Use a condition to hide data unless specific criteria are met

D. Put specific fields in a collapsible block

ANSWER: B D

Explanation:

Use a flyout action to display additional information and **Put specific fields in a collapsible block** are appropriate FlexCard design features for reducing visual clutter while preserving access to the account-product details users need. A flyout action lets the card retain a concise initial presentation and exposes secondary details only when a user requests them. This supports progressive disclosure: frequently needed summary information remains immediately visible, while less frequently used data and actions can be accessed in context without forcing every element onto the card surface.

Collapsible blocks similarly organize related fields into expandable sections. The card can display the most important product information at a glance while allowing users to expand a section for additional attributes when needed. This is especially useful when many data elements are related but do not all need simultaneous visibility. Together, flyouts and collapsible blocks create a clearer, more navigable product view and help users focus on the information relevant to their current task. Salesforce describes FlexCards as configurable UI components that can present data and actions in a guided, contextual experience; see Salesforce Trailhead: OmniStudio FlexCards and Salesforce Help: FlexCards.

QUESTION NO: 6

which two of the following use cases are best solved using Calculation Procedures & Matrices?

Choose 2 answers

A. To apply the correct factor when determining a cost

B. To return output that is calculated differently based on the date

C. To determine the list of products to display to a customer

D. To retrieve text data and convert it to an integer

ANSWER: A B

Explanation:

Calculation Procedures and Calculation Matrices are designed for configurable, data-driven business calculations and decisioning. **To apply the correct factor when determining a cost** is a core use case because a matrix can store factors, rates, or multipliers by input criteria, while a Calculation Procedure orchestrates the lookup and applies the returned value in a formula. This enables administrators to maintain business values without hard-coding each rule in an integration or custom Apex implementation.

To return output that is calculated differently based on the date is also well suited to these tools. Calculation Matrices support effective dating, allowing the same input criteria to return different values when the calculation date falls within a different configured date range. A Calculation Procedure can pass the relevant date and other input values to the matrix, then use the applicable result in downstream calculation steps. This pattern is especially useful for pricing, fees, taxes, eligibility thresholds, and other policies that change over time while retaining historical values. Salesforce describes Calculation Procedures as a way to sequence calculations and Calculation Matrices as a way to manage and retrieve configurable values for those calculations. See Salesforce Calculation Procedures documentation and Salesforce Calculation Matrices documentation.

QUESTION NO: 7

A business requires a solution to generate an event {platform event) using account information. All the event information is related to the customer and is stored in Salesforce.

Using an Integration Procedure, which two actions are necessary to design this solution?

Choose 2 answers

A. HTTP

B. Response

C. DataRaptor Post

D. DataRaptor Extract

ANSWER: C D

Explanation:

DataRaptor Extract is necessary to retrieve the Account data that supplies the platform event's payload. An Integration Procedure can invoke a DataRaptor Extract to query Salesforce data efficiently and place the selected Account fields into its JSON data structure. Those extracted values can then be mapped directly to the corresponding custom fields on the platform event.

DataRaptor Post is necessary to create the platform event record after the Account information has been retrieved and mapped. Platform events are published by creating event records, and a DataRaptor Post action enables the Integration Procedure to send the assembled field values to a Salesforce object endpoint. Posting a record for the platform event object causes Salesforce to publish the event, making it available to configured platform-event subscribers and downstream automation.

This two-step design keeps data retrieval and record creation declarative within the Integration Procedure: first obtain the customer data, then use it as the event payload when creating the event record. Salesforce describes DataRaptors and their use within OmniStudio processes in the OmniStudio DataRaptors documentation. For the platform-event publishing model, see Publish Platform Events.

QUESTION NO: 8

A company has a customer 360° FlexCard that retrieves account, billing, and service information from several Salesforce objects. The FlexCard is opened frequently by agents, and the underlying data changes only once each day.

What Integration Procedure feature should the consultant configure to reduce unnecessary processing?

A. Chaining

B. Caching

C. Try Catch

D. Batch processing

ANSWER: B

Explanation:

Caching is the appropriate feature because the Integration Procedure can store a response and return the stored result for subsequent requests that use the same cache key. In this scenario, agents frequently request the same customer information, while the data changes only daily. Caching reduces repeated Salesforce queries and server processing, which can improve the responsiveness of the FlexCard.

The cache duration and cache key should be configured so that an agent receives the appropriate account-specific response and the cached data expires when the business requires refreshed information. Caching is not a substitute for an error-handling design or for data transformation; it is intended to reduce repeated processing for data that can safely be reused for a defined period. See the OmniStudio Integration Procedures Trailhead module for Integration Procedure configuration concepts.

QUESTION NO: 9

A company wants to create a new digital interaction process that allows customers to request a quote for service from a local retail energy supplier. The process requires the following actions:

â€¢ Allow the user to select one or more energy products from a list

â€¢ Get current energy usage data from an external system via an API

â€¢ Save the data back to Salesforce as a lead

Which three OmniScript elements should the consultant recommend to meet these requirements?

Choose 3 answers

A. DataRaptor Post Action

B. Multi-select Input

C. Post to Object Action

D. Radio Input

E. HTTP Action

F. DataRaptor Post Action, Multi-select Input, and HTTP Action

ANSWER: F

Explanation:

DataRaptor Post Action, Multi-select Input, and HTTP Action is the correct combined answer because it maps directly to all three required steps in the OmniScript. A Multi-select Input presents a set of selectable choices and allows the customer

to select more than one energy product. This captures the product choices in the OmniScript's data JSON for use by later actions.

An HTTP Action calls an external REST or SOAP endpoint, making it appropriate for retrieving the customer's current energy-usage information from the supplier's external API. The returned payload can be stored in the OmniScript data JSON and used alongside the selected products. Finally, a DataRaptor Post Action writes OmniScript data to Salesforce. Configuring its DataRaptor to map the collected interaction data and retrieved usage information to Lead fields creates or updates the Lead record as required.

These elements separate user input, external integration, and Salesforce persistence while keeping the interaction declarative. Salesforce OmniStudio documentation describes OmniScript actions and inputs at the OmniStudio Developer Guide; see also the OmniStudio Data Tools and Integration Procedures Trailhead module for integration and data-mapping concepts.

QUESTION NO: 10

A business wants to create a reusable OmniScript to capture customer payment information during the order process. The business decides that the first step of the payment process should include:

• Payment type (credit card or bank account)

• Payment amount

Which two elements should the consultant recommend for this step of the process?

Choose 2 answers

A. Number

B. Radio

C. Multi-select

D. Currency

ANSWER: B D

Explanation:

Radio is the appropriate OmniScript input element for payment type because the user must make one mutually exclusive choice: credit card or bank account. A radio control presents the available payment methods clearly and records a single selected value in the OmniScript data JSON. This makes the choice easy to use later in the script for conditional navigation, such as displaying the appropriate collection fields for the selected payment method.

Currency is the appropriate element for payment amount because it is designed to capture monetary values. It provides a payment-specific user experience by formatting the value as currency and supports validation appropriate to an amount entered during an order process. Capturing the amount with a currency-aware control also makes the data more consistent for downstream pricing, payment, and integration processing.

Using these two elements together creates a reusable first payment step that collects both the selected payment method and a properly formatted monetary amount, while retaining structured values in the OmniScript's JSON data model. Salesforce's OmniScript design guidance and element reference are available in the OmniScript Elements documentation and the OmniScript Trailhead module.

QUESTION NO: 11

A company wants to create a new customer buying journey for their website. The buying journey should include the following functionality:

• Allow the user to enter contact and address information

• Require the user to enter age, gender, and optionally income bracket

• Compute a discount percentage per product based on the customer data provided

• Save the list of suggested products including discounts

Which three OmniStudio tools should the consultant use to design a solution that meets these requirements?

Choose 3 answers

A. OmniScript

B. Integration Procedures

C. Calculation Procedures and Matrices

D. FlexCard

E. OmniStudio Action

ANSWER: A B C

Explanation:

OmniScript provides the guided, web-based interaction for this buying journey. It can collect contact and address details, enforce required entries for age and gender, leave income bracket optional, and present the resulting product recommendations to the user. Its step-based structure is appropriate for organizing the data-collection and review flow.

Calculation Procedures and Matrices provide the declarative rules engine needed to determine a discount for each suggested product. A calculation procedure can evaluate the customer inputs and product context, while calculation matrices can maintain discount-rule values for combinations such as age range, gender, income bracket, and product. This separates business discount rules from the user interface so that authorized users can maintain them more easily.

Integration Procedures can orchestrate the server-side work required to persist the suggested-product list and calculated discounts. An Integration Procedure can invoke the required data operations, such as DataRaptor actions, in a single efficient request and return the saved result to the OmniScript. This supports a cohesive guided journey while keeping data-access logic outside the UI. See Salesforce Trailhead's OmniScripts module and Integration Procedures module.

QUESTION NO: 12

A business wants to create a FlexCard for mobile plans to add to their Customer 360° console application. The FlexCard needs to include the following actions:

• Start a process to retrieve plan consumption data

• Create a new case

• Open a promotions web page

• Change the SIM card

which combination should the consultant use in designing the solution?

A. Custom Event and Redirect URL

B. OmniScript and Navigate

C. Event, Navigate and Card

D. Flyout and OmniScript

ANSWER: B

Explanation:

OmniScript and Navigate is the appropriate combination because it covers both guided business processes and destination-based navigation from a FlexCard. An OmniScript action can launch a guided, interactive flow for work that requires user input, validations, orchestration, and backend processing. This makes it suitable for initiating the plan-consumption retrieval process when it involves a defined service workflow, as well as for the SIM-card change process. The same OmniScript can also collect the information necessary to create a case and invoke the required DataRaptor, Integration Procedure, or Salesforce data action to save it.

A Navigate action is designed to send the console user to a specified destination. It can therefore open the promotions web page directly and can also take a user to a Salesforce page or record-oriented destination when needed in the surrounding customer-service experience. Combining these action types keeps the FlexCard concise: it presents the mobile-plan summary and exposes clear entry points, while OmniScript handles multi-step transactional work and Navigate handles redirects. Salesforce describes FlexCards as configurable, contextual UI components with actions, while OmniScripts provide guided interactions and can orchestrate data services. See Salesforce Developer Documentation: FlexCards and Salesforce Developer Documentation: OmniScripts.

QUESTION NO: 13

A company is designing a new console for contact center agents. The cards in the console need to display the following:

• "Open" cases with case description, case open date, case type, assigned to and priority fields. Open should be highlighted with a red border.

• "Awaiting Closure" cases with case description, last action taken date, resolution, approval reason for closure, and assigned to fields. These cases should be highlighted with a grey border.

• "Closed" cases with case description, resolution, case closed date fields with a link to duplicate cases.

All cases will be fetched using a single DataRaptor.

How should the consultant design the FlexCard solution to meet these requirements?

A. Using card session variables and a single FlexCard with multiple flyouts

B. Using card session variables and multiple FlexCards

C. Using card filter and a single FlexCard with multiple flyouts

D. Using card filter and multiple FlexCards

ANSWER: B

Explanation:

Using card session variables and multiple FlexCards is the appropriate design because each case status requires a substantially different card layout, set of displayed fields, styling treatment, and—in the closed-case scenario—an additional duplicate-case link. Separate FlexCards allow each status-specific presentation to be built and maintained independently, including the red and grey borders and the fields that are relevant at that point in the case lifecycle.

A single DataRaptor can retrieve the complete set of case data once. Card session variables can then retain and share the relevant context within the console experience, such as the selected case or status, so the appropriate FlexCard can render from the data already available in the session. This avoids requiring each status-specific card to independently retrieve the same case data. The approach also keeps the console modular: changes to the closed-case duplicate link or the awaiting-closure approval details can be made within the corresponding FlexCard without disrupting the other case presentations.

Salesforce describes FlexCards as reusable UI components for displaying contextual data and supporting interactions. See the OmniStudio FlexCards Trailhead module and the Salesforce Help documentation for OmniStudio FlexCards.

QUESTION NO: 14

In an Integration Procedure, what group element will control whether an individual action executes?

A. Conditional Block

B. Cache Block

C. Try-Catch Block

D. Loop Block

ANSWER: A

Explanation:

Conditional Block is the Integration Procedure group element used to control whether an action executes. A Conditional Block evaluates a configured condition against values available in the Integration Procedure's JSON data. When that condition evaluates to true, the actions nested within the block run; when it does not, those actions are skipped. This enables a single Integration Procedure to branch according to request data, values returned by prior actions, or other data already present in the procedure's execution context.

For example, an Integration Procedure can place a DataRaptor Extract, HTTP Action, Remote Action, or another supported action inside a Conditional Block and execute it only when a relevant field meets a business rule. This keeps decision logic within the server-side orchestration flow and avoids invoking integrations or data operations that are not needed for a particular request. The condition is configured as part of the Conditional Block, while the child actions contain the work to perform when the condition is met. Salesforce's OmniStudio Integration Procedure learning material describes Integration Procedures as declarative, server-side processes that orchestrate actions and use control elements to manage flow behavior. See Salesforce Trailhead: OmniStudio Integration Procedures and Salesforce Help: Integration Procedures.

QUESTION NO: 15

A company needs to create a quoting process for its internal agents. During quoting the agent select a product that is passed to the Omniscript with product details including the unit price, and specifies the grade (A, B, C, D, E)

size (Small, Medium, Large, X-Large). The process should use the grade and size to look up a discount factor, then multiply the unit price by that discount to returnn the quote.

Which two tools should the to meet these

Choose 2 answers

A. Data Mapper Transform

B. Expression Set

C. Decision Matrix

D. Data Mapper Extract

ANSWER: B C

Explanation:

Decision Matrix and **Expression Set** meet the requirement because they separate the configurable lookup logic from the calculation logic. A Decision Matrix is designed to return an output value from combinations of input values. Here, grade and size are the input columns, while the discount factor is the returned value. Administrators can maintain the grade-and-size discount combinations declaratively, without embedding those combinations in OmniScript formulas or custom code.

An Expression Set is appropriate for calculating the resulting quote once the discount factor is available. It can define the reusable expression that multiplies the unit price already present in the OmniScript data JSON by the discount factor returned from the Decision Matrix. The stated scenario explicitly says that the selected product is passed into the OmniScript with its details, including unit price, so the required calculation can use that existing input directly. Together, the matrix provides the business-rule result and the expression applies the arithmetic needed for the quote.

Salesforce documents Decision Matrices as configurable tables that evaluate input values and return outputs, and Expression Sets as declarative expressions used in OmniStudio calculation logic. See Salesforce's Decision Matrix documentation and Expression Set documentation.

QUESTION NO: 16

An OmniStudio Consultant is designing an Integration Procedure (IP) that includes a series of Data Mapper actions. To ensure the IP is executed only when necessary, the consultant needs to implement conditional execution.

What is best practice for adding conditional execution to an IP action?

A. Use a Conditional Block to wrap the data action and define the activation condition based on the IP ' s input data.

B. Configure the IP to run asynchronously, as asynchronous execution handles all conditional logic automatically.

C. Add a custom Apex class at the beginning of the IP to check the condition and manually halt execution if false.

D. Use a Set Values action to set a true/false flag, and then configure the Data Mapper action ' s input to check the flag.

ANSWER: A

Explanation:

Use a Conditional Block to wrap the data action and define the activation condition based on the IP ' s input data. A Conditional Block is the declarative Integration Procedure construct intended to control whether one or more enclosed actions execute. The consultant configures its condition using values available in the Integration Procedure's data JSON, such as an input parameter, a value calculated earlier in the procedure, or a value returned by a prior action. When the condition evaluates as true, the contained Data Mapper action runs; when it does not, the block is skipped and processing continues according to the Integration Procedure's flow. This approach keeps the decision logic visible in the procedure's design, avoids unnecessary data operations, and permits related actions that share the same prerequisite to be grouped under one condition. It is also maintainable because the condition can be reviewed and updated directly in OmniStudio without moving the orchestration decision into custom code. Salesforce documents Conditional Blocks as an Integration Procedure feature for conditionally executing grouped elements, and its OmniStudio learning materials describe using Integration Procedures to orchestrate data retrieval and transformation declaratively. See Salesforce Help: Conditional Blocks in Integration Procedures and Trailhead: OmniStudio Integration Procedures.

QUESTION NO: 17

A consultant must design a 360 view of the customer. The business requirements are:

â€¢ A header card with account information (name, account number, next billing date, invoice method)

â€¢ A list of related contacts (first name, last name, phone)

â€¢ All the open cases related to the account (subject, priority, SLA)

An account will not have more than 2 contacts, but it could have more than 10 open cases. It is necessary to the different sections available at a glance.

Which two FlexCard features should the consultant recommend to improve the user experience?

Choose 2 answers

A. Use a Datatable element

B. Use a Block Element with the Collapse property enabled

C. Use a Zone Template

D. Use a Custom Style to adjust height and width

E. Use a Block Element with the Collapse property enabled and a Zone Template

ANSWER: E

Explanation:

Use a Block Element with the Collapse property enabled and a Zone Template. A Zone Template provides the structural layout needed for a customer 360 view: it organizes the account summary, the small related-contact section, and the potentially large open-case section into distinct, visually recognizable areas. This lets users immediately identify the information categories and scan the account-level details without losing the overall context of the page.

A collapsible Block Element is particularly appropriate for the open-cases area because that collection can exceed ten records. The block can retain a visible section heading and an expand/collapse control, so users know that open-case information is available while avoiding an unnecessarily tall card when the full case list is not needed. Users can expand that section when they need subjects, priorities, and SLA values, while the header and other key sections remain easy to access. Together, these features balance information density with discoverability and support a practical, readable FlexCard design. Salesforce's FlexCards learning material describes arranging data using card elements and templates; see Salesforce Trailhead: OmniStudio FlexCards and Salesforce Industries Developer Documentation.

QUESTION NO: 18

An existing OmniScript used to capture and update customer information displays the following information on the same page:

• Name

• Age

• Street

• Last name

• Postal code

• Gender

• State

• Phone

• City

• Country

• Email

Users report that the information displays in no specific order. Users experience errors when filling the information because it is not categorized. The process fails after submission because the mandatory fields are blank.

How can the consultant improve the user experience of the OmniScript?

A. Use a template for each category and add custom code for the required validation

B. Use a Section to break the information and Headline to the title and a required icon in the input

C. Use a Visualforce Page with the categories, labels, and the required validation

D. Use Block elements to group and required property checked in some of the inputs

ANSWER: D

Explanation:

Use Block elements to group and required property checked in some of the inputs is correct because Blocks are OmniScript structural elements designed to organize related fields into logical groups. For this customer-information flow, the consultant can place personal details such as name, age, gender, phone, and email in one Block, and address details such as street, city, state, postal code, and country in another. This creates a predictable visual layout and makes the form easier for users to scan and complete accurately.

Each input that must be supplied for the update process should also have its Required property enabled. OmniScript then validates those fields as part of normal client-side navigation and submission behavior, helping prevent users from continuing with missing mandatory values. Grouping fields with Blocks and applying declarative required validation directly to the relevant inputs uses standard OmniScript capabilities, avoids unnecessary custom development, and improves both usability and data quality. Salesforce describes OmniScript elements and their configuration in the OmniScript Elements documentation and provides guided implementation learning in the Build OmniScripts Trailhead module.

QUESTION NO: 19

Which OmniStudio tool creates a Chatter post and sends to a Chatter feed?

A. DataRaptor Load

B. Integration Procedure

C. Calculation Procedure

D. FlexCards

ANSWER: B

Explanation:

Integration Procedure is correct because it can orchestrate server-side actions and includes an action for posting a message to Chatter. The Chatter Post Action within an Integration Procedure can create a feed item and direct it to the appropriate Chatter feed, such as a record feed, a user's profile feed, or a group feed. This makes it useful when a guided OmniStudio process must notify users, document an event, or initiate collaboration as part of its automated workflow.

An Integration Procedure runs a sequence of declarative elements, can accept input from an OmniScript or FlexCard, can call other services or DataRaptors, and can return structured output. The Chatter-posting capability fits this orchestration role: the procedure can first collect or transform relevant data and then publish a message using that data without requiring custom Apex solely to create the Chatter post. Salesforce's OmniStudio learning material describes Integration Procedures as declarative, server-side processes that execute actions and integrations, while Salesforce Chatter documentation explains feed posts as collaboration records delivered to feeds. See Salesforce Trailhead: OmniStudio Integration Procedures and Salesforce Help: Chatter.

QUESTION NO: 20

A business needs to display installed products for field service technicians on service calls using a mobile device The installed product information must be summarized so the technician can seekey details at a glance. How the technician also needs to sometimes access a list of past service dates for each product.

Which two FlexCards features should the consultant recommend to meet this requirement?

Choose 2 answers

A. Use flyouts

B. Use card states

C. Enable the Responsive property

D. Customize the styling

ANSWER: A C

Explanation:

Use flyouts because they let a FlexCard retain a concise, glanceable presentation while exposing related detail only when a technician requests it. For example, the installed-product card can show the product's key fields in its primary layout and provide a flyout action that opens a contextual list of prior service dates for that specific product. This keeps the service-call

experience focused without requiring the user to leave the current card or crowd its summary with historical records. Salesforce describes FlexCards as configurable UI components that can display contextual information and invoke actions; see Get Started with OmniStudio FlexCards.

Enable the Responsive property so the card can adapt its layout to the limited screen space of the mobile device used in the field. A responsive FlexCard can reorganize or resize its displayed elements at different breakpoints, preserving readability and usability for technicians viewing installed-product summaries on phones or tablets. This directly supports the requirement that the summarized information be useful at a glance during a mobile service call, while the flyout supplies the occasional deeper history view. Salesforce's OmniStudio FlexCards documentation and learning material cover responsive card design and layout behavior; see Build an OmniStudio FlexCard.

QUESTION NO: 21

An OmniScript captures a customer's first name and last name in separate input elements. The consultant needs to create a value in the OmniScript data JSON that combines the two values into a full name for use later in the process.

Which OmniScript element should the consultant use?

A. Set Values

B. Text Block

C. DataRaptor Load

D. Messaging

ANSWER: A

Explanation:

Set Values is the appropriate OmniScript element because it creates or updates values in the OmniScript data JSON. The consultant can configure a Set Values element with a formula that references the first name and last name inputs and stores the concatenated result in a FullName node.

This keeps the data manipulation declarative and makes the derived value available to later elements, actions, DataRaptors, and Integration Procedures. A Text Block only displays information, while a DataRaptor is not required when the value can be derived directly from data already present in the OmniScript. See the OmniScript Elements documentation for details about OmniScript elements and data JSON behavior.

QUESTION NO: 22

A Salesforce Omnistudio Consultant is tasked with creating an Omniscript for a client ' s customer service process. The consultant wants to integrate Salesforce Knowledge into the Omniscript.

What is one way the consultant can configure an Omniscript to search for Salesforce Knowledge articles?

A. Using a simple search ..

B. Using a complex search algorithm

C. Based on dynamic inputs from Omniscript fields

D. Based on static values set up during configuration

ANSWER: C

Explanation:

Based on dynamic inputs from Omniscript fields is correct because an OmniScript can pass values collected during the user interaction into its configured actions. For a customer-service flow, the consultant can use field values such as a case subject, product, issue category, keyword, or customer-entered question as dynamic search criteria. This lets the Knowledge

search respond to the specific context of the current transaction rather than always retrieving articles for a fixed term. Dynamic values are typically referenced from the OmniScript's data JSON, allowing the search configuration to use information supplied or derived earlier in the script. The resulting relevant Knowledge articles can then be presented to the agent or customer as part of the guided process, helping resolve the issue without leaving the OmniScript experience. This approach also makes a single reusable OmniScript suitable for many service scenarios because its article search adapts to each user's inputs. Salesforce documents OmniScript elements and their use of data from the script's JSON in the OmniScript Elements reference. For the underlying article repository and search capabilities, see Salesforce's Salesforce Knowledge overview.

QUESTION NO: 23

A company is designing an Integration Procedure that retrieves order information and, only for orders with a financing request, invokes an external credit service. If the credit service fails, the procedure must continue and return a controlled result to the calling OmniScript.

Which two Integration Procedure features should the consultant use?

Choose 2 answers

A. Conditional Block

B. Try Catch

C. DataRaptor Turbo Extract

D. Calculation Matrix

E. Text Block

ANSWER: A B

Explanation:

Conditional Block is appropriate because it allows the Integration Procedure to execute the credit-service action only when the order data indicates that financing was requested. This prevents unnecessary external calls for orders that do not require a credit check.

Try Catch is appropriate for handling an error returned by the external credit service. The Try block can contain the HTTP action, while the Catch block can set an error response, invoke alternate processing, or populate a controlled message for the OmniScript. Together, these features provide conditional execution and resilient error handling without requiring custom Apex. See Salesforce Trailhead's OmniStudio Integration Procedures module.

QUESTION NO: 24

When designing OmniScripts, which three best practices should consultants recommend to increase user adoption?

Choose 3 answers

A. Prefill data for users when possible

B. Replicate existing processes as-is

C. Provide keystroke commands for data entry

D. Divide complex processes into sections

E. Provide user help text

ANSWER: A D E

Explanation:

Prefill data for users when possible, **Divide complex processes into sections**, and **Provide user help text** are the recommended design practices because they make a guided OmniScript faster, clearer, and less error-prone for the person completing it. Prefilling values already known by Salesforce or retrieved through DataRaptors, Integration Procedures, or other sources minimizes repetitive typing and helps users focus only on information that requires their input. This also supports data quality by avoiding unnecessary re-entry of existing customer or case information.

Complex business processes should be organized into logical sections, commonly using steps and grouped elements. A progressive, structured experience reduces cognitive load, makes the current task easier to understand, and lets users move through a long interaction in manageable stages. Clear help text supplies contextual guidance at the exact point where a user needs it, such as describing required information, expected formats, or business terminology. Together, these practices support OmniStudio's goal of providing intuitive, guided digital interactions that improve completion rates and user confidence. Salesforce's OmniScript guidance and training resources describe building guided workflows with reusable UI elements, data prefill, and user-focused interaction design: Salesforce Help: OmniScripts and Trailhead: OmniStudio Basics.

QUESTION NO: 25

In which two cases should an integration procedure be used as a data source for FlexCards and OmniScripts,

Choose 2 answers

A. To achieve elastic scaling

B. To retrieve multiple data sources in a single response

C. To minimize the number of elements to be configured

D. To separate the user interface from changes in the data sources

ANSWER: B D

Explanation:

Integration Procedures are appropriate when a FlexCard or OmniScript needs a consolidated response assembled from multiple sources. An Integration Procedure can orchestrate several server-side actions, such as DataRaptor Extracts, Apex actions, HTTP actions, and additional Integration Procedures, then return a single structured JSON payload to the calling UI component. This reduces the need for the client to coordinate separate requests and gives the FlexCard or OmniScript a purpose-built response format. Salesforce describes Integration Procedures as server-side processes for retrieving, manipulating, and saving data from multiple sources; see Salesforce Trailhead: Integration Procedures.

They are also well suited to separating the user interface from changes in underlying data sources. The FlexCard or OmniScript calls a stable Integration Procedure interface rather than directly depending on each object, API, or external system. The procedure can contain the integration and transformation logic, including response mapping, so underlying source changes can often be addressed there while preserving the payload expected by the UI. This supports reusable, maintainable OmniStudio designs and keeps presentation configuration focused on displaying or collecting data. For related OmniStudio architecture and component guidance, review Salesforce Trailhead: OmniStudio Architecture.