ExamsIndex
Practice Materials & Tests

DEMO VERSION

Salesforce

AP-223 Exam

CPQ and Billing Consultant Accredited Professional

Exam Latest Version: 8.0

Question 1. (Single Select)

You are implementing the Design Document for a large Enterprise Revenue Cloud project having multiple lookup price rules supporting a complex pricing requirement in the Build phase. During construction the customer discovers additional logic and external data stores that need to be incorporated in order to achieve the correct pricing in a particular set of use cases. You estimate the lookup price rules will need to be modified, additional rules will need to

be created and API development will be needed. As an Implementation consultant what is the appropriate course of

action that should take in this predicament?

A: Communication to the customer ongoing adjustment can be made as long as we're in the build phase.
B: Implement the lookup price rules immediately then review with the solution Architect.
C: Communicate these changes to the project manager who will evaluate the impact to scope, timeline and budget them determine the next course of action
D: Consult with the solution Architect first who will expedite the updates to the design documents, then implement the changes immediately.
E: Gather more details, if it requires a low level of effort then implement immediately before starting the next sprint. Otherwise Complete on the subsequent sprint.

Correct Answer: C

Explanation:

For a large Enterprise Revenue Cloud (Salesforce CPQ + Billing) implementation, the key themes in all Salesforce delivery guidance and project best practices are:

Governance and change control

Design-first, then build

Raising scope-impacting changes through the Project Manager

Architect accountability for solution integrity, PM accountability for scope/timeline/budget

Let’s walk through why C is correct and why the other options conflict with typical Salesforce

CPQ/Billing implementation best practices.

1. Context of the ScenarioYou are in the Build phase and:

You already have a design with:

Multiple Lookup Price Rules implementing complex pricing.

New information emerges:

Additional pricing logic

External data stores that must be incorporated

Need to modify existing lookup rules

Need to create additional rules

Need API development (integration work)

This is not a cosmetic tweak; it is:

Scope-impacting (new integration/API work, new logic)

Design-impacting (pricing architecture changes)

Potentially timeline and budget impacting

Therefore, this triggers formal change control.

2. Why Option C is CorrectC. Communicate these changes to the project manager who will evaluate the impact to scope, timeline and budget then determine the next course of action

This aligns with standard Salesforce implementation and project governance principles:

Any change that affects scope, complexity, or integration must be raised to the Project Manager (PM)

Project Manager is responsible for:

Scope management

Timeline & milestones

Budget & resourcing

Managing change requests and stakeholder approvals

The PM will:

Evaluate impact with:

Solution Architect (for technical/design impact)

Tech leads / Dev leads (for effort estimation)

Decide:

Whether a Change Request (CR) is needed

How to re-prioritize sprints, adjust backlog

Whether additional budget / time is required

How to communicate to customer stakeholders

This preserves:

Design integrity (Architect still evaluated the solution)

Project discipline (PM governs scope/timeline/budget)

Traceability and documentation (updated design docs, backlog, CRs)

This is exactly how a large enterprise Revenue Cloud (CPQ + Billing) program is expected to run.

3. Why the Other Options Are Not AppropriateA. “Adjust as long as we're in build phase”A. Communication to the customer ongoing adjustment can be made as long as we're in the build phase.

Problems:

Implies uncontrolled scope creep:

“As long as we’re in build, we can just keep adjusting.”

No mention of:

Impact to scope, timeline, budget

Formal change control

Involvement of PM or Architect

In a complex CPQ/Billing implementation, this would:

Break governance

Risk missed deadlines and budget overruns

Create misaligned expectations with the customer

So A contradicts standard methodology and enterprise delivery practices.

B . “Implement then review with the Solution Architect”B. Implement the lookup price rules immediately then review with the solution Architect.

Problems:

Sequence is wrong:

You never build first and ask the Architect later on large-scale pricing and integration changes.

This can cause:

Misalignment with overall pricing architecture

Conflicts with other CPQ/Billing components (e.g., Amendments, Renewals, Billing logic)

Rework if the Architect has a different approach

Still no mention of PM or scope/timeline/budget impact.

This violates both design governance and project governance.

D . “Architect then immediate implementation (no PM)”D. Consult with the solution Architect first who will expedite the updates to the design documents, then implement the changes immediately.

This is closer, but still incomplete:

Good:

You involve the Solution Architect.

You talk about updating design documents.

But:

No involvement of the Project Manager.

No consideration of:

Impact to scope

Impact to timeline

Impact to budget

For “large Enterprise Revenue Cloud” projects, Architect "` PM:

Architect owns technical solution integrity

PM owns project plan, change control, stakeholder approvals

So D ignores formal change management which is critical at enterprise scale.

E . “If low effort, just do it; else next sprint”E. Gather more details, if it requires a low level of effort then implement immediately before starting the next sprint. Otherwise complete on the subsequent sprint.

Problems:

Consultant is unilaterally deciding based on “low effort”:

No PM.

No formal scope/time/budget impact evaluation.

This might be okay for minor cosmetic or non-functional changes in a small project, but:

Here we have:

Complex pricing

Multiple lookup price rules

External data store integrations

API development

This is never “just low effort”.

For a large enterprise Revenue Cloud implementation:

This bypasses governance, change control, and approvals.

So E promotes ad hoc scope changes, which is against standard practice.

4. How This Ties Back to Salesforce CPQ & Billing Best PracticesIn Salesforce CPQ and Billing implementations, especially when dealing with complex pricing logic and external integrations:

Complex Pricing (Lookup Price Rules):

Changes can affect:

Quote calculation performance

Sequential dependencies with Price Rules, Discount Schedules, QCP, Billing logic

May cause downstream issues in:

Orders, Invoices, Revenue Schedules, Amendments, Renewals

External Data Stores & API Development:

Introduces:

New integration patterns

Error handling, retries, timeouts

Security and governance requirements

Impacts:

Technical design

Test strategy (SIT, UAT, performance testing)

Possibly non-functional requirements

Because of that, Salesforce project documentation and implementation guidance emphasize:

Raising such changes via Project Manager

Having the Solution Architect assess and update:

Solution design

Integration architecture

Managing it formally as a change request if it affects:

Scope

Timeline

Budget

This is exactly what Option C describes at the right level of responsibility.

Question 2. (Multi Select)

What are three risks when using too many cross object formula fields in a Revenue Cloud Project?

A: Formula field data is not always available during CPQ quote calculation
B: Formula fields have unlimited access to object many relationships away which makes it vulnerable to data changes.
C: They are computationally Expensive.
D: They can easily exceed limits if not carefully designed and tested
E: Formula Fields are editable, after the calculation completes the sales user or process automation can overwrite its value

Correct Answer: A, C, D

Explanation:

In Salesforce CPQ + Billing (Revenue Cloud), heavy use of cross-object formula fields can create serious performance, calculation, and reliability issues. Salesforce product documentation and CPQ study guides highlight several risks related to:

Quote calculation engine performance

SOQL query depth

Runtime evaluation limits

Data availability timing during synchronous calculations

Below is the breakdown of the options:

' A. Formula field data is not always available during CPQ quote

Salesforce CPQ reads values at calculation time, but cross-object formula fields may:

Not resolve in time if they depend on parent records updated within the same transaction

Return stale values because formula evaluation is not recalculated in real time mid-calculation

Fail during QCP or price rule evaluation due to record access/state issues

This is a known risk documented in CPQ technical architecture guidance.

'L B. Formula fields have unlimited access to object many relatio
vulnerable to data changes.Incorrect.

Formula fields do NOT have unlimited access. They are limited to 10 relationship levels.

While data changes on parent objects can affect formula results, this is not a primary risk emphasized in Revenue Cloud implementation guidance.

Therefore, not one of the three correct risks.

' C. They are computationally expensive.Correct.

Formula fields—especially cross-object ones—are recalculated at runtime every time:

The referenced record is queried

CPQ calculator reads them during price rule evaluation

Billing processes (Invoice Run, Usage Rating, etc.) reference them

This can significantly slow down:

Quote calculations

Order/Invoice generation

Any multi-object SOQL-heavy logic

This is a well-known performance risk.

' D. They can easily exceed limits if not carefully designed and

Cross-object formulas contribute to:

SOQL query depth limits

CPU time limits

Formula size complexity

Relationship depth limits

In CPQ/Billing, where Quote and Quote Line processing already push platform limits, too many formula fields can cause:

Calculation failures

Invoice/Order creation errors

Apex limit exceptions

Salesforce documentation warns against heavy formula usage for precisely these scalability concerns.

'L E. Formula fields are editable, after calculation a user/proces
valueIncorrect.

Formula fields are never editable by users or automation.

Their values are dynamically calculated from their formula expressions.

Therefore, this option is not a valid risk.

Question 3. (Single Select)

How can a Revenue Cloud Consultant create a new payment Method for a credit card that will be saved for future Payments?

A: Enter the credit card details into a new payment Method record Click the Tokenize button
B: From the Payment credit cards related list, click the new credit card button.
C: Enter the credit card details into a new payment method record. salesforce users should use platform encryption for PCI Compliance.
D: From the Account, Payment Method related list, then click the new Payment Method Credit Card button.

Correct Answer: D

Explanation:

To save a new credit card Payment Method for future payments, the correct Salesforce Billing process is:

Correct documented methodFrom the Account Page:

Go to the Payment Methods related list

Click New Payment Method – Credit Card

Enter card details

Card is tokenized (via Payment Gateway)

Saved for future payments

This is exactly what option D describes.

Why the other answers are incorrectOption

Why Incorrect

A . Tokenize button

Outdated UI/legacy workflow; new UI and gateways tokenize automatically.

B . Payment credit cards related list

Not the standard Billing object structure; Salesforce Billing uses Payment Method object, not "Payment Credit Card".

C . Enter card details + encryption

PCI does not allow storing full credit card numbers in Salesforce even with Platform Encryption — credit cards must be tokenized via gateway, not stored directly.

Therefore:

The only correct Salesforce Billing approach is D.

Question 4. (Single Select)

A Revenue Cloud customer has posted an invoice and now wants to add on more items from another order associated to that account. Without using invoice batches, how can this be accomplished?

A: Credit the invoice, add the new order and run an invoice scheduler to pick all the orders up.
B: use bill now on the new order and reparent the new invoice lines to the existing invoice C . Cancel and Rebill the invoice, add the new Order and run an invoice scheduler to pick all the order up.
C: Use bill now on the new Order and consolidate the invoices.

Correct Answer: C

Question 5. (Multi Select)

What are three risks when using too many cross object formula fields in a Revenue Cloud

Project?

A: Formula field data is not always available during CPQ quote calculation
B: Formula fields have unlimited access to object many relationships away which makes it vulnerable to data changes.
C: They are computationally Expensive.
D: They can easily exceed limits if not carefully designed and tested
E: Formula Fields are editable, after the calculation completes the sales user or process automation can overwrite its value

Correct Answer: A, C, D

Explanation:

In Salesforce CPQ + Billing (Revenue Cloud), heavy use of cross-object formula fields can create serious performance, calculation, and reliability issues. Salesforce product documentation and CPQ study guides highlight several risks related to:

Quote calculation engine performance

SOQL query depth

Runtime evaluation limits

Data availability timing during synchronous calculations

Below is the breakdown of the options:

' A. Formula field data is not always available during CPQ quote

Salesforce CPQ reads values at calculation time, but cross-object formula fields may:

Not resolve in time if they depend on parent records updated within the same transaction

Return stale values because formula evaluation is not recalculated in real time mid-calculation

Fail during QCP or price rule evaluation due to record access/state issues

This is a known risk documented in CPQ technical architecture guidance.

'L B. Formula fields have unlimited access to object many relatio
vulnerable to data changes.Incorrect.

Formula fields do NOT have unlimited access. They are limited to 10 relationship levels.

While data changes on parent objects can affect formula results, this is not a primary risk emphasized in Revenue Cloud implementation guidance.

Therefore, not one of the three correct risks.

' C. They are computationally expensive.Correct.

Formula fields—especially cross-object ones—are recalculated at runtime every time:

The referenced record is queried

CPQ calculator reads them during price rule evaluation

Billing processes (Invoice Run, Usage Rating, etc.) reference them

This can significantly slow down:

Quote calculations

Order/Invoice generation

Any multi-object SOQL-heavy logic

This is a well-known performance risk.

' D. They can easily exceed limits if not carefully designed and

Cross-object formulas contribute to:

SOQL query depth limits

CPU time limits

Formula size complexity

Relationship depth limits

In CPQ/Billing, where Quote and Quote Line processing already push platform limits, too many formula fields can cause:

Calculation failures

Invoice/Order creation errors

Apex limit exceptions

Salesforce documentation warns against heavy formula usage for precisely these scalability concerns.

'L E. Formula fields are editable, after calculation a user/proces
valueIncorrect.

Formula fields are never editable by users or automation.

Their values are dynamically calculated from their formula expressions.

Therefore, this option is not a valid risk.

ExamsIndex

Demo PDF Complete

Your AP-223 Demo (5 Questions)

Get the Complete Version

Full Questions with Detailed Explanations

Interactive Web-Based Exams Available

To get 30% off, use Coupon Code: NEWYEAR30

https://examsindex.com/exam/ap-223