DUMPSBOSS.

Microsoft Dynamics 365 Business Central Developer

Microsoft MB-820

Version Demo

Total Demo Questions: 10

Total Premium Questions: 108

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Develop by using AL language	45
Topic 2, Develop by using the AL development environment	15
Topic 3, Develop by using the application lifecycle management process	14
Topic 4, Develop the user interface	16
Topic 5, Develop business logic	4
Topic 6, Integrate Business Central with external applications	14
Total	108

QUESTION NO: 1 - (HOTSPOT)

You plan to create a table to hold client data.

You have the following data integrity requirements:

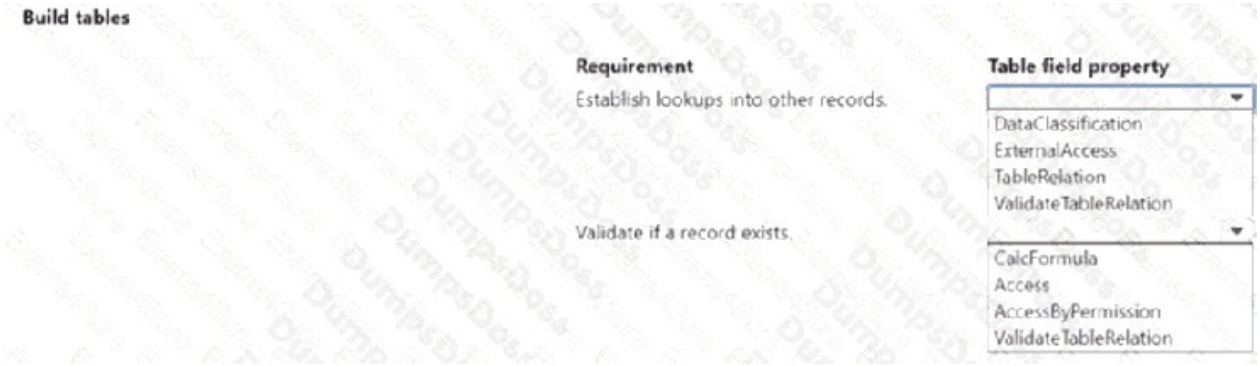
• Lookups into other records must be established.

• Validate if a record exists in a destination record.

You need to select the table field property to use for each requirement.

Which table field property should you use? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.



ANSWER:

Explanation:

Use **TableRelation** to establish lookups into other records, and use **ValidateTableRelation** to validate that an entered value exists in the related destination record. In Business Central AL, TableRelation is the field property that declares the relationship between the current field and a field in another table. For example, a client-related field can point to a key field in a client table. Once this relationship is defined, Business Central can use it to present related records for lookup and to understand which table supplies the permitted values. The relationship can also include conditional relations and filtering expressions where the business scenario requires them.

ValidateTableRelation works together with that relation. When it is enabled, Business Central verifies that a value entered into the field corresponds to a record allowed by the declared table relation. This enforces referential integrity at data entry time: the field cannot retain a value that does not identify a valid related record. That behavior is particularly important for client data because it prevents records from referring to nonexistent destination records and keeps relationships consistent as users enter, edit, import, or otherwise process data. The applicable Microsoft documentation describes TableRelation as the property for specifying relations to another table, and explains the validation behavior of ValidateTableRelation. Together, these properties provide the lookup relationship and the existence check required here.

QUESTION NO: 2

You need to determine why the extension does not appear in the tenant.

What are two possible reasons for the disappearance? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. The extension was published as a DEV extension.

B. The extension was not compatible with the new version within 60 days of the first notification.

C. The extension was published as PTE. and the Platform parameter was not updated in the application file.

D. The extension was published as PTE. and the Platform and Runtime parameters were not updated in the application file.

E. The extension was not compatible with the new version within 90 days of the first notification.

ANSWER: B E

Explanation:

The extension was not compatible with the new version within 60 days of the first notification and the extension was not compatible with the new version within 90 days of the first notification are both valid reasons that an extension can no longer be present or available after a major update. Business Central online enforces version compatibility for extensions as the service moves tenants to a new major version. The applicable grace period depends on the extension type and the update process. Publishers of marketplace extensions are given a limited period to provide a compatible version, while customers and partners responsible for per-tenant extensions have a defined remediation window to deliver and deploy a compatible update. If compatibility is not established during the applicable window, the extension can be removed so that the tenant can proceed on the supported major version. Maintaining a compatible extension package requires testing against the target release and publishing an updated package before the deadline. Microsoft documents the major-update compatibility process and extension update requirements in its major updates documentation and provides guidance for managing app.json extension metadata.

QUESTION NO: 3

You need to allow debugging in an extension to view the source code. In which file should you specify the value of the allowDebugging property?

A. settings.json

B. rad.json

C. app.json

D. launchjson

ANSWER: C

Explanation:

app.json is correct because it is the extension manifest file that defines metadata and packaging settings for an AL project. The debugging and source-exposure settings are declared in this manifest through the `resourceExposurePolicy` object. Setting `allowDebugging` to `true` authorizes debugging of the published extension, subject to the environment, user permissions, and debugging configuration. For example, an extension publisher can include `"resourceExposurePolicy": { "allowDebugging": true }` in app.json when building the app package. This setting is relevant when another developer must debug into the extension rather than treating it solely as external application behavior.

The manifest is included when the extension is packaged and published, allowing Business Central to apply the extension publisher's resource-exposure policy. In the same policy area, publishers can control related behaviors such as whether source can be downloaded and whether source is included in symbol files. Consequently, the appropriate place to specify `allowDebugging` is app.json, not a local editor or launch configuration file. Microsoft documents the manifest structure and the resource exposure policy in its AL developer documentation: JSON files in AL projects and Resource exposure policy.

QUESTION NO: 4

You are developing an app.

You plan to publish the app to Microsoft AppSource.

You need to assign an object range for the app.

Which object range should you use?

A. custom object within the range 50000 to 59999

B. custom object within the range 50000 to 99999

C. divided by countries and use specific a country within the range 100000 to 999999

D. an object range within the range of 70000000 to 74999999 that is requested from Microsoft

E. free object within the standard range 1 to 49999

ANSWER: D

Explanation:

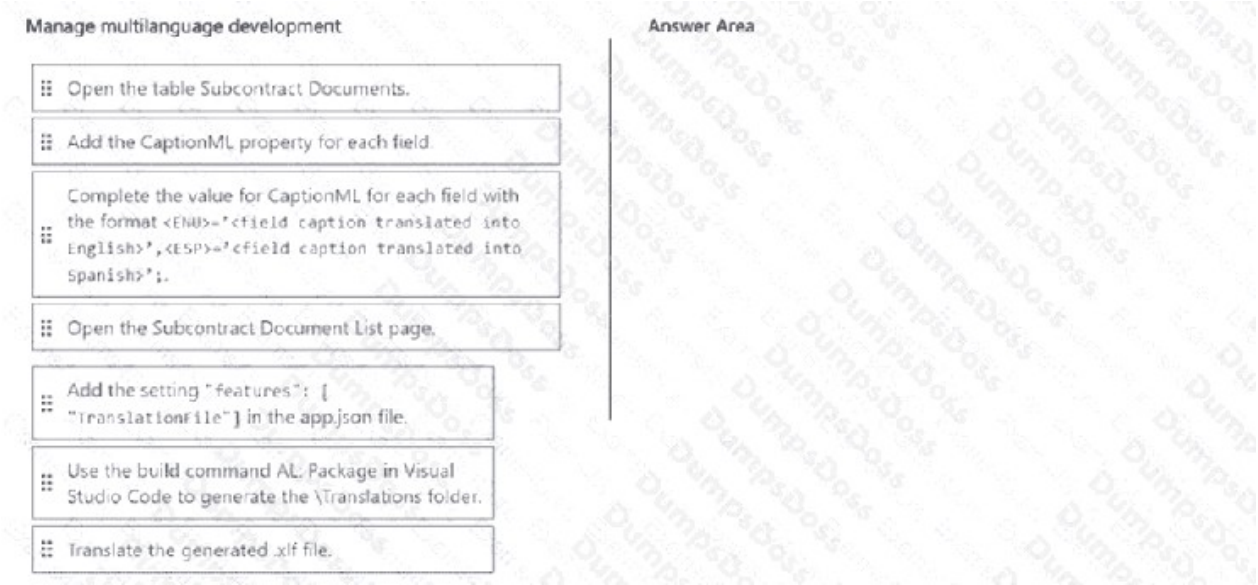
The correct choice is **an object range within the range of 70000000 to 74999999 that is requested from Microsoft**. Microsoft assigns object ID ranges to publishers that intend to distribute Business Central extensions through AppSource. This assignment ensures that every published app has a unique, reserved block of object IDs and prevents collisions with the Business Central base application and with objects supplied by other AppSource extensions.

Object IDs are part of an extension's public technical identity: tables, pages, codeunits, reports, enums, and other AL objects must use IDs that belong to the app's allocated range. Before submitting an app, the publisher requests an AppSource object range from Microsoft and uses that assigned range consistently in the app's source code and app configuration. The stated range, 70000000 through 74999999, is the range Microsoft designates for AppSource extension object IDs. This process supports reliable installation, upgrade, dependency management, and coexistence of multiple extensions in customer environments. Microsoft's guidance on object ranges and the AppSource submission process is available in Object ranges in Business Central and AppSource submission checklist.

QUESTION NO: 5 - (DRAG DROP)

You need to configure the Subcontract Docs extension to translate the fields.

Which three actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



ANSWER:

Explanation:

Business Central extensions use generated translation files for the current multilingual development workflow. First, the extension's `app.json` manifest must enable the translation-file capability by including `"features": [ "TranslationFile" ]`. This setting tells the AL compiler that the app should produce translation resources as part of the package build process. It is important that this configuration is present before creating the package, because the compiler reads the manifest while building the extension.

Next, run **AL: Package** from Visual Studio Code. Packaging compiles the extension and generates the translation output under the project's `Translations` folder. The generated XLIFF file contains translatable source strings collected from the extension metadata, such as captions, labels, and other supported user-facing text. This process keeps translations separate from the application object definitions and makes them manageable across languages and releases.

Finally, translate the generated `.xlf` file. Add the target-language translations to the XLIFF content while preserving its identifiers and structure. Business Central uses the translated XLIFF resources when the extension is deployed, allowing the appropriate localized captions and text to be shown for users' language settings. When source captions later change, packaging can regenerate or update the translation resources so the translation work can be maintained consistently.

This sequence follows Microsoft's extension localization model: enable translation-file generation, package the project to produce the source XLIFF resource, and translate that resource. Microsoft documents the manifest feature and the generated translation-file process in Working with translation files and describes extension manifest configuration in JSON files.

QUESTION NO: 6

You are creating fields in an extension table.

The table stores an employee email address and free-text notes entered by a customer during a support request.

You need to apply appropriate data classifications to the fields.

Which two classifications should you use? Each correct answer presents part of the solution. Choose two.

NOTE: Each correct selection is worth one point.

A. EndUserIdentifiableInformation for the employee email address

B. CustomerContent for the customer support notes

C. SystemMetadata for the employee email address

D. OrganizationIdentifiableInformation for the employee email address

E. AccountData for the customer support notes

ANSWER: A B

Explanation:

EndUserIdentifiableInformation is appropriate for an employee email address because the value can identify a specific person. **CustomerContent** is appropriate for free-text support notes supplied by a customer because the content is entered and owned by the customer and can contain business information relevant to the support request.

`SystemMetadata` is intended for system-generated technical metadata. `OrganizationIdentifiableInformation` applies to information that identifies an organization rather than an individual person.

QUESTION NO: 7 - (DRAG DROP)

You are treating an app tor Business Central.

You plan to specify the following parameters and properties of the server and app.

• Startup object type and object ID

• Runtime

• Dependencies

You need to configure the JSON file for the specified parameters and properties

Which JSON files should you configure? To answer, move the appropriate files to the correct object purposes You may use each file once, more than once, or not at all You may need to move the split bar between panes or scroll to view content

NOTE Each correct selection is worth one point.

ANSWER:

Explanation:

A Business Central AL project uses separate JSON configuration files for debugging settings and extension metadata. The startup object settings belong in `launch.json`. This file defines how the extension is launched from the development environment, and its configuration can include `startupObjectType` and `startupObjectId`. Those settings tell Business Central which page, report, codeunit, or other supported object should open when the project starts a debugging session.

The `app.json` file is the extension manifest. It describes the extension itself and specifies the platform/runtime compatibility requirements. Its `runtime` property declares the Business Central runtime version for which the app is compiled. The manifest also contains the `dependencies` collection. Each dependency identifies another extension that must be available, typically through its publisher, name, version, and application ID, so symbols can be resolved and the dependency relationship is recorded for packaging and installation.

Accordingly, configure `launch.json` for the startup object type and object ID, then configure `app.json` for both the runtime and dependencies properties. Microsoft documents the launch configuration settings at Launch.json file and the extension manifest schema at JSON files.

QUESTION NO: 8

You are developing a page action that evaluates the **Balance (LCY)** FlowField on a Customer record.

The action must reject processing when the balance is greater than the approved credit limit.

You need to ensure that the FlowField contains its calculated value before the comparison is made.

What should you do?

A. Call `Customer.CalcFields("Balance (LCY)");` before the comparison.

B. Call `Customer.Modify();` before the comparison.

C. Call `Customer.SetRange("Balance (LCY)");` before the comparison.

D. Call `Customer.FindFirst();` before the comparison.

ANSWER: A

Explanation:

Call `CalcFields` for `Balance (LCY)` before evaluating the field. A FlowField is a virtual field whose value is calculated from a `CalcFormula`; it is not necessarily calculated when a record is retrieved. Calling `Customer.CalcFields("Balance (LCY)");` populates the value for the current record before the comparison is performed.

`Modify` persists changes and does not calculate FlowFields. `SetRange` applies a filter, and `FindFirst` retrieves a record, but neither operation ensures that the FlowField value has been calculated.

QUESTION NO: 9

A company uses Business Central.

You plan to help users through the installation process by using Assisted Setup.

You need to create a wizard page.

Which two actions should you perform? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

A. Set the PageType property to NavigatePage.

B. For each step needed in the guide, add a group control to the root-level of the layout > area(Content) control.

C. Set the PageType property to Worksheet

D. For each step needed in the guide, add a repeater control to the root-level of the layout > area(Content) control.

ANSWER: A B

Explanation:

Set the `PageType` property to `NavigatePage` when creating a wizard for Assisted Setup. `NavigatePage` is the Business Central page type intended for guided, step-by-step processes. It provides the navigation behavior and actions commonly used in a wizard, such as moving backward and forward through the setup flow and completing or closing the guide. This makes it the appropriate foundation for an Assisted Setup experience.

For each step needed in the guide, add a `group` control at the root level of `layout > area(Content)`. A NavigatePage wizard normally contains a separate group for each logical step. AL code can control the visibility of these groups according to the current step, so users see only the content relevant to the current stage of the setup. The groups also provide a clear structure for fields, instructional text, and other controls displayed on each wizard page step.

Microsoft describes creating guided experiences with NavigatePage pages and organizing wizard content into groups in its Designing Navigate Pages documentation. For the broader setup framework in which these guided pages are used, see Assisted Setup.

QUESTION NO: 10 - (HOTSPOT)

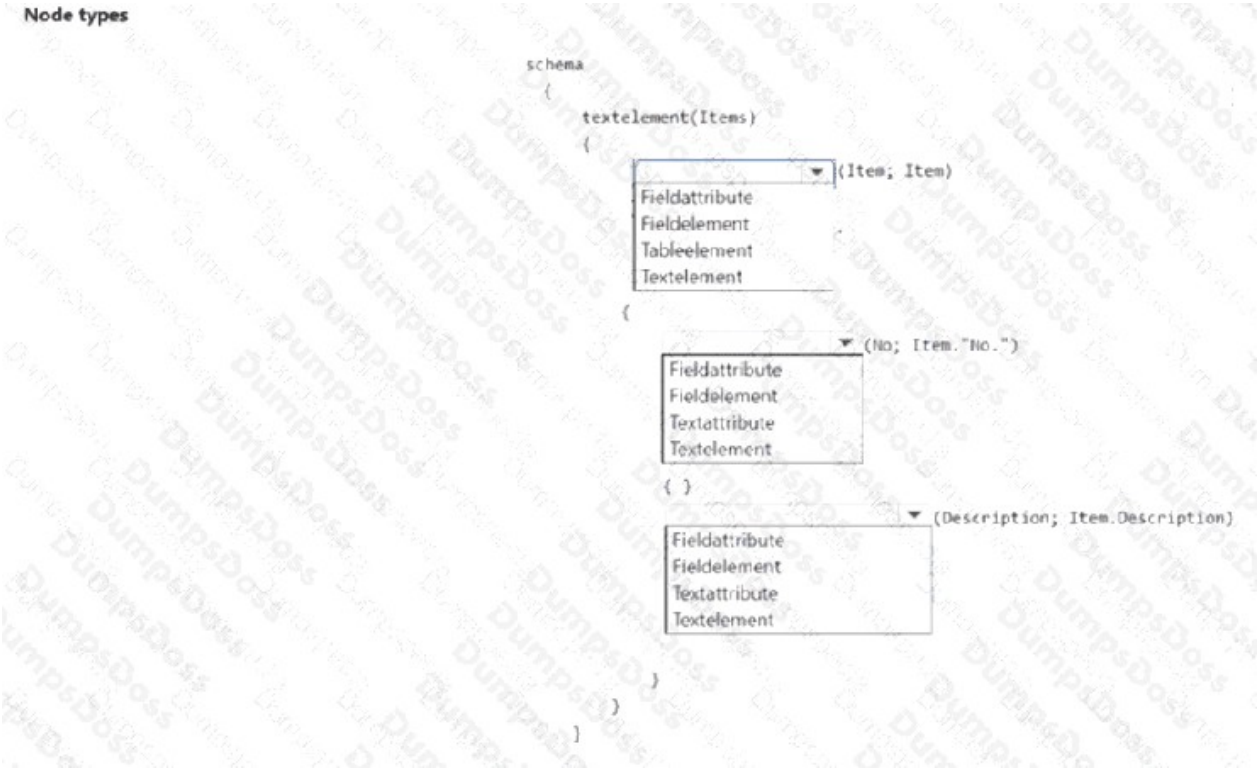
You have the following XML file sample for the Items list:

```
<Items>
  <Item No="1000">
    <Description>Table</Description>
  </Item>
  <Item No="1001">
    <Description>Chair</Description>
  </Item>
  <Item No="1002">
    <Description>Sofa</Description>
  </Item>
</Items>
```

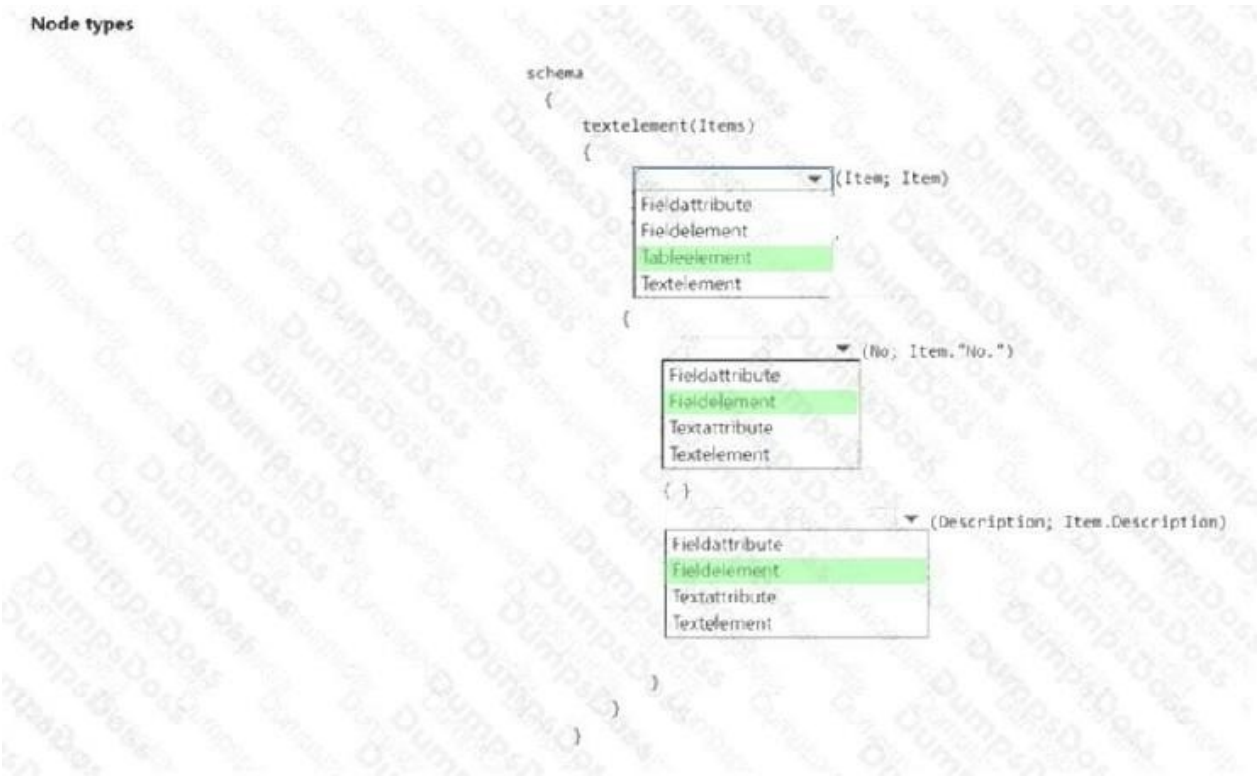
You plan to create the next XML file by using an XMLport object.

You need to complete the code segment to export the file in the required format

How should you complete the code segment? To answer, select the appropriate options in the answer area.



ANSWER:



Explanation:

An XMLport schema must mirror the required XML hierarchy and distinguish XML attributes from child elements. The outer `textelement(Items)` creates the document's root `<Items>` element. The repeating `<Item>` nodes must be based on Business Central Item records, so `tableelement(Item; Item)` is required. A table element iterates the underlying record source and emits one XML item for each exported Item record.

Within each generated Item node, the item number appears in the opening tag as `No="1000"`. This is an XML attribute, not a separate nested element. Therefore, the appropriate XMLport node is `fieldattribute(No; Item."No.")`. A field

attribute writes the value directly into the start tag of its containing element, producing the required `<Item No="1000">` format.

The description is different because the target XML shows it as its own nested element: `<Description>Table</Description>`. The appropriate node is consequently `fieldelement(Description; Item.Description)`, which exports the field value as the content of a child XML element. This combination preserves both the repeating record structure and the exact attribute-versus-element layout required by the sample. Microsoft's XMLport schema documentation describes `tableelement`, `fieldattribute`, and `fieldelement` node behavior in the XMLport object documentation and its XMLport schema documentation.