Boost up Your Certification Score

SAP
C_ACDET
SAP Certified - Configuring SAP Ariba Strategic Sourcing Suite

Exams Boost
Boost Up Your Career

For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/c-acdet

Latest Version: 4.1

1. Unified Scenario Simulation
2. Micro Skill Drill Exam

Topic: 1
Unified Scenario Simulation
Velmark Driveline Brownfield Sourcing Cutover Across Realms
Business Context
Velmark Driveline Components is a global automotive-parts manufacturer headquartered in Stuttgart, supplying transmission and driveline assemblies to vehicle producers across Europe, North America, and Southeast Asia. Its category and sourcing teams run competitive events for forged components, electronic control units, and bonded rubber assemblies, with regional buyers operating semi-autonomously under a shared category governance council. The company is roughly eighteen months into modernizing how it runs strategic sourcing, having consolidated several legacy regional tools onto SAP Ariba Strategic Sourcing Suite.
System Landscape
The landscape is organized as a multi-realm estate. A parent Procurement realm holds the corporate category structure, supplier hierarchy, and the bulk of shared field customizations, while child realms in North America and Southeast Asia carry region-specific master data, supplier records, and a handful of locally added fields. Production has been live for the European business for almost a year, and the regional child realms are now being brought onto the same shared design. Configuration work has been built and validated in dedicated test sites, and a best-practice configuration baseline was prepared earlier in the program and shared down to those test sites.
Implementation Progress
The program is in the closing stretch of Realize and preparing to enter Deploy. Because the European production realm is already in service, the team has determined that this wave is a brownfield migration onto an already-live site rather than a fresh build. The technical lead has assembled an Intelligent Configuration Manager package covering the latest object-model and field changes, and the cutover plan sequences those changes against an active business calendar. Several sourcing events are mid-flight in production, and the category council has been explicit that live events cannot be disrupted by the migration.
Object Model Context
A central piece of this wave is a revision to how sourcing objects are filtered. The category team has asked that, when a buyer selects a commodity on a forged-component event, only the suppliers qualified for that commodity appear, rather than the full supplier list. The design achieves this by adjusting existing relations through relation types and applying relation entries in field configuration, so that native object filtering narrows the selectable set. The change was modeled and tested in the test site against representative classes, and the technical lead has noted that the suite's object model uses type casting across related classes.
Master Data Context
The filtered supplier field is backed by flex master data defined on the realm. In the parent test site, the commodity-to-supplier relation entries were populated alongside the flex records, and the field behaved as intended. In one of the child test sites, a regional analyst loaded an updated supplier set late in the cycle, and during a rehearsal the filtered field showed fewer selectable suppliers than the buyers expected for two regional commodities. The same commodities behaved correctly in the parent test site. No code was changed in either site; the difference appeared after the regional load.

Visit us at: https://www.examsboost.com/test/c-acdet

Field and Form Context

The same forged-component design adds two custom fields that are surfaced on a published sourcing form template, and one of them is also expected to appear in standard reporting. During a pre-cutover review, a reporting analyst noted that a custom field visible on the form did not appear when she tried to build a report over recent events. A separate reviewer observed that a newly translated field label rendered correctly in the parent realm's primary language but fell back to the technical field name in one regional language during the same rehearsal.

Translation and Reporting Context

Velmark supports several languages across its regions, and the sourcing sites maintain translation files for custom and standard field labels and tooltips. The regional rollout added one additional language to a child site late in the program. The custom-field metadata for reporting and the additional-language support were both handled as separate steps from the core field configuration, and the program manager has flagged that not every step in that chain was confirmed before the rehearsal.

Governance Context

The category governance council and the platform team hold different priorities for the cutover window. The council wants the regional realms aligned to the shared design quickly so that quarterly sourcing targets are not delayed, while the platform team wants every change routed through the test-site-to-production path with a confirmed production-availability check rather than any direct adjustment on the live realm. A regional administrator, under time pressure, has proposed making one of the field corrections directly in the production child realm to save a migration cycle, and the platform lead has pushed back. The cutover plan also has to account for the parent realm holding shared definitions that the child realms inherit.

Close Preparation

With the migration window approaching, the team must reconcile the rehearsal observations against the packaged configuration, confirm that the dependent data behind each customized field is complete, and decide what is safe to migrate onto the live realms versus what needs another validation pass. The decisions made in this window determine whether the regional buyers see a consistent, correctly filtered, fully reportable sourcing experience on day one, or whether the brownfield cutover carries hidden gaps into production.

Your assignment

You are acting as the configuration consultant supporting Velmark Driveline Components during the closing Realize activities and the move into Deploy for its regional sourcing rollout. The European parent realm is already live in production, and the North America and Southeast Asia child realms are being aligned to the shared design through a brownfield migration. Configuration has been built and validated in dedicated test sites, and an Intelligent Configuration Manager package is assembled for cutover. Your task is to advise on how to interpret the rehearsal observations, confirm the dependencies behind each customized field, and sequence the migration onto the live realms without disrupting in-flight sourcing events. You will reason about configuration decisions and recommend the soundest direction at each point; you will not key configuration directly

1. CHALLENGE 1 — Native Object Filtering Through Relation Types

• Advise on how the supplier field is narrowed to the suppliers qualified for a selected commodity, reasoning from the object model's relations and relation types rather than from custom code.

• Distinguish a relation-design cause from a data cause when the filtered field behaves differently across realms.

• Account for the suite's use of type casting across related classes when reasoning about why a filter returns a given object set.

Checkpoints

Visit us at: https://www.examsboost.com/test/c-acdet

• Confirm the consuming field renders the intended narrowed set for a representative commodity in the test site.
• Decide whether divergent behavior points to the relation entries and types or to the data those entries reference.

2. CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields

• Advise on the dependency between the flex master data on the realm, the relation entries, and the options the filtered field renders.
• Reason about why a field that worked in the parent test site shows fewer options in a child test site after a regional data load, with no configuration change.
• Recommend how to preserve live regional master data while aligning the realm to the shared filtered-field design.

Checkpoints

• Confirm the rendered supplier set matches the expected qualified suppliers per commodity in the affected realm.
• Decide what must be present and current in the realm's flex master data before the field is relied upon.

3. CHALLENGE 3 — Custom Field Reporting and Translation Dependencies

• Advise on why a custom field that is usable on a published form template may still be absent from reporting, and what step makes it available there.
• Reason about why a translated label renders in one language but falls back to the technical field name in another.
• Recommend the steps that make a custom field both reportable and correctly labeled across every supported regional language.

Checkpoints

• Confirm the custom field is selectable when building a report over recent events.
• Confirm each supported language renders the intended label rather than the technical field name.

4. CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms

• Advise on sequencing the Intelligent Configuration Manager deployment onto an already-live parent realm with dependent child realms while sourcing events are in flight.
• Reason about the role of the production-availability check and the test-site-to-production path when a regional administrator proposes a direct production change.
• Recommend how shared parent definitions and child inheritance shape the order in which changes are migrated.

Checkpoints

• Confirm the production-availability check is satisfied before migration commences.
• Decide the migration sequence that aligns the realms without disrupting in-flight events or leaving realm drift.

Question: 1

CHALLENGE 1 — Native Object Filtering Through Relation Types

The category team wants only suppliers qualified for the selected commodity to appear on the forged-component event.
Which approach best describes how the selectable supplier set should be narrowed in this design?

A. Adjust existing relations through relation types and apply relation entries in field configuration for the field

B. Add a custom validation script that rejects unqualified suppliers after they are selected on each commodity event
C. Create a separate supplier list field for each commodity, then hide the lists that do not match the selection
D. Remove unqualified suppliers from the realm so only qualified records remain available to every sourcing event

Answer: A

Explanation:
Native object filtering is changed by adjusting existing relations via relation types and applying relation entries in field configuration, which is exactly how the consuming field is narrowed to the qualified set. This keeps the behavior in the supported object-model configuration rather than in code or data deletion.

Question: 2

CHALLENGE 1 — Native Object Filtering Through Relation Types
The filtered field returns the correct set in the parent test site but an unexpected set for two commodities in a child test site, and no configuration was changed between them.
What is the most defensible first interpretation of this divergence?

A. The object-model classes were re-cast differently in the child realm, so type casting must be rebuilt first of all
B. The relation entries or referenced data differ in the child realm, so the cause sits in the entries or data
C. The form template publishing failed in the child realm, so the consuming field is reading a stale definition now
D. The reporting metadata pull was skipped, so the field cannot resolve its selectable options at selection time

Answer: B

Explanation:
Because the relation design is shared and unchanged, divergent output for specific commodities points to the relation entries or the underlying data differing per realm, not to the design itself. This directs diagnosis to the realm-specific layer rather than re-engineering the model.

Question: 3

CHALLENGE 1 — Native Object Filtering Through Relation Types
A reviewer asks why the suite can apply one relation-based filter consistently across related sourcing classes rather than configuring it separately on each class.
Which characteristic of the object model best explains this?

A. Each class stores an independent copy of the relation, so the filter is duplicated and synced by a scheduled job

B. The classes are unrelated to one another, so the same filter must be reapplied manually wherever it is needed
C. The object model is a class structure with relationships that supports type casting across related classes
D. Filtering is a global realm switch that overrides individual class behavior regardless of any defined relationships

Answer: C

Explanation:
The object model is a class structure with defined relationships and supports type casting across related classes, which is what lets a relation-based filter apply consistently across them. Reasoning from this structure explains the shared behavior without per-class duplication.

Question: 4

CHALLENGE 1 — Native Object Filtering Through Relation Types
The technical lead wants to confirm the narrowed filter is correct before packaging it for migration.
Which validation best demonstrates that the relation-based filtering behaves as intended?

A. Confirm the migration package deploys without any errors into the production realm during the cutover window
B. Confirm the custom field appears in the reporting field list once the custom-field metadata pull has been run
C. Confirm the field label is translated in every supported regional language on the published sourcing form template
D. Confirm a representative commodity renders the qualified supplier set in the consuming field in the test site

Answer: D

Explanation:
The filtering is validated by observing that the consuming field renders the intended narrowed set for a representative commodity in the test site, which directly evidences the relation and entries are correct. This is the observable completion evidence for this challenge.

Question: 5

CHALLENGE 1 — Native Object Filtering Through Relation Types
The team debates whether the forged-component filter should be built fresh in each child realm or carried from the shared design.
Which direction best preserves consistency across the multi-realm estate?

A. Carry the shared relation design from the validated baseline and supply each realm's own entries and data

B. Rebuild the relation configuration independently in every child realm so each region can freely diverge as needed
C. Configure the filter only in the parent realm and rely on each event to fetch it remotely from there at runtime
D. Replace the relation-based filter with a manual supplier shortlist maintained by each regional buyer per commodity

Answer: A

Explanation:
Carrying the validated shared relation design while supplying each realm's own relation entries and data keeps the design common across the estate and confines variation to the data layer. This is what preserves a consistent filtered experience across parent and child realms.

Question: 6

CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields
In a child test site the filtered field showed fewer suppliers than expected for two commodities after a regional analyst loaded an updated supplier set, while the parent site was correct and no configuration changed.
What is the most likely root cause?

A. The relation types were corrupted by the regional data load and now must be redefined in the shared object model
B. The form template lost its database pre-allocation for the affected fields when the regional supplier set was loaded
C. The flex master data or relation entries the field depends on are incomplete or stale after the regional load
D. The additional regional language was not added, so the field hid the suppliers whose labels could not be rendered

Answer: C

Explanation:
The filtered set depends on the flex master data and relation entries being present and current, so a late regional load that left those incomplete or stale yields a short rendered set with no configuration change. The data layer, not the design, explains the symptom.

Question: 7

CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields
The team must decide what to verify before relying on the filtered field in the affected child realm.
Which check most directly confirms the field is trustworthy for that realm?

A. The migration package containing the custom field deployed without any error into the affected child realm

B. The custom-field metadata pull was run for the field so that the field is available and selectable in reporting
C. The field label is present in the realm's primary language on the published sourcing form for the affected events
D. The rendered supplier set matches the expected qualified suppliers per commodity using that realm's current data

Answer: D

Explanation:
Trust in the filtered field comes from confirming its rendered set matches the expected qualified suppliers per commodity against the realm's current flex data. This validates the data layer the field actually depends on for that realm.

Question: 8

CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields
The filtered supplier field is described as backed by flex master data on the realm.
What does this dependency imply for how the field behaves across realms?

A. The field's options are fixed in the shared design and are identical in every realm regardless of local data
B. The field renders from each realm's own flex records, so its options reflect that realm's data despite a shared design
C. The field reads supplier data live from the parent realm, so every child realm always mirrors the parent set exactly
D. The field stores its own static option list at publish time and then ignores any later changes to the flex records

Answer: B

Explanation:
Because the field is backed by flex master data defined on the realm, it renders from each realm's own flex records, so a shared design still produces realm-specific options. This is precisely why a complete, current regional load matters per realm.

Question: 9

CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields
The category council insists live regional supplier data must be preserved while the child realm adopts the shared filtered-field design.
Which direction best satisfies both?

A. Wipe the child realm's supplier data and reload only the parent realm's records so the two realms match exactly

B. Adopt the shared design and ensure each realm's flex data and relation entries are complete and current as well
C. Skip the shared design in the child realms entirely so that their existing regional supplier data is not touched
D. Freeze all regional master data so that no further supplier loads can occur in any child realm until after cutover

Answer: B

Explanation:
Adopting the shared design while ensuring each realm's flex data and relation entries are complete and current preserves the realm's own records and still aligns behavior. This separates common design from local data, satisfying both governance and the council.

Question: 10

CHALLENGE 2 — Flex Master Data Completeness Behind Filtered Fields
A reviewer notes the parent commodities filtered correctly but two child commodities did not, and wants to scope the fix without destabilizing the parent.
What is the most precise corrective focus?

A. Reconfigure the relation types globally so that all realms are forced into identical filtering behavior everywhere
B. Rerun the reporting custom-field metadata pull so that the two affected commodities resolve their selectable set
C. Republish the form template across all realms so that every field on the sourcing form is refreshed at once
D. Complete the child realm's flex records and relation entries for the two affected commodities only, leaving the design

Answer: D

Explanation:
Scoping the fix to the child realm's flex records and relation entries for the two affected commodities targets the realm-specific data that actually diverged, leaving the validated shared design untouched. This corrects the symptom without destabilizing the parent.

Question: 11

CHALLENGE 3 — Custom Field Reporting and Translation Dependencies
A custom field is visible on the published sourcing form but does not appear when the analyst tries to build a report over recent events.
What most likely explains this and resolves it?

A. The custom-field metadata must be pulled for reporting so the field becomes available there, separate from the form

B. The field first needs database pre-allocation before it can be used on the form, and that also exposes it to reporting
C. The field must be translated into every supported regional language before the reporting tool will list it at all
D. The field must be migrated to production first before any reporting environment anywhere is able to see the field

Answer: A

Explanation:
Custom field metadata must be pulled for reporting so the field becomes available in reporting, and this is a step separate from configuring the field on the form. A field can be form-usable yet absent from reporting until that pull is performed.

Question: 12

CHALLENGE 3 — Custom Field Reporting and Translation Dependencies
A newly translated field label renders correctly in the parent realm's primary language but falls back to the technical field name in one regional language.
What is the most defensible diagnosis?

A. The custom-field metadata pull was skipped for the field, so its label is unable to resolve in that one language
B. The relation entries for that child realm are incomplete, and that is what is suppressing the label in that language
C. The translation for that language is missing or the added-language support was not completed for that label
D. The form template was never published in that child realm, so no label of any kind can render for that field there

Answer: C

Explanation:
A label that renders in one language but falls back to the technical name in another indicates the translation for that language is missing or the additional-language support was not completed for that label. The fallback is the expected behavior when a per-language translation is absent.

Question: 13

CHALLENGE 3 — Custom Field Reporting and Translation Dependencies
The program manager wants the custom field both reportable and correctly labeled in every supported regional language before cutover.
Which combination of steps achieves this?

A. Publish the form template and confirm the field renders, treating reporting and translation as automatic outcomes

B. Perform the custom-field metadata pull and maintain the translation files for every supported language, then verify
C. Add the field to a report manually in each realm and accept the technical name wherever a translation is unavailable
D. Migrate the field to production first and then add the reporting pull and the translations directly on the live realm

Answer: B

Explanation:
Making the field reportable requires the custom-field metadata pull, and correct labels in every language require maintaining the translation files per language, after which each is verified. Treating both as explicit steps is what satisfies the program manager's requirement.

Question: 14

CHALLENGE 3 — Custom Field Reporting and Translation Dependencies
A reviewer asks what evidence confirms the custom field meets its reporting and translation goals. Which validation set is correct?

A. The field deploys cleanly into the realm and the migration package log shows no errors against the field at all
B. The field has database pre-allocation and appears on the published sourcing form template for the recent events
C. The field's relation entries return the expected qualified suppliers for each commodity on the affected events
D. The field is selectable when building a report and every supported language renders its intended label, not the name

Answer: D

Explanation:
The reporting and translation goals are evidenced by the field being selectable when building a report and by every supported language rendering the intended label rather than the technical name. These checkpoints map directly to the two dependencies in this challenge.

Question: 15

CHALLENGE 3 — Custom Field Reporting and Translation Dependencies
A child site added one additional language late in the program, and several custom labels still fall back to the technical name there.
What is the soundest direction before cutover?

A. Complete the added-language support and the translation entries for the affected labels, then verify the rendering

B. Defer the additional language to a later phase and ship the cutover with the technical field names still showing
C. Hide the affected fields in that region so the technical-name fallback is no longer visible to the regional buyers
D. Switch the child site back to the parent realm's primary language so the new regional language need not be maintained

Answer: A

Explanation:
Completing the additional-language support and the per-label translation entries in the configuration, then verifying rendering before migrating, closes the gap through the governed path. This delivers correct labels in the new language at cutover.

Question: 16

CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms
A regional administrator proposes making one field correction directly in the production child realm to save a migration cycle, and the platform lead objects.
Which response best aligns with sound cutover governance?

A. Allow the direct production change this once since the correction is small and the cutover schedule is very tight
B. Make the change in production now and then back-fill it into the test site afterward to keep the two records aligned
C. Apply the correction in the test site, package it, confirm the production-availability check, then migrate it
D. Skip the field correction entirely so that no change to the production child realm is needed during this window

Answer: C

Explanation:
Sound governance applies the correction in the test site, packages it, confirms the production-availability check, and migrates it through the sanctioned path. This preserves baseline consistency and avoids unvalidated edits on the live realm.

Question: 17

CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms
Because the European parent realm is already live, the team classifies this wave as a brownfield migration.
What does that classification most accurately imply for the cutover?

A. The production site is built completely fresh, so the existing realm configuration and master data are discarded

B. No production-availability check is needed for this wave because the production realm already exists and is in service
C. Only selected configuration is carried forward while everything else is rebuilt from zero in a separate new site
D. Changes are applied on top of an already-live environment, so sequencing must protect configuration and live events

Answer: D

Explanation:
A brownfield migration works with an already-live production environment, applying changes on top of what is already configured, so cutover sequencing must protect existing configuration and in-flight events. This framing drives the careful, governed approach the scenario requires.

Question: 18

CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms
Sourcing events are mid-flight in production and the council requires they not be disrupted during the window.
Which sequencing decision best honors this constraint?

A. Confirm the production-availability check and schedule the deployment against the cutover plan to avoid disruption
B. Migrate immediately during peak sourcing activity since the configuration package itself deploys fairly quickly
C. Pause the entire production realm for the full duration of the migration regardless of which events are active
D. Migrate only to the child realms and leave the parent realm's shared definitions unsynchronized for this window

Answer: A

Explanation:
Honoring the constraint means confirming the production-availability check and scheduling the deployment against the cutover plan so in-flight events are not disrupted. This couples the readiness gate with calendar-aware sequencing.

Question: 19

CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms
The parent realm holds shared definitions that the child realms inherit.
What does this imply for the order of migration across realms?

A. The child realms should be migrated first so that the parent realm can then be reconciled to them afterward

B. Parent and child realms can be migrated in any order because the inheritance resolves itself automatically at runtime
C. Shared parent definitions go in first so children inherit a consistent baseline, then child-specific changes follow
D. Only the parent realm requires migration because the child realms always read everything directly from it while live

Answer: C

Explanation:
Because child realms inherit shared definitions from the parent, those parent definitions should be in place first so children inherit a consistent baseline, with child-specific changes following. The inheritance direction sets the migration order.

Question: 20

CHALLENGE 4 — Brownfield Cutover Sequencing Onto Live Realms
The team must confirm readiness immediately before the brownfield migration commences.
Which confirmation is the correct precondition for proceeding?

A. The reporting custom-field metadata pull has been run for every custom field carried in the migration package
B. The form templates affected by the wave have been republished in the test site ahead of the production migration
C. Every supported regional language has been translated for all standard fields across each realm in the estate
D. The production-availability check confirms that the production realm is available before the migration commences

Answer: D

Explanation:
The migration readiness check confirms that production is available before migration commences, which is the correct precondition for proceeding with the brownfield cutover. It is the gate that protects the live realm before any change is applied.
Topic: 2
Micro Skill Drill Exam

Question: 21

Aurelia Foods, a multinational packaged-foods group, is migrating an existing Strategic Sourcing configuration into an already-live production realm using a bluefield approach. You are the migration consultant. Your plan migrated flex master data, relations, enumerations, and translations, and you imported the field customizations from the source realm. During validation, the new custom fields and relations are present in production configuration, but they do not appear to buyers on sourcing documents, and reports do not yet recognize the new fields. The cutover runbook lists a step for

handling data definition changes after configuration is migrated, which has not yet been performed. The base configuration objects migrated without error, and there were no failed Intelligent Configuration Manager packages. Business users are waiting to transact in production, and the project cannot extend the cutover window. You must make the migrated field customizations and relations effective for end users without re-running the entire migration. The cutover team has only a limited number of agreed business-downtime hours, and any approach that means re-importing the full configuration would push the go-live past the window the customer approved. Which action correctly completes the migration so the changes take effect?
As the migration consultant, which action makes the migrated field customizations and relations effective for buyers and reporting in production?

A. Refresh, review, accept, and publish the data definition changes in the production realm, so the migrated field customizations and relations become active on sourcing documents and in reporting.
B. Switch the migration approach to greenfield and reload the configuration into a fresh realm, since bluefield migrations cannot make field customizations visible in an already-live production site.
C. Reload the flex master data and translations again with end-user visibility enabled, so the custom fields inherit display permissions and appear on sourcing documents for buyers.
D. Re-deploy the Intelligent Configuration Manager packages a second time, because field customizations only register for end users after the same package is imported twice into production.

Answer: A

Explanation:
In a bluefield migration the migrated data definition changes must be refreshed, reviewed, accepted, and published before they are effective. The configuration objects imported cleanly, but until this step is completed the new fields and relations are staged rather than active, which is why buyers and reports do not yet see them. Performing it activates the changes in the live realm without re-running the migration, meeting the cutover window.

Question: 22

Cobreval Mining, a Chilean copper producer, is extending its sourcing test site so that RFP responses capture three new supplier-entered fields on a form template. You are the consultant building the template. You created the form template, added the three new fields, and attempted to publish it, but publication is blocked and the new fields cannot be activated on the template. Existing fields on the same template behave normally. You confirmed the new field definitions are valid and that the template structure is otherwise correct. Reviewing the template build guidance, you note that fields used in form templates depend on a preparatory step at the database level before they can be bound into a template. The customer's go-live depends on suppliers being able to complete the new fields in their RFP responses, and the test-site baseline cannot be shared while publication fails. Suppliers will begin completing the form within days, and the three fields capture information the category team needs to score responses, so the template cannot simply go live without them. The other fields on the same template, by contrast, published without any difficulty. Which action allows the new fields to be added to the form template and published?
As the consultant, which action lets the three new fields bind into the form template and publish successfully?

A. Complete the database pre-allocation for the new form-template fields before binding them, so the template can reference the fields and publish successfully.
B. Recreate the three fields as enumerations on the general properties tab, because form templates can only publish fields that are defined as enumerations rather than free entry.
C. Move the three fields into the conditions configuration of the template so their visibility is evaluated at runtime, which clears the publication block on the template.
D. Translate the three field labels into every supported site language first, because untranslated form-template fields are blocked from publication until their labels resolve.

Answer: A

Explanation:
Fields used in form templates rely on database pre-allocation before they can be bound; without it, the template cannot reference the new fields and publication is blocked. Completing the pre-allocation provides the storage allocation the template expects, after which the three fields bind cleanly and the template publishes. This resolves the block at the layer where it originates rather than reworking the field definitions.

Question: 23

Helmsmark Logistics runs a Procurement configuration with a parent realm and several regional child realms for its operating units. You are the consultant asked to add a customer-specific field to a procurement object so it is available consistently across all regional child realms. Working in Customization Manager, you applied the field customization while connected to one regional child realm. The field now appears in that region but is absent in the other child realms, and regional teams report inconsistent forms. Re-applying the change individually in each child realm is being considered, but the customer wants a single, governed definition that all regions inherit, and expects to add two more regions next year. You confirmed the field definition itself is correct and that Customization Manager lists the available classes for the object. Each child realm currently shows slightly different forms to its regional buyers, which has already caused confusion in shared reporting, and the customer's governance team wants one definition it can point to as the source of truth rather than several regional variants that drift apart over time. Which approach correctly delivers the field consistently across the child realms while supporting future regions?
As the consultant, which approach makes the field available consistently across the existing and future child realms from one governed definition?

A. Re-apply the field customization separately in each existing child realm and document the steps, so every region carries its own copy of the field and stays independent of the parent.
B. Type-cast the procurement object to a new class in each child realm, because inheritance across realms requires a dedicated class per region before a field can be shared.
C. Apply the field customization at the parent realm in Customization Manager, so child realms inherit one governed definition and future regions pick it up.
D. Recreate the field as flex master data in the child realm where it works, so the other child realms can reference that realm's stored values directly at runtime.

Answer: C

Explanation:
Customizing Procurement realms with a parent-and-child structure means a field defined at the parent realm is inherited by the child realms. Applying the customization at the parent level gives every region one governed definition and ensures regions added later inherit it automatically. This resolves the inconsistency at the level that controls propagation, satisfying both the single-definition and future-growth requirements.

Question: 24

Velaterra Renewables, a mid-size wind-farm operator, is beginning a Strategic Sourcing deployment and must choose its deployment scenario during the prepare phase. You are the partner consultant. The customer has no existing sourcing configuration and no legacy sourcing data to carry forward; they want to adopt SAP best-practice configuration as closely as possible and reach go-live quickly with minimal custom rework. Their procurement team is small and has limited capacity to validate large volumes of migrated historical data. During the prepare-phase workshop, three directions are debated: starting from the standard solution baseline applied to a fresh test site; lifting an unrelated affiliate's heavily customized configuration; or planning a complex data carry-forward from spreadsheets the team has not validated. The steering committee wants a deployment scenario decision that fits a net-new adopter, supports the SAP Activate prepare-to-deploy path, and avoids unnecessary migration risk. Cost and timeline rise sharply if the scenario forces data validation effort the team cannot staff, and the steering committee is equally wary of taking on configuration the small procurement group will struggle to maintain after go-live. Which deployment scenario should you recommend?
As the partner consultant, which deployment scenario best fits this net-new adopter under the SAP Activate path?

A. Plan a data carry-forward that loads the unvalidated spreadsheet history into the new site, so historical sourcing data is preserved for reporting from go-live onward.
B. Defer the deployment-scenario decision until the realize phase, so the team can configure first and choose the migration path once the build is more complete.
C. Lift the affiliate's existing customized configuration into the new site, so the customer starts from a proven build and shortens the prepare phase by reusing prior work.
D. Adopt the standard solution baseline on a fresh test site as a greenfield deployment, since the customer is a net-new adopter with no legacy data to carry forward.

Answer: D

Explanation:
A net-new adopter with no legacy configuration or data to carry forward fits a greenfield deployment built on the standard solution baseline in a fresh test site. This matches the customer's wish to adopt best practice quickly, avoids unnecessary data-validation effort the small team cannot absorb, and aligns with determining the deployment scenario during the prepare phase of the SAP Activate path.

Question: 25

Bressettoni Industriale, an Italian industrial-pump manufacturer, is standing up a sourcing test site before its first guided event. You are the partner consultant applying the configuration baseline. The category team requested a field, Incoterm Override Justification, that must appear on the sourcing event header only when the existing Requires Incoterm Override field is set to Yes; when that control is No, the justification field must stay hidden. You created the justification field, set its label, data type, and help text, and saved it. In testing, the field renders correctly but is always visible regardless of the controlling field's value. The category lead confirms the requirement is a conditional display tied to another field on the same object, not a permanent addition. You verified both fields exist on the same class and that the controlling field already carries Yes and No values. Nothing you set on the field's general properties changed the always-visible behavior. The go-live checklist requires the field to honor the conditional rule before the baseline is shared to the customer's site. Which configuration action correctly enforces the conditional display?
As the consultant preparing the baseline, which configuration action makes the justification field appear only when the controlling field is set to Yes?

A. Delete the justification field and recreate it as flex master data, because value-dependent visibility is only supported for flex master data custom fields and not for configured fields.
B. Re-open the field and change its field type on the general properties tab, since a field's display behavior is set by the field type chosen when the field is first defined.
C. Add an enumeration to the controlling field's general properties so the justification field inherits those values and is suppressed automatically whenever the control is left unset.
D. Define the display dependency in the field's conditions configuration, so the field's visibility is driven by the controlling field's value rather than by its general properties.

Answer: D

Explanation:
Value-dependent visibility is enforced through the field's conditions configuration, where a rule ties the justification field to the controlling field's Yes value. Defining the dependency there makes the field appear only when the control is Yes and stay hidden otherwise, which is exactly the conditional display the category team specified. This acts at the layer that evaluates one field's state against another, resolving the always-visible behavior at its source.

Question: 26

Norsk Foredling, a Norwegian seafood processor, is configuring its sourcing test site so that when a category manager selects a Commodity Group on a sourcing project, the Sub-Commodity field offers only the sub-values that belong to that group. You are the consultant. You created both fields as flex master data, loaded the commodity and sub-commodity value sets, and confirmed each value list displays correctly on its own. In testing, the Sub-Commodity field shows the entire sub-commodity catalogue regardless of which Commodity Group is chosen, so category managers can pick mismatched pairs. The data loads are complete and the values are spelled correctly; the problem is that the child field is not narrowing to the parent selection. You verified that both flex master data sets exist on the same realm and that the field configuration references the correct classes. The baseline cannot be shared until the dependent filtering behaves as specified, because mismatched commodity pairings corrupt downstream spend reporting. The customer's spend taxonomy is large, with many sub-commodities under each commodity group, so an unfiltered list also slows category managers down and

makes incorrect pairings more likely the longer it stays unresolved. Which configuration action establishes the dependent filtering between the two fields?
As the consultant, which action makes the Sub-Commodity field display only the values related to the selected Commodity Group?

A. Convert both fields from flex master data to enumerations on the general properties tab, because enumerations enforce parent-child filtering automatically while flex master data lists always display in full.
B. Add the sub-commodity field to a new form template and publish it, so the template binds the child values to the selected commodity group at runtime through database pre-allocation.
C. Reload the sub-commodity flex master data with the commodity group name prefixed to each value, so the displayed list visually groups sub-values under their parent and discourages mismatched selections.
D. Define a relation type linking the commodity group and sub-commodity objects and apply the relation entries in field configuration, so the child field filters to the values related to the selected parent.

Answer: D

Explanation:
Dependent filtering between two objects is established by defining a relation type between them and applying the relation entries in field configuration. This binds each sub-commodity to its parent commodity group, so the child field narrows to only the related values when a group is selected. It acts at the object-relationship layer that governs native filtering, which is the source of the unfiltered behavior.

Visit us at: https://www.examsboost.com/test/c-acdet

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer