Salesforce.Integration-Architect.v2026-04-20.q55

Exam Code:	Integration-Architect
Exam Name:	Salesforce Certified Integration Architect
Certification Provider:	Salesforce
Free Question Number:	55
Version:	v2026-04-20
# of views:	166
# of Questions views:	550
https://www.freepdfdumps.com/Salesforce.Integration-Architect.v2026-04-20.q55.html	

NEW QUESTION: 1
UC has an API-led architecture with three tiers. Requirement: return data to systems of engagement (mobile, web, Salesforce) in different formats and enforce different security protocols. What should the architect recommend?
A. Implement an API Gateway that all systems of engagement must interface with first.
B. Enforce separate security protocols and return formats at the first tier of the API-led architecture.
Answer: B (LEAVE A REPLY)
In a standard API-led connectivity model, the First Tier (Experience APIs) is responsible for tailoring data for specific systems of engagement.
The Experience APIs take the core data from the lower tiers and transform it into the specific return formats (e.g., JSON for mobile, XML for legacy web) and security protocols (e.g., OAuth for Salesforce, API Keys for web) required by each consumer. Option B correctly identifies that these transformations and security enforcements should happen at this outer layer. While an API Gateway (Option A) can provide generic security and rate limiting, it is the Experience API layer that provides the functional transformation and specific protocol requirements defined by the business needs of the engagement systems.

NEW QUESTION: 2
A company needs to integrate a legacy on-premise application that can only support SOAP API. The integration architect determines that the Fire and Forget integration pattern is most appropriate for sending data from Salesforce to the external application and getting a response back in a strongly-typed format. Which integration capabilities should be used?
A. Platform Events for Salesforce to Legacy System direction and SOAP API using Enterprise WSDL for the communication back from legacy system to Salesforce

B. Outbound Messaging for Salesforce to Legacy System direction and SOAP API using Partner Web Services Description Language (WSDL) for the communication back from legacy system to Salesforce
C. Outbound Messaging for Salesforce to Legacy System direction and SOAP API using Enterprise WSDL for the communication back from legacy system to Salesforce
Answer: C (LEAVE A REPLY)
For an outbound, declarative, Fire-and-Forget integration to a legacy SOAP-based system, Salesforce Outbound Messaging is the native tool of choice. Outbound Messaging sends an XML message to a designated endpoint when specific criteria are met. It is highly reliable as Salesforce will automatically retry the delivery for up to 24 hours if the target system is unavailable.
For the communication back from the legacy system to Salesforce, a strongly-typed SOAP API approach is required. The Enterprise WSDL is the correct recommendation here because it is a strongly-typed WSDL that is specific to the organization's unique data model (including custom objects and fields). Using the Enterprise WSDL allows the legacy system to communicate with Salesforce using specific data types, providing compile-time safety and reducing errors during the mapping process.
Option A is less efficient because Platform Events would likely require middleware to translate the event into the legacy system's SOAP format. Option B suggests the Partner WSDL, which is loosely-typed and designed for developers building tools that must work across many different Salesforce orgs. Since this is an internal integration for a specific company, the Enterprise WSDL provides a much more streamlined development experience with better data integrity. By combining Outbound Messaging (for fire-and-forget delivery) and the Enterprise WSDL (for the strongly-typed callback), the architect fulfills the technical requirements while minimizing custom code.

NEW QUESTION: 3
What should an integration architect recommend to ensure all integrations to the Northern Trail Outfitters' company portal use SSL mutual authentication?
A. Generate a certification authority (CA) signed certificate.
B. Enable My Domain and SSL/TLS.
C. Enforce SSL/TLS Mutual Authentication.
Answer: C (LEAVE A REPLY)
To ensure that all integrations calling into a Salesforce portal are secured with Mutual Authentication, the architect must enable and configure specific platform-level security settings. The primary recommendation is to Enforce SSL/TLS Mutual Authentication for the relevant integration users.
Mutual Authentication (Two-way SSL) adds a layer of trust beyond the standard session-based authentication.

When enforced, the Salesforce server requires the calling client to present a valid CA-signed certificate that matches a certificate stored in the org. This ensures that only authorized systems with the correct private key can establish a connection.
To implement this, the architect must first work with Salesforce support to enable the feature. Once enabled, a Mutual Authentication Certificate must be uploaded to the org, and a specific user profile-cloned for integration purposes-must have the "Enforce SSL/TLS Mutual Authentication" permission enabled. This configuration forces the client to use port 8443 (the dedicated port for mutual TLS) for API calls, providing a highly secure, server-to-server connection that protects against impersonation and unauthorized data access.

NEW QUESTION: 4
A global financial company with a core banking system processing 1 million transactions per day wants to build a community portal. Customers need to review their bank account details and transactions. What should an integration architect recommend to enable community users to view their financial transactions?
A. Use Salesforce Connect to display the financial transactions as an external object.
B. Use Iframe to display core banking financial transactions data in the customer community.
Answer: A (LEAVE A REPLY)
When dealing with high-volume data (1 million transactions per day) that does not need to be stored natively in Salesforce, the architect should recommend Data Virtualization via Salesforce Connect.
Salesforce Connect allows the company to display external data as External Objects. This approach provides several architectural advantages for a banking community:
* No Data Storage: Transactions remain in the core banking system, avoiding the massive storage costs and complex synchronization logic required to house millions of records natively in Salesforce.
* Real-Time Visibility: Because External Objects are queried on-demand via the OData protocol or a custom Apex adapter, customers see the most up-to-date transaction history every time they refresh the page.
While an Iframe (Option B) is technically possible, it is often discouraged due to security concerns (such as clickjacking) and a poor user experience, as the Iframe does not natively integrate with Salesforce UI components or reporting. Salesforce Connect provides a "seamless" look and feel, allowing External Objects to be used in related lists and Lightning components just like standard Salesforce records, while keeping the heavy data burden on the performant core banking system.

NEW QUESTION: 5
A company uses Customer Community for course registration. The payment gateway takes more than 30 seconds to process transactions. Students want results in real time to retry if errors occur. What is the recommended integration approach?
A. Use Request and Reply to make an API call to the payment gateway.
B. Use Platform Events to process payment to the payment gateway.

C. Use Continuation to process payment to the payment gateway

Answer: (SHOW ANSWER)

Standard synchronous Apex callouts have a timeout limit, and more importantly, Salesforce limits the number of long-running requests (those lasting longer than 5 seconds) that can execute concurrently. If a payment gateway consistently takes 30 seconds, a few simultaneous users could easily exhaust the org's concurrent request limit, causing the entire system to stop responding.

The Continuation pattern (Option C) is designed specifically for this "long-wait" scenario. It allows the Apex request to be suspended while waiting for the external service to respond, freeing up the Salesforce worker thread to handle other users. Once the gateway responds, the suspended process resumes and returns the result to the student's browser. This provides the "real-time" experience required for the student to retry immediately without the risk of bringing down the entire community due to thread exhaustion.

NEW QUESTION: 6

What is the first thing an integration architect should validate if a callout from a Lightning web component (LWC) to an external endpoint is failing?

A. The endpoint domain has been added to Cross-Origin Resource Sharing (CORS).

B. The endpoint URL has been added to Remote Site Settings.

C. The endpoint URL has been added to Content Security Policies (CSP).

Answer: (SHOW ANSWER)

When an integration initiated from the client-side (the browser) fails, the architect must first look at the browser's security policies. In Salesforce, Lightning Web Components are subject to the Lightning Component framework's Content Security Policy (CSP).

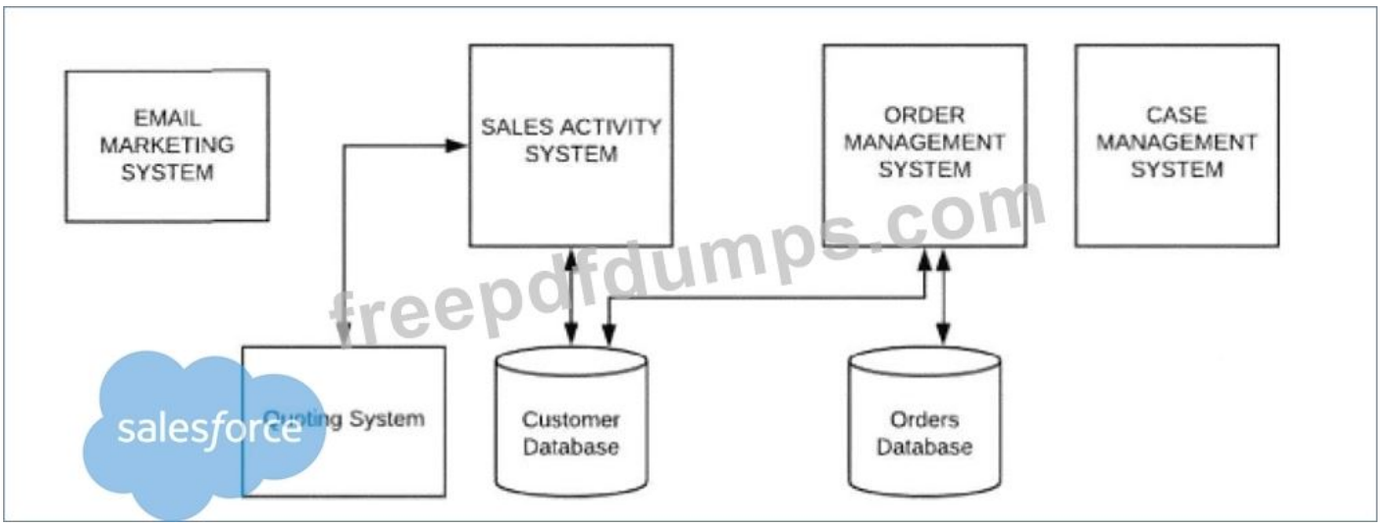
CSP is a security layer that prevents cross-site scripting (XSS) and other code injection attacks by restricting which domains the browser is allowed to communicate with. If an LWC attempts to make a fetch() call to an external REST endpoint, the browser will block the request unless that specific domain is whitelisted in CSP Trusted Sites.

Option B (Remote Site Settings) is a common distractor; these settings are strictly for server-side Apex callouts and have no effect on client-side JavaScript requests. Option A (CORS) is also a browser security mechanism, but it must be configured on the external server to allow Salesforce to access its resources.

While CORS is necessary, the first thing to validate within the Salesforce environment for a failing LWC callout is the CSP Trusted Site entry. Without this whitelisting, the request will be terminated by the browser before it even leaves the client, regardless of how the external server is configured.

NEW QUESTION: 7

A large business-to-consumer (B2C) customer is planning to implement Salesforce CRM to become a customer-centric enterprise. Below is the B2C customer's current system landscape diagram.



The goals for implementing Salesforce include:

* Develop a 360-degree view of the customer.
* Leverage Salesforce capabilities for marketing, sales, and service processes.
* Reuse Enterprise capabilities built for quoting and order management processes.

Which three systems from the current system landscape can be retired with the implementation of Salesforce?

A. Email Marketing, Sales Activity, and Case Management

B. Sales Activity, Order Management, and Case Management

C. Order Management, Case Management, and Email Marketing

Answer: A (LEAVE A REPLY)

In the framework of a Salesforce Platform Integration Architect's landscape evaluation, the primary goal is to determine the "system of record" for each business function and identify redundancies between legacy systems and the proposed Salesforce architecture. This process is driven by the alignment of Salesforce's native "Customer 360" capabilities with the specific goals defined by the enterprise stakeholders.

According to Goal 2, the customer intends to leverage Salesforce specifically for marketing, sales, and service processes. Within the standard Salesforce ecosystem, these domains are addressed by the three core cloud products:

* Marketing Cloud provides the capabilities found in the legacy Email Marketing System.
* Sales Cloud replaces the functions of the Sales Activity System.
* Service Cloud is the native replacement for the Case Management System.

By migrating these three domains to a single platform, the organization directly fulfills Goal 1- developing a

360-degree view of the customer. Consolidating these interactions onto the Salesforce platform allows for a unified data model where customer behaviors in marketing, sales, and support are visible in one place, eliminating the silos inherent in the previous landscape.

However, a critical constraint is presented in Goal 3, which explicitly mandates the reuse of existing enterprise capabilities for quoting and order management. In an integration architecture, this signals that the Quoting System and Order Management System (OMS) are designated as external systems of record that must remain active. These systems often contain complex logic,

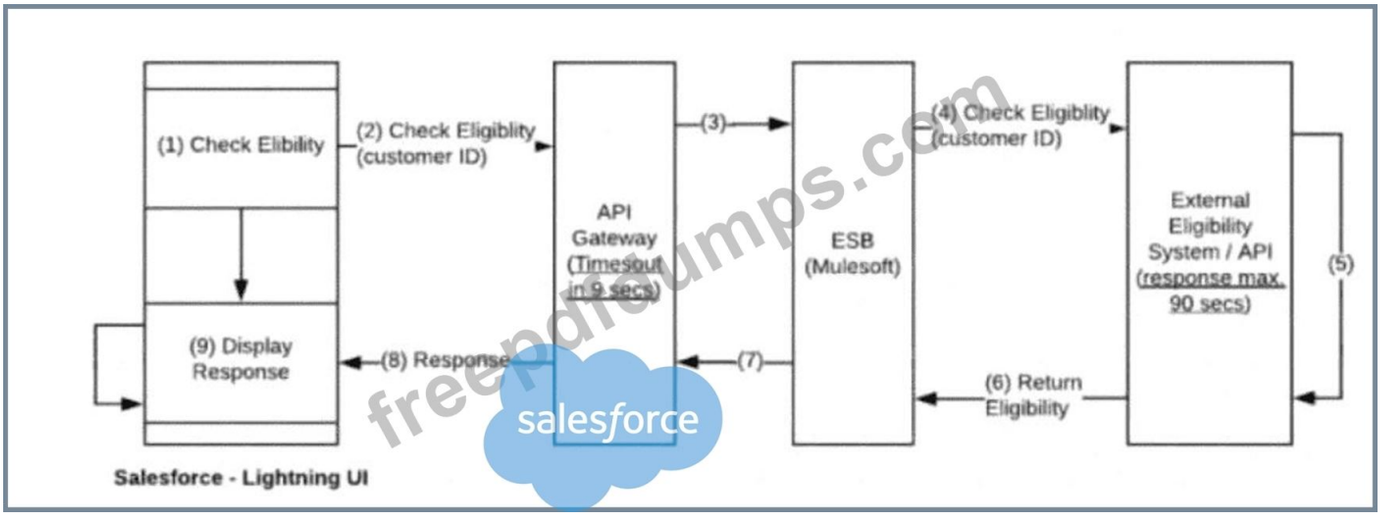
tax calculations, or supply chain integrations (such as with an SAP Business Suite) that the business is not currently ready to migrate.
Therefore, since the Quoting and Order Management systems must be retained, they are excluded from the retirement list. The remaining three systems-Email Marketing, Sales Activity, and Case Management- overlap with Salesforce's native strengths and are not protected by the "reuse" requirement. Retiring them streamlines the technology stack and allows the architect to focus on building robust integration patterns (such as REST or SOAP callouts) to connect Salesforce to the retained Quoting and Order Management systems.

NEW QUESTION: 8
A subscription-based media company's system landscape forces many subscribers to maintain multiple accounts and to log in more than once. An Identity and Access Management (IAM) system, which supports SAML and OpenId, was recently implemented to improve the subscriber experience through self-registration and single sign-on (SSO). The IAM system must integrate with Salesforce to give new self-service customers instant access to Salesforce Community Cloud.
Which requirement should Salesforce Community Cloud support for self-registration and SSO?
A. SAML SSO and Registration Handler
B. OpenId Connect Authentication Provider and JIT provisioning
C. SAML SSO and Just-in-Time (JIT) provisioning
Answer: B (LEAVE A REPLY)

NEW QUESTION: 9
A customer is migrating from an old legacy system to Salesforce. As part of the modernization effort, the customer would like to integrate all existing systems that currently work with its legacy application with Salesforce. Which constraint/pain-point should an integration architect consider when choosing the integration pattern/mechanism?
A. Reporting and usability requirements
B. Data volume and processing volume
C. Multi-language and multi-currency requirement
Answer: B (LEAVE A REPLY)
When migrating from a legacy environment to a multi-tenant cloud platform like Salesforce, Data volume and processing volume represent the most critical technical constraints. Legacy systems often operate without the strict governor limits found in Salesforce, meaning they may push large datasets or high- frequency updates that could easily overwhelm standard Salesforce APIs.
An integration architect must evaluate these volumes to determine the appropriate integration pattern:
* Pattern Selection: If the daily volume involves millions of records, the architect must recommend the Bulk API rather than standard REST or SOAP APIs to avoid hitting daily API limits.

* Synchronous vs. Asynchronous: High processing volumes often necessitate asynchronous patterns (such as Fire-and-Forget or Batch) to prevent user-interface lag and "Concurrent Request Limit" errors.
* Data Virtualization: If the legacy data volume is massive but only needs to be viewed occasionally, the architect might consider Salesforce Connect to avoid consuming expensive internal data storage.
While reporting (Option A) and multi-currency (Option C) are important functional requirements, they do not fundamentally dictate the technical "plumbing" or scalability of the integration architecture. By prioritizing the analysis of volume and processing needs, the architect ensures the new modernization effort is stable, performant, and capable of scaling as the business grows within the bounds of the Salesforce platform.

NEW QUESTION: 10
An enterprise architect has requested the Salesforce integration architect to review the following (see diagram and description) and provide recommendations after carefully considering all constraints of the enterprise systems and Salesforce Platform limits.
* About 3,000 phone sales agents use a Salesforce Lightning user interface (UI) concurrently to check eligibility of a customer for a qualifying offer.
* There are multiple eligibility systems that provide this service and are hosted externally.
* However, their current response times could take up to 90 seconds to process and return (there are discussions to reduce the response times in the future, but no commitments are made).
* These eligibility systems can be accessed through APIs orchestrated via ESB (MuleSoft).
* All requests from Salesforce will have to traverse through the customer's API Gateway layer, and the API Gateway imposes a constraint of timing out requests after 9 seconds.



Which recommendation should the integration architect make?
A. Recommend synchronous Apex callouts from Lightning UI to External Systems via Mule and implement polling on an API Gateway timeout.
B. Use Continuation callouts to make the eligibility check request from Salesforce Lightning UI at page load.

C. Create a platform event in Salesforce via Remote Call-In and use the empAPI in the Lightning UI to serve 3,000 concurrent users when responses are received by Mule.

Answer: C (LEAVE A REPLY)

The primary architectural challenge in this scenario is the massive discrepancy between the backend response time (up to 90 seconds) and the API Gateway timeout constraint (9 seconds). In any synchronous integration pattern, the connection must remain open across the entire path; if the API Gateway closes the connection at 9 seconds, a standard Salesforce "Request-Reply" callout will fail long before the 90-second eligibility check is complete.

Option A is non-viable because synchronous polling at a high scale (3,000 concurrent users) would likely hit Salesforce concurrent request limits and place an immense, unnecessary load on the API Gateway. Option B, using Continuation, is designed to handle long-running callouts (up to 120 seconds) without blocking Salesforce threads, but it still requires the external connection path to remain open. It does not bypass the 9- second timeout imposed by the customer's API Gateway.

The optimal recommendation is Option C, which implements an Asynchronous Request-Reply pattern using Platform Events and the empAPI.12

* Request Phase: The Salesforce UI initiates the request. To bypass the 9-second gateway timeout, the ESB (MuleSoft) should be configured to receive the request3 and immediately return an acknowledgment (e.g.,4 HTTP 202 Accepted). This allows the initial Salesforce callout to complete successfully within the 9-second window.56
* Processing Phase: MuleSoft then proceeds with the long-running (up to 90 seconds) call to the external eligibility systems.78
* Callback Phase (Remote Call-In)9: Once the eligibility result is received, MuleSoft calls back into Salesforce via the REST API to publish a Platform Event containing the result.10
* UI Update (empA11PI): The 3,000 sales agents' browsers, having subscribed to the event channel using the empAPI (Lightning's built-in library for streaming events), receive the notification in real- time. The UI then updates to display the "Display Response" step.

This event-driven architecture effectively "insulates" Salesforce and the API Gateway from the backend's high latency, ensures scalability for 3,000 concurrent users, and provides a seamless, real-time user experience without hitting governor limits or timeout constraints.

NEW QUESTION: 11

Northern Trail Outfitters is creating a distributable Salesforce package. The package needs to call into a custom Apex REST endpoint in the central org. The security team wants to ensure a specific integration account is used in the central org that they will authorize after installation of the package. Which item should an architect recommend to secure the integration?

A. Use an encrypted field to store the password.

B. Create a connected app in the central org and add the callback URL for each org in the package it is installed in to redirect after a successful authentication.

C. Contact Salesforce Support and create a case to temporarily enable API access for managed packages.

Answer: (SHOW ANSWER)
To securely integrate a distributed package with a central "Hub" org, the architect should utilize the OAuth
2.0 Web Server Flow. In this model, a Connected App is created in the central org to act as the identity and access provider.1 A critical component of this setup is the Callback URL (Redirect URI). When a user in the "Subscriber" org (where the packag2e is installed) initiates the connection, Salesforce redirects them to the central org to authorize the access. After successful authentication, the central org needs the correct callback URL to return the authorization code to the specific subscriber org.
Using this flow satisfies the security team's requirement for a specific integration account. The administrator in the central org can pre-authorize specific profiles or permission sets to use the Connected App, ensuring that only the designated integration user's credentials are used to fulfill requests. Option A is a security "anti- pattern" (storing passwords in fields), and Option C is unnecessary as API access is a standard feature. This OAuth-based approach provides a secure, revocable, and standardized way to manage cross-org communication in a multi-tenant environment.

NEW QUESTION: 12
A customer is evaluating the Platform Events solution and would like help in comparing/contrasting it with Outbound Messaging for real-time/near-real time needs. They expect 3,000 customers to view messages in Salesforce. What should be evaluated and highlighted when deciding between the solutions?12
A. In both Platform Events and Outbound Messaging, the event messages are retried by and delivered in sequence, and 3only once. Sales4force ensures there is no duplicate message delivery.
B. Message sequence is possible in Outbound Messaging, but not guaranteed with Platform Events. Both offer very high reliability. Fault handling and recovery are fully handled by Salesforce.
C. Both Platform Events and Outbound Messaging are highly scalable. However, unlike Outbound Messaging, only Platform Events have Event Delivery and Event Publishing limits to be considered.
Answer: C (LEAVE A REPLY)
When comparing Platform Events and Outbound Messaging for a near-real-time architecture, a Salesforce Platform Integration Architect must evaluate fundamental differences in their delivery models and governance. While both provide declarative, asynchronous "Fire-and-Forget" capabilities, their technical constraints differ significantly, particularly regarding scalability and platform limits.
The key architectural highlight in this scenario is that Platform Events operate on a specialized event bus with specific Event Publishing and Event Delivery limits. Unlike Outbound Messaging, which is governed by more general daily outbound call limits (often tied to user licenses), Platform Events have a dedicated allocation for the number of events that can be published per hour and

delivered in a 24-hour period to external clients via the Pub/Sub API or CometD. For example, the number of concurrent subscribers to a Platform Event channel is typically capped at 2,000 for standard configurations. Since the customer expects
3,000 customers to view these messages, this limit is a critical evaluation point; the architecture would need to account for this gap, perhaps by using middleware to fan out messages to the larger audience.
In contrast, Outbound Messaging does not have an "Event Delivery" limit in the same sense. It is a point-to- point SOAP-based push mechanism where Salesforce manages retries for up to 24 hours if the receiving endpoint is unavailable. However, it is less flexible for multi-consumer scenarios because it requires a separate configuration for every unique destination.
Regarding the other options: Option A is incorrect because neither system strictly guarantees "exactly-once" delivery without the possibility of duplicates; in fact, Outbound Messaging may deliver a message more than once if it doesn't receive a timely acknowledgment. Option B is incorrect because Platform Events do not have built-in "fault recovery" handled by Salesforce in the same way as Outbound Messaging's automatic retry queue; with Platform Events, it is the subscriber's responsibility to use a Replay ID to retrieve missed events within the 72-hour retention window. Therefore, highlighting the unique delivery and publishing limits is the most vital step for the architect.

NEW QUESTION: 13
Northern Trail Outfitters is creating a distributable Salesforce package. The package needs to call into a Custom Apex REST endpoint in the central org. The security team wants to ensure a specific integration account is used in the central org that they will authorize after installation.
Which item should an architect recommend?
A. Contact Salesforce Support and create a case to temporarily enable API access for managed packages.
B. Create a connected app in the central org and add the callback URL for each org in the package it is installed in to redirect after a successful authentication.
C. Use an encrypted field to store the password that the security team enters.
Answer: B (LEAVE A REPLY)
For a distributable package to securely access a central "Hub" org, the architecture must support the OAuth
2.0 Web Server Flow. This flow is designed for applications (like the package installed in a "Spoke" org) that can securely store a Client Secret and need to act on behalf of a specific user.
The Connected App in the central org acts as the "Identity and Access" gatekeeper. A critical component of the Connected App configuration is the Callback URL (Redirect URI). When a user in the "Subscriber" org clicks "Authorize," Salesforce redirects them to the central org to log in. After successful authentication, the central org needs to know where to send the "Authorization Code" back to.
In a multi-org packaging scenario, each subscriber org will have a unique instance URL (e.g., na15.salesforce.

com). The architect must ensure that the Connected App's callback URLs are correctly configured to handle these redirects.
Option C (Encrypted Passwords) is a major security risk and is considered an "anti-pattern" in modern integration. Option A is unnecessary, as API access is a standard feature. By using the Connected App with correct Callback URLs, the architect allows the security team in the central org to oversee exactly which
"Spoke" orgs have authorized access. They can use the "Connected Apps OAuth Usage" page to monitor, rotate secrets, or revoke access for individual orgs, providing the granular security control required for an enterprise-grade distributed Salesforce architecture.

NEW QUESTION: 14
Northern Trail Outfitters needs to use Shield Platform Encryption to encrypt social security numbers in order to meet a business requirement. Which action should an integration architect take prior to the implementation of Shield Platform Encryption?
A. Use Shield Platform Encryption as a user authentication or authorization tool.
B. Encrypt all the data so that it is secure.
C. Encrypt the data using the most current key.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 15
Northern Trail Outfitters (NTO) has an affiliate company that would like immediate notifications of changes to opportunities in the NTO Salesforce instance. The affiliate company has a CometD client available.
Which solution is recommended in order to meet the requirement?
A. Create a connected app in the affiliate org and select "Accept CometD API Requests".
B. Create a PushTopic update event on the Opportunity object to allow the subscriber to react to the streaming API.
C. Implement a polling mechanism in the client that calls the SOAP API getUpdated method to get the ID values of each updated record.
Answer: (SHOW ANSWER)
To provide "immediate notifications" to a client that already supports CometD, the architect must utilize Salesforce's Streaming API. PushTopic Events are the classic Streaming API implementation for broadcasting changes to specific records.
A PushTopic is a definition that includes a SOQL query and a set of trigger events (Create, Update, Delete).
When an Opportunity in NTO's instance is updated and matches the query, Salesforce automatically pushes a notification to the topic's channel. The affiliate's CometD client, having subscribed to this channel, receives the data payload instantly.12 Option A (Polling) is the architectural opposite of "immediate notifications". Polling is inefficient, consumes API limits, and int13roduces latency between the actual record change and the client's discovery of that change.
Option C is incorrect as there is no "Accept CometD" setting in a Connected App; while a

Connected App is used for OAuth authentication, the PushTopic is the mechanism that actually defines and delivers the stream. By using PushTopic, NTO provides a decoupled, event-driven architecture that fulfills the requirement for real-time visibility while minimizing the technical burden on the affiliate's infrastructure.

NEW QUESTION: 16
Northern Trail Outfitters needs a synchronous callout from Salesforce to an Order Management System (OMS) when an opportunity is "Closed/Won" with products attached. What should an integration architect do to satisfy these requirements?
A. Build a Lightning component that makes a synchronous Apex REST callout to the OMS when a button is clicked.
B. Write a trigger that invokes an Apex proxy class to make a REST callout to the OMS.
C. Develop a batch Apex job that aggregates closed opportunities and makes a REST callout to the OMS hourly.
Answer: A (LEAVE A REPLY)
To satisfy a requirement for a synchronous callout triggered by a user action, the architect should use a UI- driven approach, such as a Lightning component and a button.
In Salesforce, triggers (Option B) are primarily used for asynchronous logic in integration contexts. Because a trigger executes as part of the database save operation, making a synchronous callout directly from a trigger is prohibited as it would block the database transaction until the external system responds, leading to performance degradation and "uncommitted work pending" errors. If a trigger must initiate an integration, it must do so asynchronously (using @future or Queueable Apex), which violates the requirement for a synchronous call.
By using a Lightning component, the architect can initiate a synchronous Request-and-Reply pattern. When the sales rep clicks the "7Submit to OMS" button, the componen8t invokes an Apex method that makes the REST callout to the OMS in real-time. The user remains on the page while the system waits for the OMS to respond, allowing for immediate feedback-such as an order confirmation number or an error message-to be displayed in the UI. A Batch Apex job (Option C) is inherently asynchronous and delayed, making it unsuitable for a synchronous, real-time fulfillment requirement.

NEW QUESTION: 17
An integration architect has designed a mobile application for Salesforce users to get data while on the road using a custom user interface (UI). The application is secured with OAuth and is currently functioning well.
There is a new requirement where the mobile application needs to obtain the GPS coordinates and store them on a custom geolocation field. The geolocation field is secured with field-level security, so users can view the value without changing it. What should be don4e to meet the requirement?
A. The mobil10e device makes a REST Apex inbound call.
B. The 15mobile device makes a REST API inboun16d call.
C. The mobile device receives19 a REST Apex callout call.
Answer: B (LEAVE A REPLY)
When a custom mobile application already secured with OAuth needs to update a record in Salesforce, the standard architectural recommendation is to use the REST API. The REST API is optimized for mobile environments because it uses lightweight JSON payloads and follows standard HTTP methods (such as PATCH for updates), which are highly compatible with mobile development frameworks.
In this specific scenario, the architect must address the Field-Level Security (FLS) constraint. Because the geolocation field is set to read-only for users, a standard UI-based update would typically fail. However, when using an inbound REST API call with a properly authorized integration user or via a "System Mode" context (if utilizing a custom Apex REST resource), the system can be configured to bypass UI-level restrictions while maintaining data integrity.
The mobile device captures the coordinates via the device's native GPS capabilities and initiates an inbound call to the Salesforce REST endpoint. Option A (Apex inbound call) is a subset of REST functionality but is only necessary if complex server-side logic is required that the standard REST API cannot handle. Option C is technically incorrect as mobile devices do not typically "receive" callouts from Salesforce in this pattern; they initiate the requests. By leveraging the standard REST API, the architect ensures a scalable, secure, and standardized integration that adheres to Salesforce's mobile-first integration principles.

NEW QUESTION: 18
Northern Trail Outfitters (NTO) wants to improve the quality of callouts from Salesforce to its REST APIs.
For this purpose, NTO will require all API Clients/consumers to adhere to REST API Markup Language (RAML) specifications that include the field-level definition of every API request and response Payload. The RAML specs serve as interface contracts that Apex REST API Clients can rely on. Which design specification should the integration architect include in the integration architecture to ensure that Apex REST API Clients' unit tests confirm Adherence to the RAML specs?
A. Require the Apex REST API Clients to implement the HttpCalloutMock
B. Call the HttpCalloutMock Implementation from the Apex REST API Clients.

C. Implement HttpCalloutMock to return responses per RAML specification.

Answer: C (LEAVE A REPLY)

In a contract-first integration strategy using RAML (RESTful API Modeling Language), the specification defines the exact structure of requests and responses. Because Salesforce unit tests cannot perform actual network callouts, the platform requires d1evelopers to use the HttpCalloutMock interface to simulate responses.

To ensure that the integration code strictly adheres to the established RAML contract, the integration architect must mandate that the HttpCalloutMock implementation returns responses that mirror the RAML specification. This means the mock must include all required fields, correct data types, and the expected HTTP status codes (e.g., 200 OK, 201 Created) as defined in the contract. By doing this, the unit tests verify that the Apex client code can successfully parse and process the specific JSON or XML payloads defined in the RAML spec.

Option A and B are technically imprecise. The Apex client does not "implement" the mock; rather, the test class provides a separate mock implementation to the runtime via Test.setMock(). The value of the integration architecture lies in the content of that mock. If the mock is designed to return contract-compliant data, then any change to the RAML that breaks the Apex code's ability to process it will be caught immediately during the testing phase. This "Mock-as-a-Contract" approach provides a safety net, ensuring that Salesforce remains compatible with external services even as those services evolve, provided the RAML is kept up to date.

NEW QUESTION: 19

Universal Containers (UC) support agents would like to open bank accounts on the spot. During the process, agents execute credit checks through external agencies. At any given time, up to 30 concurrent reps will be using the service. Which error handling mechanisms should be built to display an error to the agent when the credit verification process has failed?

A. Handle Integration errors in the middleware in case the verification process is down, then the middleware should retry processing the request multiple times.

B. In case the verification process is down, use fire and forget mechanism instead of Request and Reply to allow the agent to get the response back when the service is back online.

C. Handle the error in the synchronous callout and display a message to the agent. (Note: While not explicitly in the user's snippet, A and B are provided options; the standard architect answer for "displaying an error to the agent" in a synchronous flow is handling the exception in the UI layer).

Answer: A (LEAVE A REPLY)

In a synchronous Request-Reply scenario where a bank agent is waiting "on the spot" for a credit check, the error-handling strategy must balance immediate feedback with system resilience.

Option A is the recommended architectural approach for enterprise resiliency. By placing a Middleware layer (like MuleSoft) between Salesforce and the credit agencies, the architect can implement sophisticated error- handling patterns that are invisible to the user but critical for success. If a credit agency's API is momentarily unreachable, the middleware can perform automated retries (e.g., three attempts with 500ms intervals). If the retries still fail, the middleware sends a clean, structured error response back to Salesforce.

Option B (Fire and Forget) is fundamentally unsuitable for this use case because the agent needs the result immediately to open the account; they cannot wait for a callback that might arrive hours later. Option C (Mock service) is only a testing tool and provides no value in a production environment where real financial data is required. By delegating the retry logic to the middleware, the architect protects Salesforce's concurrent request limits (since the agent only occupies a thread for the duration of the final response) and ensures that transient network issues do not result in a "failed" bank account application for the customer.

NEW QUESTION: 20
A customer's enterprise architect has identified requirements around caching, queuing, error handling, alerts, retries, event handling, etc. Which recommendation should the integration architect make?
A. Transform a Fire and Forget mechanism to Request and Reply, which should be handled by middleware tools (like ETL/ESB) to improve performance.
B. Event handling in a publish/subscribe scenario; the middleware can be used to route requests or messages to active data-event subscribers from active data-event publishers.
C. Message transformation and protocol translation should be done within Salesforce.
Answer: B (LEAVE A REPLY)
When an enterprise architect identifies complex infrastructure needs such as caching, queuing, and sophisticated event routing, it signals that a point-to-point integration architecture is insufficient. In such cases, the Integration Architect should recommend a Middleware-mediated architecture.
Middleware tools, such as an Enterprise Service Bus (ESB) or an iPaaS (Integration Platform as a Service), are specifically designed to fulfill these complex "Quality of Service" (QoS) requirements. In a publish/subscribe scenario, the middleware acts as the central orchestrator. It can receive a single "Fire and Forget" event from a publisher (like Salesforce) and then manage the technical complexities of routing that message to multiple active subscribers.
Middleware handles infrastructure-level tasks such as message queuing for offline systems, automatic retries with exponential backoff, and error handling with alerts-capabilities that are either unavailable or difficult to scale within Salesforce natively. Performing message transformation and protocol translation (e.g., SOAP to REST) within the middleware layer also protects Salesforce's Apex CPU limits. By leveraging middleware for these concerns, the architect ensures that Salesforce remains a performant engagement layer while the middleware provides the robust technical backbone for a resilient enterprise landscape.

NEW QUESTION: 21
A media company recently implemented an IAM system supporting SAML and OpenId. The IAM system must integrate with Salesforce to give new self-service customers instant access to Salesforce Community Cloud. Which requirement should Salesforce Community Cloud support for self-registration and SSO?
A. SAML SSO and Registration Handler

B. SAML SSO and Just-in-Time (JIT) provisioning
C. OpenId Connect Authentication Provider and JIT provisioning
Answer: (SHOW ANSWER)
To provide "instant access" for new customers via an external IAM system using SAML, Salesforce provides a declarative feature called Just-in-Time (JIT) provisioning.
When a customer attempts to log in to the Community (Experience Cloud) through the IAM system, the IAM system (acting as the Identity Provider) sends a SAML assertion to Salesforce. If JIT provisioning is enabled, Salesforce parses the user information contained in that assertion-such as name, email, and federation ID. If a corresponding User record does not exist, Salesforce automatically creates one on-the-fly and then logs the user in. This eliminates the need for a manual registration step or pre-provisioning accounts.
Option A is slightly incorrect because Registration Handlers are specifically associated with Authentication Providers (which use OpenID Connect/OAuth), not SAML SSO. Option C is incorrect because JIT provisioning is a feature of SAML, while Authentication Providers use the Registration Handler class to achieve the same result. For a "self-service" scenario where speed to market and standard protocols are key, SAML SSO with JIT provisioning is the architect's primary choice for automating user management and providing a seamless single-entry point for subscribers.

NEW QUESTION: 22
A large consumer goods manufacturer operating in multiple countries is planning to implement Salesforce for its sales and support operations globally. The Manufacturer has the following security requirements:
* Internal users from each country have to be authenticated with their local active directory.
* Customers can create their own login or use Google login.
* Partners have to be authenticated through a central system which is to be determined.
* Internal users will have access to the central Enterprise Resource Planning (ERP) with their credentials maintained in the ERP system.
* Additional internal systems will be integrated with Salesforce for sales and support business processes.
Which requirement should the integration architect evaluate while designing the integration needs of this project?
A. Evaluate Salesforce native authentication mechanism for all users including customers and partners.
B. Evaluate the build of a custom authentication mechanism for users in each country and support for customers and partners.
C. Consider a third-party single sign-on (SSO) solution supporting all user authentication including customer and partner.
Answer: C (LEAVE A REPLY)
Managing identity across a global enterprise with diverse user personas (Employees, Customers, Partners) requires a centralized Identity and Access Management (IAM) strategy. In a landscape

involving multiple local Active Directories, social logins (Google), and a central ERP system, attempting to manage authentication natively within Salesforce or through custom-built local silos would result in high technical debt and security vulnerabilities.
The architect should recommend a third-party Single Sign-On (SSO) solution, acting as a central Identity Provider (IdP). This IdP serves as the orchestration layer for all authentication requests.
* For Internal Users: The IdP can federate with the various local Active Directories, allowing users to log in with their existing corporate credentials.
* For Customers: The IdP can handle "Social Sign-On" (OpenID Connect) with Google and manage self-registration.
* For Partners: It provides the "central system" required for their authentication.
By using a central SSO solution, Salesforce acts as a Service Provider (SP). When a user attempts to access Salesforce, the request is redirected to the IdP via the SAML 2.0 or OpenID Connect protocol. Once the IdP validates the user against the appropriate backend (AD, Google, or its own directory), it sends a secure assertion back to Salesforce to grant access.
Furthermore, this central IdP can facilitate access to the ERP system and other internal systems. If these systems support SAML, the same SSO session used for Salesforce can be extended to them, providing a true single sign-on experience. This architecture centralizes security auditing, simplifies user de-provisioning (the
"kill switch" effect), and ensures a consistent user experience across the global manufacturing landscape.
Implementing a thir2d-party IdP is the industry-standard approach for complex integrations where security, scalabi3lity, and multi-protocol support are primary requirements.

NEW QUESTION: 23
Northern Trail Outfitters wants to use Salesforce as a front end for creating accounts using the lead-to- opportunity process. An order is created in Salesforce when the opportunity is Closed/Won, but the back-end Enterprise Resource Planning (ERP) system is the data master for order. The customer wants to be able to see within Salesforce all the stages of order processing like Order Created, Order Shipped, and Order Paid that are within the retention window. Which message durability consideration should an integration architect make when designing a solution to meet these business requirements?
A. High-volume event messages are stored for 24 hours (1 day).
B. High-volume event messages are stored for 72 hours (3 days).
C. When subscribing to Salesforce Event Bus, ReplayID is used with a value of -1 to be able to see new events.
Answer: B (LEAVE A REPLY)
When designing a solution that requires Salesforce to receive and display updates from a back-end ERP (such as order status changes), message durability is a critical factor for ensuring data consistency. In an event- driven architecture using Platform Events or Change Data Capture (CDC), Salesforce utilizes an event bus to handle these incoming notifications.

For high-volume event messages, the Salesforce platform provides a native 72-hour (3-day) retention window. This is a significant architectural advantage for several reasons:
* System Resilience: If the Salesforce org or the integration middleware experiences a temporary disruption or is undergoing maintenance, the event messages published by the ERP remain stored in the bus for up to 3 days.
* Data Recovery: Once the connection is restored, the subscribing system (Salesforce) can use the Replay ID to catch up on any missed events from the last 72 hours, ensuring that order stages like
"Order Shipped" or "Order Paid" are not missed.
* SLA Management: This 3-day window exceeds the 24-hour limit of older technologies like PushTopics or Outbound Messaging (Option A), providing more breathing room for disaster recovery scenarios.
While ReplayID -1 (Option C) is used to subscribe only to new events published after the subscription starts, it does not address the durability or retention of historical events needed for recovery. By highlighting the 72- hour retention window, the integration architect provides a design that is robust against outages and guarantees that the "System of Engagement" (Salesforce) stays synchronized with the "System of Record" (ERP).

NEW QUESTION: 24
An integration architect has received a request to prevent employees that leave the company from accessing data in Salesforce after they are deactivated in the company's HR system. What should the integration architect determine before recommending a solution?
A. Data access prevention requirements, then identify frequency
B. Inbound integration requirements, then identify frequency
C. Data access prevention requirements, integration requirements, and system constraints
Answer: (SHOW ANSWER)

NEW QUESTION: 25
Which Web Services Description Language (WSDL) should an architect consider when creating an integration that might be used for more than one Salesforce org and different metadata?
A. Enterprise WSDL
B. Partner WSDL
C. SOAP API WSDL
Answer: (SHOW ANSWER)
In the world of Salesforce SOAP APIs, the choice of WSDL depends on the nature of the application being built. When an architect needs to build an integration that is org-agnostic-meaning it can work across multiple different Salesforce organizations with varying custom objects and fields-the Partner WSDL is the correct choice.
The Partner WSDL is loosely-typed. It does not contain information about an org's specific custom metadata; instead, it uses a generic sObject structure. This allows a single client application (like a third-party integration tool or a mobile app) to connect to any Salesforce org

and dynamically discover its schema at runtime. Because it is not tied to a specific org's metadata, it does not need to be regenerated every time a custom field is added to one of the target orgs.
In contrast, Option A (Enterprise WSDL) is strongly-typed. It is generated specifically for one org and contains hard-coded references to that org's custom objects and fields. While this provides better compile-time safety for internal, single-org integrations, it is unsuitable for an application intended for "more than one Salesforce org" because the WSDL would be invalid for any org that doesn't have the exact same metadata.
Option C is a general term; the actual choices provided by Salesforce for integration are the Partner or Enterprise WSDLs. Therefore, for maximum flexibility and reusability across a multi-org landscape, the Partner WSDL is the industry standard.

NEW QUESTION: 26
An integration architect needs to build a solution that will use the Streaming API, but the data loss should be minimized, even when the client re-connects every couple of days. Which two types of Streaming API events should be considered?
A. High Volume Platform and Generic Events
B. Change Data Capture and High Volume Platform Events
C. Push Topic and Change Data Capture Events
Answer: B (LEAVE A REPLY)
In a robust event-driven architecture, "durability" is the ability of a system to retain events so that a subscriber can retrieve them even after being offline. For an architect needing to minimize data loss over "every couple of days," the selection must focus on event types that support a high retention window.
Salesforce provides two modern event types specifically designed for high-scale, durable messaging:
* Change Data Capture (CDC): This automatically broadcasts changes to Salesforce records (Create, Update, Delete, Undelete). CDC events are highly durable and are retained in the event bus for 72 hours (3 days).
* High-Volume Platform Events: These are custom events defined by the architect. Like CDC, High- Volume Platform Events also provide a 72-hour retention window.
Earlier versions of the Streaming API, such as PushTopic Events and Generic Events (Option C and parts of A), typically offered a 24-hour retention window, which would not satisfy the requirement of a client re- connecting "every couple of days" (potentially 48-72 hours later).
By utilizing CDC and High-Volume Platform Events, the architect can leverage the Replay ID. When the client re-connects, it can send the Replay ID of the last event it successfully processed. Salesforce then
"replays" all events that occurred during the client's downtime from that specific point, up to the 72-hour limit. This mechanism ensures zero data loss for planned or unplanned outages lasting up to three days, making it the most resilient choice for the specified requirements.

NEW QUESTION: 27
Northern Trail Outfitters wants to use Salesforce as a front end for creating accounts using the lead-to- opportunity process.
* An order is created in Salesforce when the opportunity is Closed/Won, but the back-end Enterprise Resource Planning (ERP) system is the data Master for order.
* The customer wants to be able to see within Salesforce all the stages of order processing, like Order Created, Order Shipped, and Order Paid, that are within the retention window.
Which message durability consideration should an integration architect make when designing a solution to meet these business requirements?
A. When subscribing to Salesforce Event Bus, ReplayID is used with a value of -1 to be able to see new events.
B. High-volume event messages are stored for 24 hours (1 day).
C. When subscribing to Salesforce Event Bus, ReplayID is used with a value of -2 to be able to see old and new events.
Answer: C (LEAVE A REPLY)
When designing an event-driven architecture to track order processing stages (Created, Shipped, Paid), the Integration Architect must ensure the solution provides "durability"-the ability to recover messages that were sent while a subscriber was offline or during a system failure. In Salesforce, this is managed through the Event Bus and the ReplayID mechanism.
For High-Volume Platform Events and Change Data Capture, Salesforce provides a standard retention window of 72 hours (3 days). This means that events are stored in the bus for this duration, allowing clients to "replay" events that occurred in the past. To leverage this durability, the subscribing client must specify where in the event stream they wish to begin receiving messages.
There are two special values for the ReplayID:
* ReplayID = -1 (Tip of the Stream): The subscriber receives only new events that are published after the subscription is established. Any events published while the client was disconnected are missed. This does not meet the requirement of seeing processing stages that occurred within the retention window if the connection was interrupted.
* ReplayID = -2 (All Available Events): The subscriber receives all events that are currently stored in the event bus (up to 72 hours old) as well as all new events.
By recommending the use of ReplayID = -2, the architect ensures that even if the Salesforce frontend or the integration middleware experiences downtime, the system can "catch up" by retrieving all order status updates (Shipped, Paid, etc.) that were published during that window. This provides a robust and resilient user experience, ensuring that the Opportunity and Order records in Salesforce accurately reflect the state of the ERP system without data gaps. This configuration is essential for maintaining data synchronization in a distributed landscape where Salesforce acts as the engagement layer for a back-1end ERP master.

NEW QUESTION: 28

Northern Trail Outfitters (NTO) has a requirement to encrypt a few widely-used standard fields. NTO also wants to be able to use these fields in record-triggered flows.
Which security solution should an integration architect recommend to fulfill the business use case?

A. Shield Platform Encryption

B. Classic Encryption

C. Data Masking

Answer: A (LEAVE A REPLY)

To satisfy the requirement of encrypting standard fields while maintaining their functionality within record- triggered flows, Shield Platform Encryption is the recommended architectural solution.1 Shield Platform Encryption is a modern security layer that allows for encryption at rest while 2preserving critical platform features. Unlike Classic Encryption (Option B)-which is limited to a specific "Encrypted Text" custom field type and often breaks platform features like search and automation-Shield is designed to work with standard fields such as Name, Email, and Phone.
Key architectural considerations for Shield include:

* Compatibility with Automation: Shield fields can be used in Flows, Apex triggers, and validation rules. This allows NTO to implement the required record-triggered business logic without needing to decrypt the data manually in code.
* Search and Filtering: By using Deterministic Encryption, Shield allows users to filter and search for records based on encrypted fields, which is often a requirement for "widely-used" standard fields.
* Compliance and Governance: Shield provides advanced key management (Bring Your Own Key - BYOK) and auditing, ensuring that NTO meets corporate security guidelines while data is being processed by the platform.

Data Masking (Option C) is primarily used for sandboxes to obfuscate PII during testing and is not a production encryption-at-rest solution. By recommending Shield, the architect provides a transparent security model that protects sensitive data without sacrificing the declarative power of Flow Builder.

NEW QUESTION: 29

Northern Trail Outfitters (NTO) wants to improve the quality of callouts from Salesforce to its REST APIs by requiring all API clients to adhere to RAML (REST API Markup Language) specifications. The RAML specs serve as interface contracts. Which design specification should the integration architect include in the integration architecture to ensure that Apex REST API Clients' unit tests confirm adherence to the RAML specs?

A. Implement HttpCalloutMock to return responses per RAML specification.

B. Call the HttpCalloutMock implementation from the Apex REST API Clients.

C. Require the Apex REST API Clients to implement the HttpCalloutMock.

Answer: A (LEAVE A REPLY)

In a contract-first integration approach using RAML, the specification acts as the single source of truth for request and response structures. Since Salesforce unit tests are prohibited from

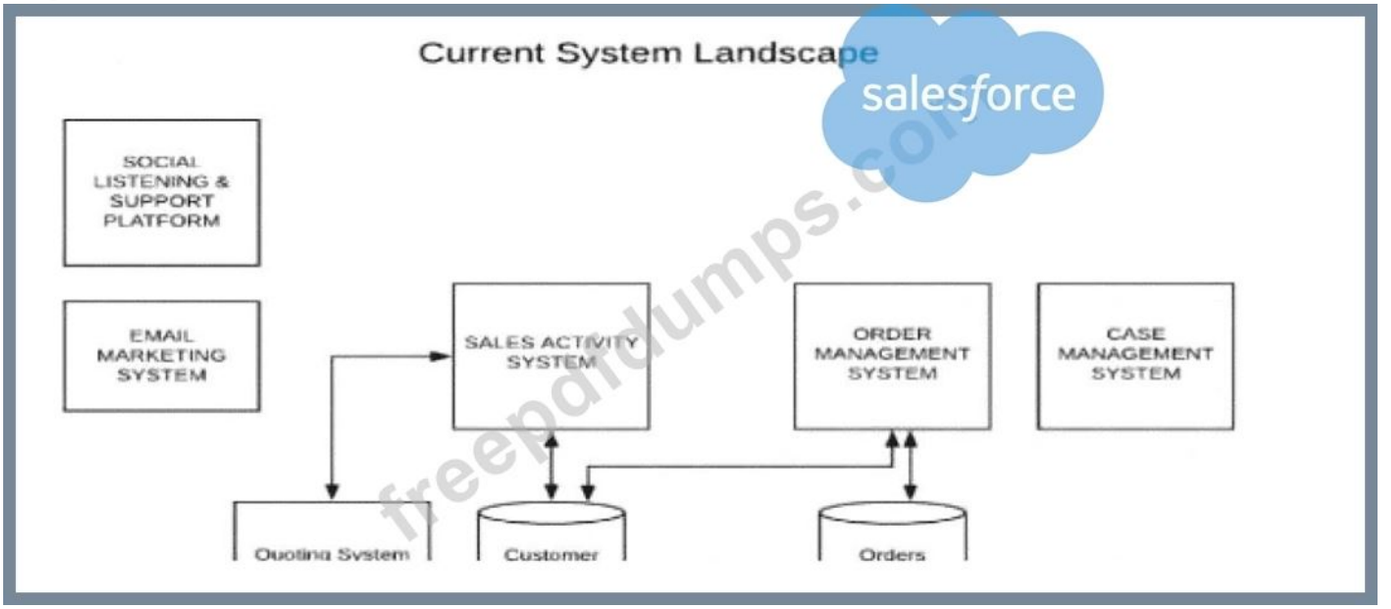
performing actual network callouts, the HttpCalloutMock interface must be used to simulate external API behavior.
To ensure unit tests truly confirm adherence to the RAML contract, the architect must mandate that the mock implementation specifically returns responses formatted per the RAML specification. This means the mock's JSON or XML body, headers, and HTTP status codes (e.g., 200 OK, 400 Bad Request) must exactly match the "interface contract" defined in the RAML file. By strictly aligning the mock with the RAML spec, developers ensure that the Apex client's parsing logic (e.
g., JSON.deserialize()) is tested against the agreed-upon data model. If the external service later changes its schema in a way that deviates from the RAML, the unit tests-which are based on that contract-will help identify where the Apex code might fail. Options B and C are technically incorrect: the client does not "call" or "implement" the mock; rather, the test runtime provides the mock instance to the client via Test.setMock().

NEW QUESTION: 30
Northern Trail Outfitters submits orders to the manufacturing system web service. Recently, the system has experienced outages that keep service unavailable for several days. Which solution should an integration architect recommend to handle errors during these types of service outages?1718
A. Use middleware queuing and buffering to insulate Salesforce from1920 system outages.
B. Use Outbound Messaging to automatically retry failed service calls.
C. Use Platform Event replayId and custom scheduled Apex process to retrieve missed events.
Answer: A (LEAVE A REPLY)
When a target system experiences prolonged outages (lasting "several days"), point-to-point integrations built directly within Salesforce are prone to failure because they lack the persistence required for long-term retries.
The architecturally sound recommendation is to utilize middleware queuing and buffering to "insulate" Salesforce from the target system's instability.
In this architecture, Salesforce sends the order to a middleware layer (such as an ESB or iPaaS). The middleware immediately acknowledges receipt of the message, freeing up Salesforce resources. If the manufacturing system is offline, the middleware stores the order in a persistent Message Queue. Unlike Salesforce Outbound Messaging (Option B), which only retries for up to 24 hours, enterprise middleware can be configured to hold messages for days or even weeks. Middleware also provides sophisticated Quality of Service (QoS) features, such as "Dead Letter Queues" for manual intervention and customized retry schedules (e.g., retrying every hour instead of every few minutes).
This decoupling ensures that Salesforce users can continue to create and "send" orders without seeing technical errors, even while the backend manufacturing system is down. Once the manufacturing service is restored, the middleware "drains" the queue, delivering all buffered orders in the correct sequence. This strategy provides the highest level of reliability and resilience for mission-critical business processes.

NEW QUESTION: 31
Northern Trail Outfitters is planning to perform nightly batch loads into Salesforce from an external system with a custom Java application using the Bulk API. The CIO is curious about monitoring recommendations for the jobs from the technical architect. Which recommendation should help meet the requirements?
A. Use the getBatchInfo method in the Java application to monitor the status of the jobs from the Java application.
B. Write the error response from the Bulk API status to a custom error logging object in Salesforce using an Apex trigger, and create reports on the object.
C. Set the Salesforce debug logs level to "finest", and add the user ID running the job to monitor in the
"Debug Logs" in the setup menu.
Answer: A (LEAVE A REPLY)
For high-volume data loads using the Bulk API, monitoring should be performed programmatically by the orchestrating client-in this case, the custom Java application. The Bulk API is asynchronous, meaning that when you submit a job, Salesforce acknowledges the request and processes it in the background.
The Java application must actively track the state of its own jobs. Using the `getBatchInfo` (or `getJobInfo` in Bulk API 2.0) method allows the application to retrieve the real-time status of each batch. The application can check for statuses such as `Queued`, `InProgress`, `Completed`, or `Failed`. Once a batch is marked as
`Completed`, the application can then call `getBatchResult` to retrieve a list of successes and failures for individual records.
Option B is architecturally unsound because Bulk API operations are designed to bypass most synchronous Apex logic to ensure performance; furthermore, creating custom records for every error in a "nightly batch load" would likely hit other platform limits (like storage or CPU) and defeat the purpose of using the Bulk API. Option C is ineffective for Bulk API monitoring, as debug logs do not capture the background processing of bulk batches and would quickly hit the log size limits.
By recommending Option A, the architect ensures that the Java application maintains full control over the integration lifecycle. The application can log errors locally, implement automated retries for transient failures, and provide the CIO with accurate, high-level reporting on the success rate of the nightly loads without placing unnecessary overhead on the Salesforce platform.

NEW QUESTION: 32

A large business-to-consumer (B2C) customer is planning to implement Salesforce CRM to become a customer-centric enterprise. Below is the B2C customer's current system landscape diagram.



The goals for implementing Salesforce include:

* Develop a 360-degree view of the customer.
* Leverage Salesforce capabilities for marketing, sales, and service processes.
* Reuse Enterprise capabilities built for quoting and order management processes.

Which three systems from the current system landscape can be retired with the implementation of Salesforce?

A. Order Management, Case Management, and Email Marketing

B. Sales Activity, Order Management, and Case Management

C. Email Marketing, Sales Activity, and Case Management

Answer: C (LEAVE A REPLY)

Comprehensive and Detailed 250 to 350 words of Explanation From Salesforce Pl14atform Integration Architect documents: In the rol15e of a Salesforce Platform Integration Architect, evaluating a legacy landscape requires a clinical mapping of current system functions against Salesforce's native capabilities, while strictly adhering to the "Constraints and Goals" provided by the business. The objective here is to maximize the ROI of the Salesforce implementation by consolidating redundant systems into the core platform.

According to Goal 2, the business intends to utilize Salesforce for Marketing, Sales, and Service processes.

Salesforce is architected to handle these three domains through its core clouds: Marketing Cloud (replacing the legacy Email Marketing System), Sales Cloud (replacing the Sales Activity System), and Service Cloud (replacing the Case Management System). By consolidating these three specific functions into Salesforce, the organization achieves Goal 1, which is the creation of a 360-degree view of the customer. When these activities occur on a single platform, the data is unified, eliminating the silos that existed in the previous landscape.
However, the architect must also respect the technical constraints defined in Goal 3, which explicitly states the need to reuse enterprise capabilities built for quoting and order management. In architectural design, this designates the "Quoting System" and the "Order Management System" as systems of record that must remain in the future-state landscape. These systems likely contain complex, proprietary logic or are tightly coupled with back-end ERP systems like SAP Business Suite, making them "non-negotiable" for retirement at this stage.
Therefore, because Email Marketing, Sales Activity, and Case Management map directly to Salesforce's primary strengths and are not excluded by the "reuse" requirement, they are the three systems that should be retired. This strategic retirement simplifies the integration architecture, allowing the architect to focus on building robust integration patterns (such as Request-Reply or Fire-and-Forget) between Salesforce and the remaining Quoting and Order Management systems.

NEW QUESTION: 33
What is the first thing an integration architect should validate if a callout from a Lightning web component to an external endpoint is failing?
A. The endpoint URL has been added to Content Security Policies.
B. The endpoint domain has been added to Cross-Origin Resource Sharing.
C. The endpoint URL has been added to Remote Site Settings.
Answer: A (LEAVE A REPLY)
When an integration callout initiated from a Lightning Web Component (LWC) fails, the architect must distinguish between client-side and server-side security layers. Unlike Apex callouts, which are governed by Remote Site Settings at the server level, LWC requests originate directly from the user's browser.
Consequently, they are subject to the browser's Content Security Policy (CSP).
CSP is a security layer that helps detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. It prevents a website from loading content from a third party unless that domain is explicitly safe-listed. If an LWC attempts to connect to an external API endpoint that is not listed in the CSP Trusted Sites in Salesforce Setup, the browser will block the request before it is even sent, often returning a "Refused to connect because it violates the document's Content Security Policy" error. While Cross-Origin Resource Sharing (CORS) is also a browser-level security mechanism, it must be configured on the external server to allow the browser to access its resources; however, the first validation step within the Salesforce environment for a failing LWC callout is ensuring the domain is allowed by the org's CSP.

NEW QUESTION: 34
Universal Containers (UC) is a global financial company. UC support agents would like to open bank accounts on the spot for customers who inquire about UC products. During the bank account opening process, the agents execute credit checks for the customers through external agencies. At any given time, up to 30 concurrent reps will be using the service to perform credit checks for customers. Which error handling mechanisms should be built to display an error to the agent when the credit verification process has failed?
A. Handle integration errors in the middleware in case the verification process is down, then the middleware should retry processing the request multiple times.
B. In case the verification process is down, use fire and forget mechanism instead of Request and Reply to allow the agent to get the response back when the service is back online.
C. In case the verification process is down, use mock service to send the response to the agent.
Answer: A (LEAVE A REPLY)
In a synchronous Request-Reply integration-where a bank agent is waiting for a real-time credit check to open an account-the error handling strategy must balance user experience with system resilience. Handling these errors at the Middleware layer is the architecturally preferred solution for managing complex retry logic and providing a clean response to Salesforce.
If the external credit agency's service is momentarily unavailable, the middleware (such as an ESB or MuleSoft) can automatically retry the request multiple times using a pre-defined strategy (e.g., exponential backoff). This "self-healing" behavior can often resolve transient network issues before the Salesforce agent even realizes there was a problem. If the retries fail, the middleware then returns a structured error message to Salesforce, which is displayed to the agent via the UI. Option B (Fire and Forget) is unsuitable for this use case because the agent needs the result immediately to proceed with the bank account opening; they cannot afford to wait for a background process to finish hours later. Option C (Mock Service) is a testing tool and has no place in a production environment where real financial decisions are being made. By delegating error management to the middleware, UC ensures that its Salesforce instance remains performant (avoiding long-running request timeo1112uts) while maximizing the chances of a successful credit check through automated, controlled retries.1314

NEW QUESTION: 35
A company needs to integrate a legacy on-premise application that can only support SOAP API. After the integration architect evaluates the requirements and volume, they determine that the Fire and Forget integration pattern will be most appropriate for sending data from Salesforce to the external application and getting response back in a strongly-typed format.
Which integration capabilities should be used to integrate the two systems?
A. Outbound Messaging for Salesforce to Legacy System direction and SOAP API using Enterprise WSDL for the communication back from legacy system to Salesforce
B. Platform Events for Salesforce to Legacy System direction and SOAP API using Enterprise WSDL for the communication back from legacy system to Salesforce

C. Outbound Messaging for Salesforce to Legacy System direction and SOAP API using Partner Web Services Description Language (WSDL) for the communication back from legacy system to Salesforce

Answer: A (LEAVE A REPLY)

When integrating with a legacy SOAP-only application using a Fire and Forget pattern, Salesforce Outbound Messaging is the native, declarative choice.

Outbound Messaging is a platform feature that sends a SOAP message to a designated endpoint when specific criteria are met. It is inherently asynchronous and provides built-in reliability; if the legacy system is offline, Salesforce will automatically retry the delivery for up to 24 hours. This perfectly fits the "Fire and Forget" requirement.

For the strongly-typed response back (Remote Call-In), the Enterprise WSDL is the correct recommendation.1415

* Enterprise WSDL: This is a strongly-typed WSDL generated specifically for one org's metadata. It includes specific references to16 custom objects and fields, ensu17ring that the legacy system can communicate with Salesforce using a rigid, predictable data structure.
* Partner WSDL (Option C): This is a loosely-typed WSDL designed for developers building tools that must work across many different orgs; it is not ideal for an internal, strongly-typed legacy integration.

While Platform Events (Option B) are a modern alternative, they typically use REST or Streaming APIs, making them a less natural fit for an application that "can only support SOAP". By combining Outbound Messaging and the Enterprise WSDL, the architect provides a robust, SOAP-native solution that satisfies the business's technical constraints and reliability requirements.

NEW QUESTION: 36

A company captures orders and needs to send them to the Order fulfillment system. The user is not required to have confirmation from the Order fulfillment system. Which system constraint question should be considered when designing an integration to send orders from Salesforce to a fulfillment system?

A. What latency is acceptable for orders to reach the fulfillment system?

B. Can the fulfillment system implement a contract-first Outbound Messaging interface?

C. Which system will validate order shipping addresses?

Answer: (SHOW ANSWER)

When designing an integration where the user does not require immediate confirmation, the architect is moving away from a synchronous "Request-Reply" pattern toward an asynchronous "Fire-and-Forget"16 or

"Batch Processing" pattern. In such scenarios, the most critical architectural constraint is defining the latency requirements.

Latency dictates the technical choice of the integration tool. If the fulfillment system needs the order within seconds of creation to begin a high-speed picking process, the architect might choose Salesforce Outbound Messaging or an Apex Callout triggered by a Platform Event. If the

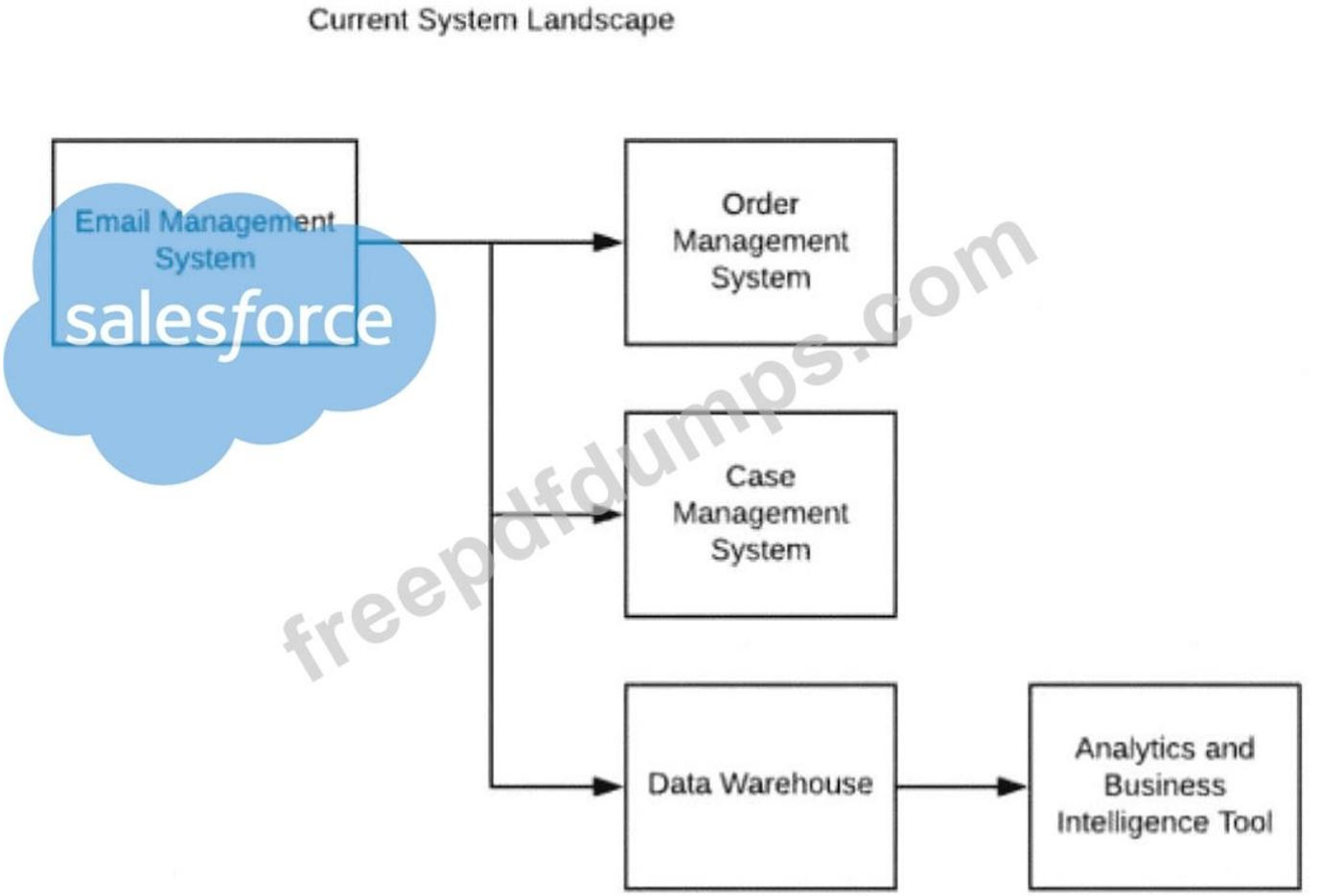
system only needs to process orders once an hour or overnight, a Batch ETL process is more appropriate. Understanding the acceptable delay (latency) ensures that the solution meets business expectations without over-engineering for real-time performance where it isn't required. While Option B (Outbound Messaging) is a valid technical capability, it is a specific solution rather than a high-level "system constraint question" that drives the initial design phase. Option C (Address Validation) is a functional requirement regarding data integrity, but it does not define the architectural framework of the integration as effectively as latency does. By identifying the latency threshold, the architect can determine if the integration should be near real-time, hourly, or daily, which in turn influences how the system handles error recovery, retries, and transaction volumes.

NEW QUESTION: 37
Northern Trail Outfitters is planning to perform nightly batch loads into Salesforce using the Bulk API. The CIO is curious about monitoring recommendations for the jobs from the technical architect. Which recommendation should help meet the requirements?
A. Visually monitor in the Salesforce UI using the "Bulk Data Load Jobs" in Salesforce in the setup menu.
B. Set the Salesforce debug logs level to "finest", and add the user ID running the job to monitor in the
"Debug Logs" in the setup menu.
C. Write the error response from the Bulk API status to a custom error logging object in Salesforce using an Apex trigger, and create reports on the object.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 38
Northern Trail Outfitters (NTO) uses a custom mobile app to interact with its customers. One of the features of the app is Salesforce Chatter Feeds. NTO wants to automatically post a Chatter item to Twitter whenever the post includes the #thanksNTO hashtag.
Which API should an integration architect use to meet this requirement?
A. REST API
B. Streaming API to generate PushTopic
C. Connect REST API
Answer: C (LEAVE A REPLY)
When designing integrations that specifically involve Chatter, social collaboration, or Experience Cloud sites, the Connect REST API (formerly known as the Chatter REST API) is the architecturally recommended choice. While the standard REST API is optimized for CRUD operations on data records, the Connect REST API is specifically designed to handle the complex, nested structures of social feeds, including posts, comments, hashtags, and mentions. The Connect REST API provides a specialized resource for feed elements that simplifies the process of identifying specific social markers like hashtags. In this use case, the mobile app or a middleware service can subscribe to or query the feed. The Connect REST API returns structured

data where hashtags are identified as distinct message segments, making it trivial for an application logic to detect the #thanksNTO tag and trigger a subsequent call to the Twitter API. From an architectural standpoint, using the Connect REST API offers several advantages over the standard REST API or Streaming API for this requirement:
* Efficiency: Connect REST API responses are structured specifically for presentation in mobile applications, providing only the relevant social metadata and localized content needed for the feed.
* Feature Richness: It provides native support for social actions such as liking, sharing, and following, which are often required alongside feed monitoring.
* Scalability: It is designed to handle the high-volume, real-time nature of social interactions within Experience Cloud and mobile ecosystems.
While the Streaming API (Option B) can notify an application of record changes, it does not provide the rich, formatted social context that the Connect REST API delivers. The standard REST API (Option A) could technically access FeedItem records, but it would require significant custom parsing logic to identify hashtags within the raw body text, whereas the Connect REST API handles this segmenting natively. Therefore, for building custom mobile experiences that interact with Chatter data, the Connect REST API is the superior solution.

NEW QUESTION: 39
An enterprise customer is planning to implement Salesforce to support case management.



Below is their current system landscape diagram. Considering Salesforce capabilities, what should the integration architect evaluate when integrating Salesforce with the current system landscape?

A. Integrate Salesforce with Data Warehouse, Order Management and Email Management System.

B. Integrate Salesforce with Order Management System, Data Warehouse and Case Management System.

C. Integrate Salesforce with Email Management System, Order Management System and Case Management System.

Answer: A (LEAVE A REPLY)

An Integration Architect's primary responsibility when evaluating a landscape for a new Salesforce implementation is to identify the system of record for each business process and determine which legacy systems will be replaced by Salesforce. In this scenario, the customer is implementing Salesforce specifically to support case management.

According to the provided landscape diagram, the Case Management System currently exists as a standalone entity. Since Salesforce Service Cloud provides native, best-in-class case management capabilities, this legacy system is the primary candidate for retirement. Retiring the legacy Case Management system avoids data fragmentation and ensures that Salesforce serves as the single source of truth for support interactions.

However, for Salesforce to function effectively as a new case management hub, it must integrate with the remaining surrounding systems:

* Email Management System: This system likely handles inbound customer communications. An architect must evaluate integrating this with Salesforce (via Email-to-Case or a specialized connector) so that incoming emails automatically generate or update cases.
* Order Management System (OMS): Support agents often need to view order history or status to resolve customer inquiries. Integrating Salesforce with the OMS allows for a 360-degree view, enabling agents to see relevant order data directly within the Salesforce case console.
* Data Warehouse: For long-term reporting, trend analysis, and a unified customer profile, case data from Salesforce needs to be pushed to the Data Warehouse. This ensures that the Analytics and Business Intelligence Tool downstream can report on support metrics alongside other enterprise data.

Therefore, the architect should evaluate integrations with the Data Warehouse, Order Management, and Email Management System. Option B and C are incorrect because they suggest integrating with the "Case Management System," which is the very system being superseded by Salesforce's native capabilities. By focusing on the integration of these three supporting systems, the architect ensures a seamless transition where Salesforce is fully enriched with the necessary external data to drive support excellence.

NEW QUESTION: 40

Northern Trail Outfitters requires an integration to be set up between one of its Salesforce orgs and an External Data Source using Salesforce Connect. The External Data Source supports Open Data Protocol.
Which configuration should an integration architect recommend be implemented in order to secure requests coming from Salesforce?
A. Configure Special Compatibility for OData connection.
B. Configure CSRF Protection for OData connection.
C. Configure Identity Type for OData connection.
Answer: (SHOW ANSWER)
In the context of Salesforce Connect, securing the integration depends heavily on how the platform authenticates with the external system. The Identity Type configuration is the fundamental security setting for an External Data Source.
The architect must choose between two Identity Types:
* Named Principal: Salesforce uses the same set of credentials for all users to access the external system. This is simple to manage but does not allow the external system to distinguish between individual Salesforce users for auditing or permission purposes.
* Per User: Each Salesforce user must have their own credentials for the external system. This is the most secure option as it ensures that the data visible in Salesforce respects the specific permissions the user has in the source system.
By correctly configuring the Identity Type, the architect ensures that the requests coming from Salesforce are properly authorized at the target system. Option B (CSRF Protection) is a security measure to prevent cross- site request forgery but is not the primary mechanism for authenticating the Salesforce service itself. Option A is a technical compatibility setting for non-standard OData implementations and does not directly relate to security. Therefore, recommending the appropriate Identity Type-typically "Per User" for sensitive data-is the key step in securing the OData connection.

NEW QUESTION: 41
A company has an external system that processes and tracks orders. Sales reps manage their leads and opportunity pipeline in Salesforce. The company decided to integrate Salesforce and the Order Management System (OMS) with minimal customization and code. Sales reps need to see order history in real-time. The legacy system is on-premise and connected to an ESB. There are 1,000 reps creating 15 orders each per shift, mostly with 20-30 line items. How should an integration architect integrate the two systems based on these requirements?
A. Use Salesforce standard object, REST API, and extract, transform, load (ETL).
B. Use Salesforce custom object, custom REST API, and extract, transform, load (ETL).
C. Use Salesforce external object and OData connector.
Answer: C (LEAVE A REPLY)
To meet the requirements of minimal customization, low developer resources, and real-time visibility without data replication, the architect should utilize Salesforce Connect with External Objects and an OData connector.

Salesforce External Objects allow the OMS data to be viewed within Salesforce as if it were stored natively, but the data remains in the on-premise system. This fulfills the requirement for sales reps to see "up-to-date information" because every time they view the record, Salesforce Connect fetches the latest data via the ESB's OData endpoint. This Data Virtualization pattern is the most efficient choice for real-time history where users only need to view the data occasionally. Options A and B involve Data Replication via ETL, which would store the order data inside Salesforce.
Given the volume (15,000 orders/shift with 25 line items each = 375,000 records daily), this would rapidly consume Salesforce data storage limits and require significant custom development for the ETL logic and REST APIs. Furthermore, ETL is typically batch-oriented and would not provide the true "real-time" view requested. By using an OData connector, the architect leverages a declarative, "no-code" solution that satisfies the timeline constraints and provides immediate access to order details and line items without the cost of data storage.

NEW QUESTION: 42
The URL for a business-critical external service providing exchange rates changed without notice. Which solutions should be implemented to minimize potential downtime for users in this situation?
A. Remote Site Settings and Named Credentials
B. Enterprise Service Bus (ESB) and Remote Site Settings
C. Named Credentials and Content Security Policies
Answer: A (LEAVE A REPLY)
To minimize downtime when an external endpoint changes, an Integration Architect must ensure that the URL is not "hardcoded" within Apex code or configuration. The standard Salesforce mechanism for abstracting and managing external endpoints is Named Credentials.
Named Credentials specify the URL of a callout endpoint and its required authentication parameters in one definition. If the URL changes, an administrator simply updates the "URL" field in the Named Credential setup. This change takes effect immediately across all Apex callouts, Flows, and External Services that reference it, without requiring a code deployment or a sandbox-to-production migration.
Along with Named Credentials, Remote Site Settings (or the more modern External Website Configurations) are required. Salesforce blocks all outbound calls to URLs that are not explicitly whitelisted.
By having both in place, the remediation process is:
* Update the URL in the Named Credential.
* Update (or add) the new URL in the Remote Site Settings.
This approach follows the "Separation of Concerns" principle. Option B (ESB) could technically handle this, but it adds an extra layer of failure and complexity for a simple URL change. Option C (Content Security Policies) is used to control which resources (like scripts or images) a browser is allowed to load in the UI; it does not govern server-side Apex callouts. Therefore, the combination of Named Credentials and Remote Site whitelisting is the most efficient and standard way to provide architectural agility and minimize downtime.

NEW QUESTION: 43
A subscription-based media company's system landscape forces many subscribers to maintain multiple accounts and to log in more than once. An Identity and Access Management (IAM) system, which supports SAML and OpenID, was recently implemented to improve the subscriber experience through self-registration and single sign-on (SSO). The IAM system must integrate with Salesforce to give new self-service customers instant access to Salesforce Community Cloud.
A. OpenID Connect Authentication Provider and Just-in-Time (JIT) provisioning1
B. OpenID Connect Authentication Provider and Registration Handler2
C. SAML SSO and Registration Handler
Answer: B (LEAVE A REPLY)
To provide "instant access" and a seamless experience for Community (Experience Cloud) users, the architect must choose an authentication and provisioning strategy that handles user creation on-the-fly. While both SAML and OpenID Connect (OIDC) are viable for SSO, OpenID Connect is the modern standard for consumer-facing "Social" or external identity integrations because it is built on OAuth 2.0.
The critical component for "self-service" is the Registration Handler. When an OpenID Connect Authentication Provider is configured in Salesforce, you must associate it with an Apex class that implements the Auth.RegistrationHandler interface. This handler is executed during the SSO flow if the user does not already exist. It provides the architect with full programmatic control to:
* Match the incoming identity to an existing Contact or Account.
* Create a new Contact record if one doesn't exist.
* Provision a new User record with the correct Profile, Permission Sets, and Locale settings.
* Link the User to the correct Account hierarchy, which is vital for Community security models.
Option A suggests Just-in-Time (JIT) provisioning, which is a declarative way to create users. However, JIT is often too rigid for Experience Cloud requirements, as it has limited ability to perform complex data lookups or handle the specific linking of Contacts to Accounts required for external users. Option C is technically mismatched in common Salesforce terminology; while SAML uses JIT, the Registration Handler is the native, specific mechanism designed to work with Authentication Providers (like OIDC). By using B, the company ensures that a subscriber logging in for the first time via the IAM system is instantly and accurately provisioned in Salesforce, eliminating the need for multiple accounts.

NEW QUESTION: 44
Northern Trail Outfitters uses Salesforce to track leads and opportunities, and to capture order details.
However, Salesforce isn't the system that holds or processes orders. After the order details are captured in Salesforce, an order must be created in the Remote system, which manages the order's lifecycle. The integration architect for the project is recommending a remote system that

will subscribe to the platform event defined in Salesforce. Which integration pattern should be used for this business use case?

A. Request and Reply

B. Fire and Forget

C. Remote Call-In

Answer: (SHOW ANSWER)

In this scenario, Salesforce acts as the trigger for a business process that completes in an external system. The architect's recommendation for the remote system to subscribe to a platform event is the classic implementation of the Remote Process Invocation-Fire and Forget pattern.1 In a Fire and Forget pattern, Salesforce initiates a process by publishing a message (the event) to the event bus and then immediately continues its own2 processing without waiting for a functional response from the target system. The "Fire" part occurs when the order details are captured and the event is published; the

"Forget" part refers to Salesforce handing off the responsibility of order creation to the remote system. This pattern is ideal for improving user experience and system performance, as it avoids blocking the user interface while waiting for potentially slow back-office systems to respond.

Option A (Request and Reply) is incorrect because that would require Salesforce to make a synchronous call and wait for the remote system to confirm the order was created before allowing the user to proceed. Option C (Remote Call-In) is the inverse of what is described; it would involve the remote system actively reaching into Salesforce to "pull" the data, whereas here Salesforce is "pushing" the notification via an event stream.

By using Platform Events to facilitate this hand-off, Northern Trail Outfitters ensures a decoupled, scalable architecture where the remote system can process orders at its own pace while Salesforce remains responsive to sales users.

NEW QUESTION: 45

Northern Trail Outfitters (NTO) has recently changed its Corporate Security Guidelines requiring all cloud applications to pass through a secure firewall before accessing on-premise resources. NTO is evaluating middleware solutions. Which consideration should an integration architect evaluate before choosing a middleware solution?

A. The middleware solution enforces the OAuth security protocol.

B. The middleware solution is able to interface directly with databases via an Open Database Connectivity (ODBC) connection string.

C. The middleware solution is capable of establishing a secure API Gateway between cloud applications and on-premise resources.

Answer: C (LEAVE A REPLY)

When corporate guidelines mandate a firewall-protected entry point for cloud traffic, the middleware architecture must include a component capable of residing in a Demilitarized Zone (DMZ) or perimeter network. The architect must evaluate the solution's API Gateway capabilities.

A secure API Gateway acts as the intermediary that terminates external (cloud) TLS connections and inspects incoming traffic before proxying it to internal systems. It allows the security team to implement:
* IP Whitelisting: Ensuring only Salesforce's IP ranges can access the gateway.
* Mutual Authentication: Using certificates to verify that the request is genuinely coming from the Salesforce org.
* Rate Limiting: Protecting on-premise resources from being overwhelmed by cloud requests.
Option A (OAuth) is an authorization framework and does not satisfy the network-level firewall requirement on its own. Option B (ODBC) is an internal database protocol that should generally never be exposed to a cloud-facing firewall due to security risks. By prioritizing a solution with a hardened API Gateway, the architect ensures that NTO meets its new security mandates while providing a scalable and secure bridge for Salesforce to access back-office services.

NEW QUESTION: 46
A customer of Salesforce has used Platform Events to integrate their Salesforce instance with an external third-party artificial intelligence (AI) system. The AI system provides a prediction score for each lead that is received by Salesforce. Once the prediction score is received, the lead information is saved to Platform Events for other processes. The trigger on the Platform Events has failed ever since it was rolled out to production.
Which type of monitoring should the integration consultant have considered to monitor this integration?
A. Set up debug logs for Platform Event triggers to monitor performance.
B. Validate that the Platform Event definition matches lead's definition.
C. Monitor Platform Events created per hour limits across the Salesforce instance.
Answer: A (LEAVE A REPLY)
Troubleshooting failures in Platform Event-triggered logic is challenging because these triggers execute under the "Automated Process" system user, making them invisible to standard user-level monitoring. To diagnose why a trigger is failing in production, an Integration Architect must set up debug logs specifically for that trigger or the automated process user.
Debug logs provide a granular view into the execution execution path, including Apex errors, governor limit consumption, and specific DML failures. Without these logs, it is impossible to determine if the failure is due to a null pointer exception, a validation rule violation, or a record locking conflict.678 Option B is a design-time validation step; while important, it would not help mon9itor or troubleshoot a runtime failure in a deployed trigger. Option C focuses on high-level consumption limits; while reaching the
"Created Per Hour" limit would prevent events from1011 being published, it would not explain why an existing trigger is failing once the event has already arrived in the bus. By proactively establishing debug logs for the integration's triggers, the consultant can pinpoint the exact line of code or system constraint causing the failure, ensuring a faster "Mean Time to Repair" (MTTR) for critical AI-driven business processes.

NEW QUESTION: 47
Universal Containers (UC) is decommissioning its legacy CRM system and migrating data to Salesforce. The data migration team asked for a recommendation to optimize the performance of the data load. Which approach should be used to meet the requirement?
A. Contact Salesforce Support to schedule performance load.
B. Use Bulk API to process jobs in serial mode.
C. Use Bulk API to process jobs in parallel mode.
Answer: C (LEAVE A REPLY)
For large-scale data migrations, the Bulk API is the primary architectural tool for high-performance loading.
To maximize throughput and "optimize performance," the architect should recommend processing jobs in parallel mode.
In parallel mode, Salesforce processes multiple batches of a job simultaneously, taking advantage of the multi-tenant platform's concurrent processing capabilities. This significantly reduces the total time required for a massive data migration compared to serial mode (Option B), which processes batches one by one.
However, the architect must warn the team about potential lock contention. If multiple parallel batches attempt to update the same parent record or participate in complex sharing calculations at the same time,
"Unable to lock row" errors may occur. To mitigate this while maintaining parallel speed, the data should be sorted by Parent ID to ensure that batches do not overlap on the same records. Option A is rarely necessary for standard migrations unless the volume exceeds extreme thresholds. Parallel Bulk API is the standard "best practice" for ensuring the migration completes within the allotted cutover window.

NEW QUESTION: 48
Northern Trail Outfitters has had an increase in requests from other business units to integrate opportunity information with other systems from Salesforce. The developers have started writing asynchronous @future callouts directly into the target systems. The CIO is concerned about the viability of this approach and scaling for future growth. What should be done to mitigate the CIO's concerns?

A. Refactor the existing @future methods to use Enhanced External Services, import Open API 2.0 schemas, and update flows to use services instead of Apex.78
B. Implement an Enterprise Service Bus for service o9rchestration, mediation, routing, and decouple dependencies across systems.10
C. Implement an extract, transform11, load (ETL) tool and perform nightly batch data loads to reduce network traffic.
Answer: (SHOW ANSWER)
The CIO's concern regarding "viability" and "scaling" is rooted in the risks associated with tightly coupled, point-to-point integrations. Using @future methods for direct callouts creates a "spaghetti" architecture where Salesforce must manage the specific endpoints, authentication, and error logic for every external system.
The architect should recommend implementing an Enterprise Service Bus (ESB). An ESB acts as a centralized middleware layer that provides mediation, routing, and orchestration. By moving the integration logic to an ESB, Salesforce only needs to send a single message to the bus. The ESB then takes responsibility for delivering that data to multiple business units and external systems. This decouples Salesforce from the downstream systems; if a target system changes its API or is replaced, only the ESB configuration needs to be updated, not the Salesforce Apex code.
While External Services (Option A) provide a low-code way to call APIs, they still represent point-to-point connections and do not solve the broader orchestration and scaling challenges. ETL tools (Option C) are designed for bulk data movement and would not satisfy the need for the near real-time updates that the existing callout logic likely supports. An ESB provides the "quality of service" features-such as guaranteed delivery, retries, and protocol transformation-that are necessary for a growing enterprise to maintain a stable and scalable integration landscape.

NEW QUESTION: 49
A business requires automating the check and updating of the phone number type classification (mobile vs.
landline) for all incoming calls delivered to its phone sales agents. The following conditions exist:
* At peak, the call center can receive up to 100,000 calls per day.
* The phone number type classification is a service provided by an external service API.
* Business is flexible with timing and frequency to check and update the records (throughout the night or every 6-12 hours is sufficient).
A Remote-Call-In pattern and/or Batch Synchronization (Replication via ETL: System -> Salesforce) are determined to work with a middleware hosted on customer premise. In order to implement these patterns and mechanisms, which component should an integration architect recommend?
A. ConnectedApp configured in Salesforce to authenticate the middleware
B. An API Gateway that authenticates requests from Salesforce into the middleware (ETL/ESB)
C. Remote Site Settings configured in Salesforce to authenticate the middleware
Answer: (SHOW ANSWER)

In this scenario, the architecture involves a Remote-Call-In pattern or Batch Synchronization, where an external system (the middleware or ETL tool) initiates communication with Salesforce to update records. For any external system to securely access Salesforce APIs and perform these updates, it must be authenticated and authorized.
The Connected App is the foundational framework that allows an external application to integrate with Salesforce using APIs and standard protocols, such as OAuth 2.0 and SAML. By configuring a Connected App, the architect can define which permissions (Scopes) the middleware has, such as the ability to access data via the REST or Bulk API. This is the correct choice because the middleware needs to "log in" to Salesforce to push the phone classification data back into the Account or Contact records.
Option B, an API Gateway, is typically used to manage and secure requests going out of an organization to external services, or to provide a facade for on-premise APIs; it does not handle the inbound authentication into Salesforce itself. Option C, Remote Site Settings, is a configuration used solely to permit Salesforce to make outbound calls to a specific external URL (for example, if Salesforce were calling the phone classification service directly via Apex).
Given that the business is flexible with timing (allowing for nightly or 12-hour syncs) and handles 100,000 calls, a Batch Synchronization pattern via an ETL tool is highly efficient. The ETL tool will authenticate against the Connected App using a secure OAuth flow (such as the JWT Bearer Flow for server-to-server integration), retrieve the new phone numbers, call the external classification API, and then bulk-update the Salesforce records. This setup ensures a secure, scalable, and manageable integration that respects Salesforce's security architecture while meeting the high-volume data requirements of the call center.

NEW QUESTION: 50
When a user clicks "Check Preferences" as part of a Lightning flow, preferences from an externally hosted RESTful service are to be checked in real time. The service has OpenAPI 2.0 definitions. Which integration pattern and mechanism should be selected?
A. Data Virtualization: Salesforce Connect maps external REST data in external objects.
B. Request and Reply: Enhanced External Services invokes a REST API.
C. Remote Call-In: Salesforce REST API with REST Composite Resources.
Answer: (SHOW ANSWER)
This scenario describes a classic Request and Reply pattern where a user action in the UI requires an immediate, synchronous response from an external system to determine the next step in a business process (the Flow).
The requirement specifies that an OpenAPI 2.0 (Swagger) definition is available. For an Integration Architect, this is a prime use case for External Services. External Services allow you to import an OpenAPI schema and automatically generate "Invocable Actions" that can be used directly in Flow Builder without writing a single line of Apex code.
Why this is the best fit:
* Low Code: It fulfills the requirement purely through declarative configuration, which reduces maintenance and development costs.

* Real-Time: It performs a synchronous HTTP callout and waits for the Boolean/String values to be returned to the Flow variables.
* Type Safety: Because it uses the OpenAPI definition, Salesforce understands the data types (Boolean
/String) natively.
Option A (Data Virtualization) is more suitable for viewing and searching large external datasets as if they were records; it is over-engineered for a simple "check status" function. Option C (Remote Call-In) is the inverse of the requirement; it refers to an external system calling into Salesforce. By using Enhanced External Services, the architect provides a scalable, declarative solution that perfectly aligns with modern Salesforce development best practices for real-time external system interaction.

NEW QUESTION: 51
A new Salesforce program requires data updates between internal systems and Salesforce. Which relevant detail should an integration architect seek to solve for integration architecture needs?
A. Core functional and non-functional requirements for User Experience design, Encryption needs, Community and license choices
B. Integration skills, SME availability, and Program Governance details
C. Timing aspects, real-time/near real-time (synchronous or asynchronous), batch and update frequency
Answer: C (LEAVE A REPLY)
In the "Discovery" phase of integration architecture, the architect must translate abstract business needs into technical requirements. The most critical variables that define the Integration Pattern are Timing and Volume.
An architect cannot choose between the REST API, Streaming API, Bulk API, or Outbound Messaging without knowing:
* Latency Requirements: Does the business need the update in 200 milliseconds (Synchronous), 2 minutes (Near Real-Time), or 24 hours (Batch)?
* Frequency: Is the data updated every time a user clicks a button, or once at the end of the day?
* Volume: Are we moving 10 records at a time or 10 million?
Option A focuses on UI/UX and licensing, which are project management concerns. Option B focuses on resource allocation and governance. While important for the project, they do not inform the technical design of the data flow.
By specifically seeking out Timing aspects (Synchronous vs. Asynchronous) and Update Frequency, the architect can apply the Salesforce Integration Decision Matrix. For instance, a "Real-time" requirement for small volumes leads to a Request-Reply pattern via Apex Callouts. A "Nightly" requirement for large volumes leads to a Batch Data Synchronization pattern via the Bulk API. Identifying these "Non-Functional Requirements" (NFRs) early is the only way to ensure the architecture is scalable and stays within platform governor limits.

NEW QUESTION: 52
Northern Trail Outfitters needs to present shipping costs and estimated delivery times to its customers.
Shipping services used vary by region and have similar but distinct service request parameters.
Which integration component capability should be used?
A. Apex REST Service to implement routing logic to the various shipping service
B. Enterprise Service Bus to determine which shipping service to use and transform requests to the necessary format
C. Enterprise Service Bus user interface to collect shipper-specific form data
Answer: B (LEAVE A REPLY)
When dealing with multiple external service providers (like different regional shippers) that have varying API requirements, the most scalable architectural choice is an Enterprise Service Bus (ESB) or middleware solution. This scenario describes a classic "Orchestration" and "Transformation" use case.
An ESB excels at:
1. Routing/Mediation: The ESB can receive a single, standardized request from Salesforce ("Get Shipping Rates") and use business logic to determine which regional carrier's API should be called.
2. Transformation: Each carrier likely has a distinct XML or JSON schema. The ESB can transform the common data model sent by Salesforce into the specific format required by the chosen shipper.
3. Protocol Bridging: If one shipper uses SOAP and another uses REST, the ESB handles these technical differences, allowing Salesforce to interact with a single, simplified interface.
Option A (Apex REST) would require the Salesforce team to maintain custom code for every new shipping provider added, leading to high technical debt and maintenance challenges. Option C is incorrect because an ESB is a back-end infrastructure component, not a user interface tool.
By utilizing an ESB, Northern Trail Outfitters decouples Salesforce from the complexities of the shipping providers' APIs. This "Hub and Spoke" model means that adding a new regional shipper in the future only requires configuration within the ESB, rather than a full code deployment in Salesforce. This provides the agility and architectural separation necessary for a global retail operation.

NEW QUESTION: 53
A customer is migrating from an old legacy system to Salesforce. As part of the modernization effort, the customer would like to integrate all existing systems that currently work with its legacy application with Salesforce. Which constraint/pain-point should an integration architect consider when choosing the integration pattern/mechanism?
A. System types APIs, File systems, Email
B. Reporting and usability requirements
C. Multi-language and multi-currency requirement
Answer: A (LEAVE A REPLY)

When migrating from a legacy landscape to a modern platform like Salesforce, the most immediate technical hurdle is the diversity of system types and communication protocols used by the existing systems.
In a legacy environment, integrations are often not standardized. An architect may encounter systems that communicate via modern REST/SOAP APIs, but they will also likely find older systems that rely on Flat File exchanges (FTP/SFTP), Email-based triggers, or direct Database connections. These "System Types" are a fundamental constraint because they dictate the choice of integration middleware. For example, Salesforce cannot natively poll a file system or read an on-premise database; therefore, an architect must identify these constraints to justify the need for an ETL or ESB tool that can bridge these legacy protocols with Salesforce's API-centric architecture.
While reporting (Option B) and multi-currency (Option C) are important functional requirements for the Salesforce implementation, they do not dictate the integration pattern (e.g., Request-Reply vs. Batch) as much as the technical interface of the source/target systems does. By evaluating the APIs, file systems, and email capabilities of the legacy landscape first, the architect ensures that the chosen integration mechanism- whether it be the Streaming API, Bulk API, or middleware orchestration-is technically capable of actually communicating with the legacy debt.

NEW QUESTION: 54
Northern Trail Outfitters (NTO) is planning to create a native employee-facing mobile app with the look and feel of Salesforce Lighting Experience. The mobile app needs to integrate with NTO's Salesforce org. Which Salesforce API should be used to implement this integration?
A. Connect REST API
B. REST API
C. User Interface API
Answer: (SHOW ANSWER)
When building custom mobile or web applications that aim to replicate the look and feel of Salesforce Lightning Experience, the User Interface (UI) API is the architecturally recommended choice.
The UI API is specifically designed to provide the metadata and data needed to build high-fidelity user interfaces. Unlike the standard REST API (Option B), which returns raw record data, the UI API returns both data and metadata in a single response. This includes information about page layouts, field-level security, picklist values, and localized labels. By using the UI API, the mobile app can dynamically render fields according to the user's permissions and the organization's layout configurations, ensuring that the custom app stays in sync with changes made in Salesforce Setup without requiring code updates in the mobile app.
Connect REST API (Option A) is primarily used for Chatter, Communities (Experience Cloud), and CMS content, and while it is useful for those specific social features, it does not provide the layout and record-level metadata required for a full CRM interface. The UI API is the same underlying technology that powers the Salesforce mobile app and Lightning Experience itself. Therefore, utilizing this API allows NTO's developers to build a native app that perfectly mimics

the Lightning Experience while reducing the amount of custom logic needed to handle complex Salesforce UI requirements.

NEW QUESTION: 55
Northern Trail Outfitters has recently implemented middleware for orchestration of services across platforms.
The Enterprise Resource Planning (ERP) system being used requires transactions be captured near real-time at a REST endpoint initiated in Salesforce when creating an Order object.
Additionally, the Salesforce team has limited development resources and requires a low-code solution. Which option should fulfill the use case requirements?
A. Use Lightning Flow to create a platform event, selecting the record type as the platform event name on insert of record.
B. Implement Change Data Capture on the Order object and leverage the replay ID in the middleware solution.
C. Use Remote Process Invocation fire and forget pattern on insert on the order object using Flow Builder.
Answer: C (LEAVE A REPLY)
To satisfy a requirement for near real-time REST updates with limited development resources, the architect should utilize Flow Builder. Flow Builder is Salesforce's primary low-code tool for automating complex business logic and outbound integrations.
The Remote Process Invocation-Fire and Forget pattern is the most efficient way to signal an external system (or middleware) that a record was created without blocking the user. Using a Record-Triggered Flow on the Order object, the architect can configure an Action (such as an External Service or a simple HTTP Callout) to send the order data to the middleware's REST endpoint.
Option A is slightly incorrect because creating a platform event is just one step in an event-driven flow; the
"Fire and Forget" pattern more accurately describes the end-to-end intent. Option B (Change Data Capture) is a powerful tool, but it is considered a "pro-code" or high-configuration so12lution on the middleware side, requiring the middleware to manage Replay IDs and Bayeux subscriptions. Option C leverages the native strengths of Flow to fulfill the requirement declaratively, 13allowing the team to14 deliver a functional integration without writing Apex code while meeting the near-real-time performance expectations of the ERP.