PassQueen
a.k.a dumpsqueen.com

Alfresco Content Services Certified Engineer

Alfresco ACSCE-5X

Version Demo

Total Demo Questions: 9

Total Premium Questions: 99

Buy Premium PDF

https://passqueen.com

support@passqueen.com

QUESTION NO: 1

Share UI configuration will allow you to control which aspects a user sees. What other two things (out of the box) can be controlled? (Choose two.)

A. The combinations of aspects that can be applied together.

B. The aspects that can be applied based on the underlying type.

C. The aspects that can be removed once applied.

D. The aspects a user can add.

E. The aspects that are mandatory.

ANSWER: C D

Explanation:

In Alfresco Share's Document Library configuration, the `<aspects>` section provides separate lists that govern aspect availability and lifecycle actions in the user interface. In addition to the `<visible>` list, which controls the aspects presented to users, the `<addable>` list controls the aspects a user can add. Administrators can populate this list with aspect QNames so that eligible aspects are offered through Share's Manage Aspects action for the applicable document-library context.

The `<removeable>` list controls the aspects that can be removed once applied. This lets an administrator expose removal as an allowed Share UI action for selected aspects while retaining control over the set of aspects presented to users. These settings are part of Share configuration and can be scoped through the relevant Document Library configuration, such as by site, type, or evaluator context. They affect the UI choices exposed in Share; Alfresco repository rules, model definitions, and permissions continue to apply when the requested aspect operation is performed.

Alfresco's Share configuration documentation describes the Document Library `aspects` configuration, including its visible, addable, and removeable aspect lists. See Alfresco Share extension points documentation and Alfresco documentation on configuring aspects in Share.

QUESTION NO: 2

Select two ways to audit Alfresco Content Services events in an Alfresco Content Services extension? (Choose two.)

A. Mark a method in the extension with an @Auditable annotation.

B. Open a connection to the database and write the audit information to the audit tables.

C. Inject AuditComponent into the extension and call recordAuditValues.

D. Use the Alfresco REST API to record an audit event.

E. Call the Java Audit Service.

ANSWER: A C

Explanation:

Marking an extension method with an `@Auditable` annotation is a supported way to make method invocations available to Alfresco's audit infrastructure. The auditing interceptor captures the configured method data, which can then be processed through an enabled audit application and its audit configuration. This is useful where auditing should be associated consistently with a service operation rather than manually emitted throughout the implementation.

Injecting `AuditComponent` and calling `recordAuditValues` is also the direct programmatic mechanism for an extension to submit a map of audit values under an audit application path. Alfresco routes those values through the audit subsystem,

where configured data generators, path mappings, filters, and audit applications determine whether and how an audit entry is stored. The extension should provide serializable values and use an application/path structure that corresponds to the deployed audit configuration. These approaches preserve the audit subsystem's normal configuration, enablement, and persistence behavior rather than bypassing it.

See Alfresco's audit subsystem and extension guidance in the Alfresco Content Services audit documentation and the repository audit extension documentation.

QUESTION NO: 3

What is the purpose of widgetUtils.findObject function in the Aikau framework?

A. Used to find an existing widget while creating new Aikau pages.

B. Used to find a list of all Aikau widget objects used in the application.

C. Used to find an existing widget while extending Out-of-the-box Aikau pages.

D. Used to find a list of all Aikau pages used in the application.

ANSWER: C

Explanation:

Used to find an existing widget while extending Out-of-the-box Aikau pages. The `widgetUtils.findObject` utility is intended for locating a specific configuration object in an existing Aikau page model, typically by matching a property such as the widget's identifier. Once the target widget configuration has been located, an extension can safely amend its configuration, add child widgets, or otherwise tailor the supplied page without replacing the entire page definition.

This is particularly useful in Share/Aikau customization because standard pages are represented by JSON models containing nested widget definitions. An extension module can access the relevant model structure, use `findObject` to identify the preconfigured widget it needs to work with, and then apply a focused change. This approach preserves the surrounding out-of-the-box page behavior and reduces duplication of standard configuration. The Aikau source includes the widget utility module used by page customizations; see the widgetUtils source. Alfresco's Aikau extension guidance also describes extending existing pages through page-model configuration: Aikau extension points documentation.

QUESTION NO: 4

Which of the following items appear within a custom content model? (Choose two.)

A. The model's namespace.

B. References to other namespaces.

C. Property sheet definitions.

D. Policy behaviors for a content type.

E. Localization strings.

ANSWER: A B

Explanation:

A custom Alfresco content model is defined in an XML model file. The file declares the model's own namespace, which gives the model a unique URI and prefix used to qualify its types, aspects, properties, associations, and constraints. This namespace declaration is a core part of the model definition and prevents naming conflicts with repository and other custom models.

The model can also contain references to other namespaces through its imports section. Imports make definitions from existing models available for use in the custom model, such as extending a type from the standard content model or using a

type defined by another model. An import identifies the referenced model namespace and its prefix, allowing those qualified names to be resolved when Alfresco loads and validates the model.

These declarations belong to the XML content-model structure itself and are normally specified near the beginning of the model, before the custom type and aspect definitions that use them. Alfresco's content-model extension documentation describes both namespace declarations and imported model namespaces as elements of a repository content model. See Alfresco Content Services: Content models and Alfresco Content Services: Model structure.

QUESTION NO: 5

Which one of the following steps is necessary when customizing the "Advanced Search" in Share and adding a date property to the search form?

A. The date range control needs to be specified in share-config-custom.xml.

B. A Java-backed web script to deal with dates needs to be created and referenced in share-config-custom.xml.

C. Share will automatically provide a date range control for dates.

D. A date range control needs to be created in FreeMarker, then specified in share-config-custom.xml.

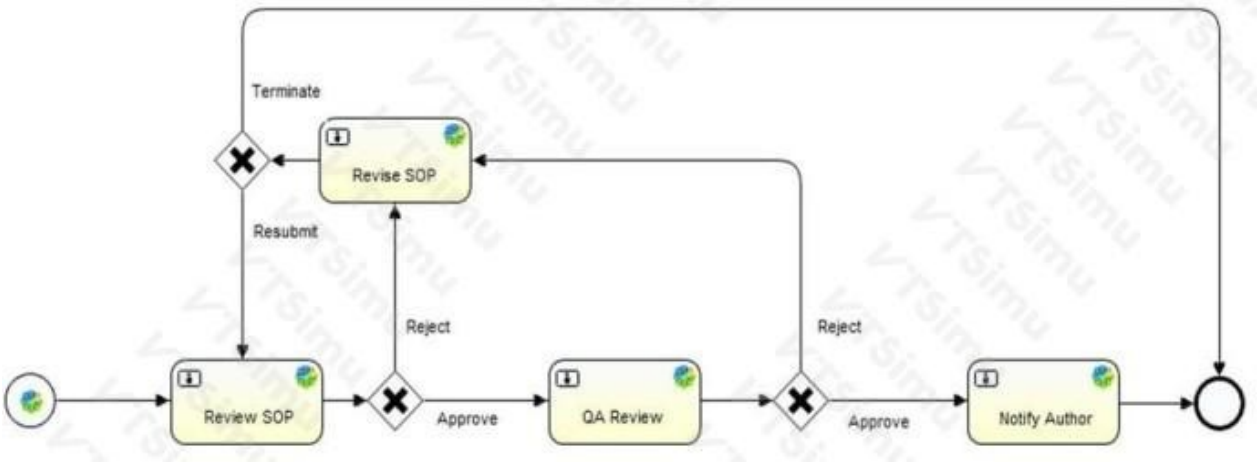
ANSWER: A

Explanation:

The date range control needs to be specified in share-config-custom.xml. Share Advanced Search is driven by the Share configuration model, and a date metadata property requires a search control that can collect a range rather than merely one unqualified text value. In the Advanced Search form appearance configuration, the property is associated with Share's date-range control template, typically `/org/alfresco/components/search/advsearch-daterange.ftl`. This control presents the appropriate from/to date inputs and submits the range in the format expected by the repository search implementation. The configuration belongs in `share-config-custom.xml`, normally in an extension module or the relevant Advanced Search configuration section, so that the customization survives Share upgrades and overlays the standard configuration correctly. The property must also be available in the applicable content model and selected for the Advanced Search form, but selecting the date-range control is the necessary step specific to handling a date property as a date range. Alfresco's Share configuration extension mechanism is documented in the Share extension points documentation; the Share source also includes the relevant search control templates in its template resources.

QUESTION NO: 6

The workflow pictured, shows 3 gateways. What type of gateways are they?



A. Exclusive

B. Parallel

C. Inclusive

D. Fork

E. Join

ANSWER: A

Explanation:

Exclusive is correct. In BPMN workflow diagrams, an exclusive gateway is represented by a diamond containing an “X” marker, which is the visual notation shown for the gateways in the workflow. Alfresco Process Services uses BPMN 2.0 modeling concepts, including exclusive gateways, to control the path followed by a process instance. When execution reaches an exclusive gateway, the configured sequence-flow conditions are evaluated and one eligible outgoing path is selected. This supports decision-based routing, such as sending a task for one outcome when a condition is met and a different outcome when it is not. An exclusive gateway can also be used where multiple alternative paths later converge, while preserving the behavior that only one branch is active for a given process instance. The marker and the decision-routing behavior identify the pictured gateways as exclusive gateways. Alfresco’s Process Services documentation describes configuring an exclusive gateway and its conditional outgoing flows at Alfresco Process Services: Exclusive gateway. The underlying BPMN exclusive-gateway semantics are also described in the OMG BPMN 2.0 specification.

QUESTION NO: 7

Which one of the following steps is necessary when customizing the “Advanced Search” in Share and adding a date property to the search form?

A. The date range control needs to be specified in share-config-custom.xml.

B. A Java-backed web script to deal with dates needs to be created and referenced in shareconfig-custom.xml.

C. Share will automatically provide a date range control for dates.

D. A date range control needs to be created in FreeMarker, then specified in share-configcustom.xml.

ANSWER: A

Explanation:

The date range control needs to be specified in share-config-custom.xml. Share’s Advanced Search page is driven by Share configuration, including the definition of search properties and the controls used to render them. A date property requires an appropriate date-range control declaration so that Share renders fields suitable for entering a start and end date and submits search constraints in the expected form. This configuration is normally placed in the project’s `share-config-custom.xml` extension file, allowing the standard Share Advanced Search component to use the control without replacing the page or implementing custom server-side date handling.

The configuration-based approach also keeps the customization aligned with Share’s extension model: search form behavior, property visibility, labels, and renderer/control selection are declared in configuration rather than embedded in a bespoke web script. The configured date control works with the repository’s search model to produce date-oriented search criteria, while preserving the normal Advanced Search user experience. For Share configuration and extension concepts, see the Alfresco Content Services developer documentation and the Alfresco Community repository.

QUESTION NO: 8

Which searches are possible to execute with CMIS 1.1 cmis:item support on Alfresco? (Choose two.)

A. select * from cmis:item

B. select * from sys:base

C. select * from rule:rule

D. select * from bpm:package

E. select * from cm:person

ANSWER: A E

Explanation:

CMIS 1.1 adds the `cmis:item` base type for repository objects that are neither documents nor folders. Alfresco implements this support by exposing eligible non-document and non-folder repository types through the CMIS item model, allowing them to be addressed in CMISQL queries. Therefore, `select * from cmis:item` is a valid broad query: it searches the repository's CMIS item objects and can return instances of types represented as items.

`select * from cm:person` is also valid because Alfresco exposes its person objects as CMIS items. A CMISQL query may target a concrete Alfresco type that is available through the item mapping, so this query can retrieve person objects and their queryable properties. This is useful for integrations that need to locate Alfresco user/person records through the same CMIS query interface used for other repository objects.

Alfresco's CMIS documentation describes how CMIS 1.1 item support exposes additional repository object types and enables querying them through CMISQL. See the Alfresco CMIS 1.1 item support documentation and the CMIS 1.1 specification.

QUESTION NO: 9

Which Alfresco Java Public API service can be used to inspect the currently loaded content models?

A. NodeService

B. DictionaryService

C. ContentService

D. ModuleService

ANSWER: B

Explanation:

DictionaryService is the Alfresco Java Public API service intended for inspecting the repository dictionary: the content models, types, aspects, properties, associations, constraints, and data types currently known to the repository. Alfresco loads its model definitions into this dictionary during startup and when model definitions are deployed, making the service the appropriate programmatic entry point for examining available model metadata.

For example, application code can use DictionaryService to retrieve a type or aspect definition from its qualified name, determine whether a class is a type or aspect, inspect inherited definitions, and enumerate registered model definitions. This is especially useful in repository extensions that must adapt their behavior according to the models installed in a particular environment, rather than relying on hard-coded assumptions about properties or aspects.

The public interface exposes methods such as `getClass(QName)`, `getProperty(QName)`, `getAssociation(QName)`, and `getAllModels()`, which directly support this form of metadata inspection. Alfresco's source definition for the service is available in the DictionaryService public API. See also the Alfresco Content Services developer documentation for repository extension and API guidance.