Atmic Networks

CERTIFICATION EXAM SUPPORT

EXAM PREPARATION MATERIAL

Practice files designed to help you understand exam patterns, question styles and the overall assessment format.

IMPORTANT NOTE

Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns.

Stay connected with our support team
for the latest file updates and exam-related information.

www.atmicnetworks.com

Knowledge that moves beyond memorization.

Question: 1

A customer wants to display a small toolbar with three icons - a "sad face," a "face with neutral expression," and a "happy face" - on the bottom of every page in their application.

Users will be instructed to use the icon that best expresses their current experience using the application. This will allow the customer to collect valuable data about users.

Which object type should be called from each page to implement this feature?

A. An interface, because the component must render user interface elements.

B. An expression rule, because the component captures expressions of user sentiment for analysis.

C. A decision, because the component captures a choice that users select from an array of custom selection components, rather than a standard dropdown or radio button.

Answer: A

Explanation:

In Appian, interfaces are used to design and render user interface elements, allowing for the creation of custom layouts and components that can interact with users. In this scenario, the toolbar with emotion icons is a UI element that needs to be consistently presented across various pages. Using an interface ensures that these icons can be interactively used by users to express their sentiments, and the data collected can be analyzed for user experience insights. Interfaces are capable of embedding such UI elements and can be invoked or included across multiple pages within an application, making them the ideal choice for this requirement.

Reference:

Appian Documentation on Interfaces: Provides detailed guidance on how to design and use interfaces to create user-centric UI components.

Question: 2

You select the "Generate groups and folders to secure and organize objects" option while creating a new application, Acme, with the prefix ACM.

By default, which two groups are generated by Appian? (Choose two.)

A. ACM Administrators

B. ACM Designers

C. ACM Viewers

D. ACM Users

Answer: A, B

Explanation:

When creating a new application in Appian and opting to generate groups and folders for organization and security, Appian automatically creates specific groups to facilitate application development and management. The default groups include "Administrators" and "Designers" with the application prefix, in this case, ACM. The ACM Administrators group is intended for users who will have full control over the application, including configuration and deployment aspects. The ACM Designers group is designated for users primarily involved in the design and development of the application, granting them necessary permissions to create and modify application components without full administrative privileges.

Reference:

Appian Documentation on Application Design: Offers insights into best practices for structuring and securing Appian applications, including the use of groups for effective application management.

Question: 3

You need to remove an unused field from an existing record type Product, which has data sync enabled and is backed by a database table.

What should you do?

A. Delete the field from the record type and optionally delete the column from the database table.

B. Delete the field from the product Custom Data Type (CDT) and perform a full resync of the record type.

C. Delete the column from the database table and perform a full resync of the record type.

Answer: B

Explanation:

In Appian, when dealing with a record type that has data sync enabled and is backed by a database table, changes to the structure of the underlying data model, such as removing a field, should be carefully managed. The correct approach involves deleting the unused field from the Custom Data Type (CDT) that defines the structure of the data for the record type. Following this change, a full resynchronization of the record type is necessary to ensure that the changes in the CDT are reflected in the record type and its associated data in Appian. This process ensures data integrity and consistency across the application and the database.

Reference:

Appian Documentation on Data Management: Provides guidelines on managing data structures, including CDTs and record types, within Appian applications.

Question: 4

You have a record action that should only be visible to certain users under conditions specified by an expression.

How should you configure this?

A. Set security on the process model.

B. Set permissions directly on the user object.

C. Set security on the record action.

Answer: C

Explanation:

To control the visibility of a record action based on certain user conditions or expressions, the security settings of the record action itself must be configured. This involves defining conditions or expressions within the record action's security settings that evaluate whether a user meets the criteria to see and interact with the action. This method allows for dynamic control over the accessibility of actions based on user roles, specific attributes, or other contextual factors, ensuring that only authorized users can see and execute the action under specified conditions.

Reference:

Appian Documentation on Record Actions: Details how to configure and secure record actions, including visibility and permissions, to tailor the user experience within record views.

Question: 5

Which Appian feature is used to automate repetitive manual tasks such as extracting data from a system for which there is no API?

A. RPA

B. Process Mining

C. Connected Systems

Answer: A

Explanation:

Robotic Process Automation (RPA) is the Appian feature designed to automate repetitive manual

tasks, especially in scenarios where interacting with systems without APIs is necessary. RPA bots can mimic human

actions, such as navigating through system interfaces, extracting data, and entering information, effectively bridging the gap between digital processes and systems that lack API integrations. This capability is particularly useful for integrating legacy systems into modern workflows, streamlining operations that would otherwise require manual intervention.

Reference:

Appian Documentation on RPA: Explains how RPA can be leveraged within Appian to automate tasks, providing examples and best practices for implementing RPA bots in business processes.

Question: 6

Review the following expression rule: union(ri!fruit, ri!vegetables)

The rule inputs are configured as text arrays.

What is the expected output?

A. All items in ri!fruit followed by items in ri!vegetables, including duplicate values.

B. Only items that are in both ri!fruit and ri!vegetables.

C. All items in ri!fruit and ri!vegetables combined, with duplicates removed.

Answer: C

Explanation:

The union() function in Appian combines the elements of two or more arrays into a single array, removing any duplicate values. Given that the rule inputs ri!fruit and ri!vegetables are configured as text arrays, the expected output of union(ri!fruit, ri!vegetables) would be an array containing all unique items from both ri!fruit and ri!vegetables, with any duplicates removed. This function is useful for combining lists without repetition, ensuring a clean, unique set of elements.

Reference: Appian Documentation - Expression Functions

Question: 7

You need to pass data into a process from other parts of your Appian application.

Which configuration is required in your process model?

A. Toggle the Parameter field to 'True" on the configuration of a process variable.

B. Create process variables on the Data Management tab of Process Model Properties.

C. Add an interface as a Process Start Form.

Answer: B

Explanation:

To pass data into a process from other parts of an Appian application, you need to configure process variables. This is done on the Data Management tab within the Process Model Properties. Here, you can define process variables that can receive data from external sources, such as interfaces, other processes, or direct user input, when the process is started. These variables serve as placeholders for the data that will be used throughout the execution of the process.
Reference: Appian Documentation - Process Model Properties

Question: 8

HOTSPOT

Match each node to the correct description for the node.

Note: Each description will be used once. To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Answer Area

Timer Event

9 ---------- D ---------- T
kan accept multiple paths: when all paths amve, all outgoing flows are executed at the same time, i
T ---------- ^---------- T
Can be added to a flow or attached to a smart service to trigger an Exception Flow or an Escalation.

Can run automated activities.

Can accept one incoming path and select one outgoing path.

XOR Gateway

Can accept multiple paths: when all paths arrive, all outgoing flows are executed at the same time.

Can be added to a flow or attached to a smart service to trigger an Exception Flow or an Escalation.

Can run automated activities.

Can accept one incoming path and select one outgoing path.

Script Task

Can accept multiple paths: when all paths arrive, all outgoing flows are executed at the same time.

Can be added to a flow or attached to a smart service to trigger an Exception Flow or an Escalation.

Can run automated activities.

Can accept one incoming path and select one outgoing path.

AND Gateway B

Can accept multiple paths: when all paths arrive, all outgoing flows are executed at the same time.

Can be added to a flow or attached to a smart service to trigger an Exception Flow or an Escalation.

Can run automated activities.

Can accept one incoming path and select one outgoing path.

Answer:

Explanation:

Timer Event: Can accept one incoming path and select one outgoing path.

XOR Gateway: Can accept multiple paths; when all paths arrive, all outgoing flows are executed at the same time.

Script Task: Can run automated activities.

AND Gateway: Can accept multiple paths; when all paths arrive, all outgoing flows are executed at

the same time.

Timer Events in a process model are used to delay the process flow until a specific time condition is met. They are configured to have one input and one output.

XOR Gateways are decision points that can diverge the process into different paths based on conditions. If used to converge, they bring together different paths but will continue as soon as one of the incoming paths is activated.

Script Tasks are automated activities within a process that run scripts or expressions without user intervention.

AND Gateways are used to synchronize parallel flows in a process. When used to converge parallel paths, they wait for all incoming paths to complete before continuing.

Reference:

Appian Documentation: Process Modeler Objects

Question: 9

You have a record type, ABC_Author, backed by a database table.

You need to retrieve the total number of authors without loading all the data.

According to Appian best practices, which code snippet accomplishes this goal in the most efficient way?

A)

```
a! queryRecordType(
   recordType: recordType1 ABC Author
   paginginfo: a!paginglnfo(
      startindex: 1,
      batchsize: 1
   )*
   fetchTotalCount: true )
```

B)

```
a!queryRecordType(
   recordType: recordType!ABC Author, fields: {
      alaggregationfields( measures: {
            a'measure(
               field: recordType!ABC Author.fields.id, function: "COUNT", alias: "count"


   paginginfo: a!paginglnfo( startindex: 1, batchsize: 1,
```

C)

```
count(
   a!queryRecordType(
      recordType: recordType!ABC Author, fields: {
         recordType!ABC Author.fields.id

      paginginfo: a!paginglnfo(
         startindex: 1, batchsize: 5000,
      ),
   ).data
```

A. Option A

B. Option B

C. Option C

Answer: B

Explanation:

According to Appian best practices, the most efficient way to retrieve the total number of authors without loading all the data is to use a query that includes an aggregation function. Option B uses an aggregation field with a COUNT function, which is designed to return the number of rows that match a query without the need to retrieve and load all row data, thus optimizing performance.

Reference:

Appian Documentation: a!queryRecordType Function

Question: 10

You are creating a new customer onboarding application. Documents are required from customers for verification and onboarding purposes. You need to store these documents within Appian.

Which two areas in Appian should you configure? (Choose two.)

A. Knowledge Center

B. Decision Object

C. Folder

D. Feed

Answer: AC

Explanation:

In Appian, to store documents such as those required for customer onboarding, you should configure a Knowledge Center and within that, Folders. The Knowledge Center in Appian serves as the toplevel container for storing and organizing documents and folders. Folders within a Knowledge Center provide a way to categorize and manage access to documents, making them an essential part of document management in Appian applications. Decision Objects and Feeds

are not related to document storage.

Reference: Appian Documentation - Knowledge Centers and Folders

Question: 11

Where can an Appian Developer connect with and share their expertise with other Appian Developers?

A. Appian Learning Paths via Appian Academy

B. Appian Knowledge Base

C. Appian Community discussions

Answer: C

Explanation:

Appian Community discussions provide a platform for Appian Developers to connect with, share expertise, and learn from each other. The community is a vibrant space where developers can ask questions, share solutions, and discuss best practices related to Appian development. While Appian Learning Paths via Appian Academy and Appian Knowledge Base are valuable resources for learning and troubleshooting, the Community discussions specifically facilitate peer-to-peer interaction and knowledge sharing among developers.

Reference: Appian Community Website

Question: 12

You created and published a new process model.

The process model has a start form with two synchronous subprocesses with 40 and 66 nodes each. All nodes are chained from the start node through the subprocesses to the end node. After the tasks and subprocesses, there is a second User Input Task in which the user can confirm the entries and add a comment.

When testing as a normal Acme business user, you see that the confirmation screen is not shown to you.

What might be the reason for this behavior?

A. The maximum number of activity chained nodes is exceeded and breaks.

B. The second User Input Task is assigned to the process initiator.

C. The second User Input Task is assigned to the Acme business user group.

Answer: A

Explanation:

In Appian, there is a limitation on the number of activities that can be chained in a process, known as the "chaining limit." If a process model exceeds this limit, which includes synchronous subprocesses and their nodes, the process may break or not behave as expected. In this scenario, with two large subprocesses chained from start to end, the maximum number of activity chained nodes could be exceeded, resulting in the confirmation screen not being shown. Adjusting the process model to reduce chaining or using asynchronous patterns where possible can help mitigate this issue.
Reference: Appian Documentation - Process Model Best Practices

Question: 13

Your customer wants to change the name of a field of an existing Custom Data Type (CDT) to match a renamed database field.

The CDT is backed by a database entity, whose data store has the Automatically Update Database Schema option disabled. The old column name was BIRTHDATE and the new column name is DATE_OF_BIRTH.

How should you proceed?

A. Download the CDT as XSD, make the appropriate changes, and re-upload the XSD. Verify and publish the data store.

B. Rename the field in the record type in Appian to automatically update the CDT field.

C. Rename the field in the CDT in Appian. Verify and publish the data store.

Answer: C

Explanation:

When a field name in an existing Custom Data Type (CDT) needs to be changed to match a renamed database field, and the Automatically Update Database Schema option is disabled, the correct approach is to rename the

field in the CDT within Appian. After renaming the field in the CDT to match the new database column name (from BIRTHDATE to DATE_OF_BIRTH in this case), you should verify the changes and publish the data store to reflect the updates. This approach ensures that the Appian data model remains in sync with the underlying database schema.
Reference: Appian Documentation - Data Types and Data Stores

Question: 14

Which step can be critical in passing information from a form back to a process model?

A. Configure the Data Management tab.

B. Configure the activity class parameters of a Write to Data Store Entity node, a

C. Configure inputs on the Data tab of a User Input Task.

Answer: C

Explanation:

The critical step in passing information from a form back to a process model is to configure inputs on the Data tab of a User Input Task. When you create a User Input Task, it includes a form for users to interact with. The data entered into this form can be mapped to process variables via the Data tab configuration. This ensures that the information collected in the form is available to the process for further use.
Reference: Appian Documentation - User Input Tasks

Question: 15

Review the following expression rule:

```
wherecontains( ri Jname, index(ri!directory, "name")
```

ri!name is defined as "Maria".

ri!directory is defined as the following:

a!map(name: "Marla", employeeNumber: 100

a!map(name: "Steven", employeeNumber: 101

What is the expected output?

A. Maria

B. 0

C. 1

Answer: C

Explanation:

Given that ri!name is defined as "Maria" and ri!directory contains two a!map() structures, one of which includes the name "Maria," the expression wherecontains(ri!name, index(ri!directory, "name")) will evaluate as follows: The index() function will return a list of values from ri!directory for the key "name," which will be {"Maria", "Steven"}. The wherecontains() function will then check where "Maria" is found within this list. Since "Maria" is the first element, the function will return a list of indices where "Maria" is found, in this case, {1}. Appian lists are 1-indexed, so the first position is represented by 1, not 0.

Reference: Appian Expression Language Documentation - Functions

Question: 16

HOTSPOT

Match each scenario to the correct relationship type in your data model design.

Note: Each relationship type will be used once. To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Answer Area

You have a product with basic fields and an additional detailed product description withall the technical details that is regularly updated

One-to-many

One-to-one

Many-to-many

You have many employees on multiple projects.

One-to-many

One-to-one

Many-to-many

You have an order with order items

One-to-many

One-to-one

Many-to-many

Answer:

Explanation:

"You have a product with basic fields and an additional detailed product description with all the technical details that is regularly updated." - One-to-many

"You have many employees on multiple projects." - Many-to-many

"You have an order with order items." - One-to-many

A one-to-many relationship is used when a single record from one table (product) is associated with multiple records in another table (product descriptions).

A many-to-many relationship exists when multiple records from one table (employees) are associated with multiple records from another table (projects).

An order with order items also represents a one-to-many relationship where one order can have multiple order items.

Reference:

Appian Documentation: Data Modeling in Appian

Question: 17

You are configuring a Related Action for an entity-backed record type.

What is the proper domain prefix to reference the record data that will be passed into the Process Model as context for the Record Action?

A. ac!

B. pv!

C. rv!

Answer: B

Explanation:

When configuring a Related Action for an entity-backed record type, the proper domain prefix to reference the record data passed into the Process Model as context for the Record Action is pv!. This prefix stands for process variables, which are used to pass data into and out of a process model. In the context of a Related Action, pv! would be used to reference the specific process variables that are configured to receive the record data.

Reference: Appian Documentation - Process Variables and Record Types

Question: 18

HOTSPOT

Match each Appian Design Object name to the most applicable use case.

Note: Each use case will be used once or not at all. To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Answer Area

Custom Data Type (CDTl

Store a reference to a group that will be assigned s task or receive a notification.

Connect Appian to a relational database, establishing relationships between custom data types in Appian and conesponding tables in the database.

Establish a data structure into which a process model can set values prior to writing to the database.

Establish a workflow in which results are written to a database after three (3) users receive a review task in a defined order.

Process Model

Store a reference to a group that will be assigned a task or receive a notification.

Connect Appian to a relational database, establishing relationships between custom date types in Appian and cone3ponding tables in the database.

Establish a data structure into which a process model can set values prior to writing to the database.

Establish a workflow in 'which results are written to a database after three (3) users receive a review task in a defined order.

Constant

Store a reference to a group that will be assigned a task or receive a notification.

Connect Appian to a relational database, establishing relationships between custom data types in Appian and conesponding tables in the database.

Establish a data structure into which a process model can set values prior to writing to the database.

Establish a workflow in 'which results are written to a database after three (3) users receive a review task in a defined order.

Answer:

Explanation:

Custom Data Type (CDT): Connect Appian to a relational database, establishing relationships between custom data types in Appian and corresponding tables in the database.

Process Model: Establish a data structure into which a process model can set values prior to writing to the database.

Constant: Store a reference to a group that will be assigned a task or receive a notification.

Custom Data Types (CDTs) define the structure of data within Appian and can be mapped to existing database tables to leverage relational database functionalities. They establish how data is structured and stored in the Appian application.

Process Models depict the workflow and logic of a business process. They can include user and automated tasks, and they can manipulate and store data before, during, and after the execution of the process.

Constants are reusable static values that can be referenced within Appian. They can store values like group references, which can then be used for task assignment or notification purposes in process models or other expressions.

Reference:

Appian Documentation: Data Types, Process Models, Constants

Question: 19

You are configuring an employee onboarding User Input Task that will be assigned to the human resources group.

Based on the default behavior for task assignments, which statement is valid?

A. Multiple users can accept the task at the same time up until the point that the first user completes it.

B. For each user in the group, a task is generated and assigned to them to complete.

C. One user in the group can accept the task for themselves and complete it.

Answer: C

Explanation:

Based on the default behavior for task assignments in Appian, when a User Input Task is assigned to a group, any one member of the group can accept the

task. Once accepted, the task becomes locked to that user, and they are responsible for completing it. This prevents multiple users from working on the same task simultaneously and ensures clear accountability.

Reference: Appian Documentation - Task Assignments and User Input Tasks

Question: 20

You are using a local variable in an expression rule to describe the height of an applicant.

Which statement correctly describes the application of Appian best practices for naming your local variable?

A. local!hoaa - This employs the naming convention of abbreviating "Height of an applicant" to minimize both the typing required by developers and the length of code Appian is required to parse.

B. local!applicantHeight - This employs the naming convention of specifically describing the value contained by the variable.

C. local!x - This employs the naming convention of using algebraic variables for a value that may either change over time or be used by future developers for other purposes.

Answer: B

Explanation:

The best practice for naming variables in Appian is to use clear and descriptive names that convey the purpose or content of the variable. Therefore, local!applicantHeight is the best option as it precisely describes the value contained by the variable, which is the height of an applicant. This naming convention aids in readability and maintainability of the code, making it easier for developers to understand and modify the code in the future.

Reference: Appian Best Practices - Expression Writing and Naming Conventions

Question: 21

You need to start a process using an email trigger.

How should you configure this process model using the Process Model Properties dialog?

A. Go to File > Properties > Alerts. Configure the Custom Alert settings.

B. Go to File > Properties Set the proper Process Display Name

C. Go to File > Properties. Select the Public Events checkbox to allow anyone to fire triggers.

Answer: A

Explanation:

To start a process using an email trigger, you need to configure the process model to listen for an email event. This is done by going to File > Properties > Alerts in the process model properties dialog and configuring the Custom Alert settings. Here you can specify the email address that will trigger the process when an email is sent to it.

Reference: Appian Documentation - Configuring Alerts in Process Models

Question: 22

What are three ways to trigger a process from a process model? (Choose three.)

A. Use a subprocess.

B. Call a web API.

C. Use a!recordActionItem().

D. Use a Start Process smart service.

E. Run a!startProcess() from a script task.

Answer: A, D, E

Explanation:

There are multiple ways to trigger a process from a process model in Appian:

A subprocess is a process model that is called from within another process model.

A Start Process smart service is a flow element that you can use within a process model to start another process.

The a!startProcess() function can be used within a script task to start a process programmatically. These methods allow a developer to initiate a new process instance from an existing process.

Reference: Appian Documentation - Process Model Triggers and Smart Services

Question: 23

You have two record types: ACME_Student and ACME_Class.

You need to add a many-to-many relationship between these two record types.

What should you do?

A. Create a new record type with data sync enabled, then add a one-to-many relationship from each record type (ACME_Student and ACME_Class) to the new mapping record type.

B. This relationship cannot be modeled using record type relationships.

C. Add the relationship from either ACME_Student or ACME_Class, then select Many-to-Many under Relationship Type, and add the corresponding keys.

Answer: A

Explanation:

To establish a many-to-many relationship between two record types in Appian, you should create a junction or mapping record type that will handle the many-to-many association. This new record type should have two one-to-many relationships, one to each of the original record types (ACME_Student and ACME_Class). This setup allows each student to be associated with multiple classes and each class to be associated with multiple students.

Reference: Appian Documentation - Record Type Relationships

Question: 24

You need to add a new attribute to your database-backed synced Acme employee record.

What should you do?

A. Add the new field manually to the ACME_Employee database table and run a database script to resync the Acme employee record.

B. In the record data model, you add the new field in the ADD SOURCE FIELDS. Save the field with the Update Table checkbox active.

C. In the record data model, you add the new field with ADD RELATIONSHIPS. Save the newly created relationship and use the field.

Answer: B

Explanation:

To add a new attribute to a database-backed synced record type, such as an Acme employee record, you should add the new field under the 'Add Source Fields' section in the record data model configuration. When you save this new field with the 'Update Table' checkbox selected, Appian will automatically update the underlying database table and resync the record type with the new field included.

Reference: Appian Documentation - Modifying Record Data Models

Question: 25

You configured a Custom Data Type (CDT), ACME_video. You need to set up a node in your process model to write a new video as a row in the CDT's corresponding database table.

Which set of inputs must you configure on your Write to Data Store smart service node?

A. The data store and an input corresponding to each field of the ACME_video CDT.

B. The data store entity and an input of type ACME_video.

C. The data store and an input of type ACME_video.

Answer: B

Explanation:

When configuring a Write to Data Store Entity smart service node to write a new row to a CDT's corresponding database table, you must specify the data store entity where the CDT is stored. Additionally, you need to provide an input of the type that matches the CDT, in this case, an input of type ACME_video, which will contain the data to be written to the database.

Reference: Appian Documentation - Write to Data Store Entity Smart Service

Question: 26

Review the following code snippet:

index({"a", "b", "c"}, 1, "x")

Which value is returned?

A. a

B. b

C. c

D. x

Answer: D

Explanation:

The index() function in Appian is used to replace an element at a specified index in a list. In the code snippet index({"a", "b", "c"}, 1, "x"), the function is called to replace the element at index 1 of the list {"a", "b", "c"} with the string "x". Since Appian lists are 1-indexed, this means that the first element "a" will be replaced by "x". However, since the function is used to modify a list and not to return a value, if this function is used as is without being part of a larger expression that outputs the list, it

would not return any of the listed values.

Reference: Appian Expression Language Documentation - Functions

Question: 27

Which set of out-of-the-box features is only available when data sync is enabled on a record type?

A. Generate record actions

Define record type object security

Add custom record fields

B. Define record type relationships

Add custom record fields

Configure record-level security

C. Define record type relationships

Add hidden record fields

Configure record-level security

Answer: C

Explanation:

Data sync enables additional features for record types in Appian. With data sync enabled, you can define relationships between different record types, add fields to a record type that do not appear in the source database (hidden fields), and configure record-level security to control access to individual records based on user roles or other criteria. These features are part of the enhanced functionality provided by data sync to ensure efficient data management and security within Appian applications.

Reference: Appian Documentation - Record Type Features and Data Sync

Question: 28

Which statement about local variables is valid?

A. The data type of a local variable is determined by its value.

B. Local variables can only store primitive data types, such as numbers and strings.

C. Local variables must have an initial value set when defining them.

Answer: A

Explanation:

In Appian, the data type of a local variable is inferred from the value it is set to. Unlike some other programming languages where the data type must be explicitly declared, Appian determines the data type automatically based on the initial value assigned to the local variable. Local variables in Appian are quite flexible and can store various types of data, including complex data types, not just primitive ones.

Reference: Appian Documentation - Local Variables

Question: 29

Which Appian feature can help the implementation team analyze the event log data of an existing process?

A. RPA

B. Process Mining

C. Workflow

Answer: B

Explanation:

vent logs. It uses the data from process models to give insights into process performance, bottlenecks, and compliance with the designed process flow. This feature is valuable for

understanding the actual performance of business processes and for identifying areas for improvement.

Reference: Appian Documentation - Process Mining

Question: 30

Review the following code snippet:

displayvalue(1, {0, 1, 2}, {"Low", "Medium", "High"}, "Unknown")

The definition of displayvalue is:

Tries to match a value in a given array with a value at the same index in a replacement array and returns either the value at the same index or a default value if the value is not found.

What does this code snippet return?

A. Low

B. Medium

C. High

Answer: B

Explanation:

The displayvalue() function matches a given value with an array and returns the value at the same index from a replacement array or a default value if not found. In the snippet displayvalue(1, {0, 1, 2}, {"Low", "Medium", "High"}, "Unknown"), the value 1 is matched in the array {0, 1, 2} at index 1. The function then returns the value at index 1 in the array {"Low", "Medium", "High"}, which is "Medium".

Reference: Appian Expression Language Documentation - Functions

Question: 31

You receive a bug ticket that states "After selecting a value for the drop-down field, the value disappears."

You investigate and notice that when you select the drop-down, the proper choice labels display.

When you select an option, the value updates properly in the corresponding rule input.

What is the issue and how can you fix this bug?

A. The value parameter is improperly configured on the drop-down component. You need to map the value to the proper rule input or variable.

B. The user group for the lookup table is incorrect. You need to add the user to the proper group.

C. The choice labels parameter of the drop-down field is not configured as a list. You need to wrap the value with curly brackets.

Answer: A

Explanation:

The described bug typically occurs when the value parameter of the drop-down component is not correctly mapped to the corresponding rule input or variable that is supposed to hold the selected value. To fix the issue, you should ensure that the drop-down's value parameter is correctly mapped so that the selected option is retained and displayed properly.
Reference: Appian Documentation - Dropdown Field Component

Question: 32

ACME Automobile uses Appian to manage their vehicle fleet. Vehicle records can have a status of either "active" or "inactive".

Users are primarily concerned with active vehicles and want to see only those records by default when viewing the Vehicle records list. However, it is important for users to be able to see the unfiltered list of Vehicle records on demand to address occasional auditing requests from managers.

Which configuration supports the desired Vehicle record list behavior?

A. Visibility on the Status column in the Vehicle record list set with conditional logic.

B. A source filter set to exclude vehicles with status "inactive".

C. A user filter for the status field with a default option corresponding to "active".

Answer: C

Explanation:

To achieve the behavior where users see only "active" vehicle records by default but can also view all records when needed, you should configure a user filter for the status field on the Vehicle record list. This user filter should have a default value set to "active", which will filter the list to only show active records initially. However, users will still have the option to adjust the filter to see all records, thus accommodating occasional auditing requests.

Reference: Appian Documentation - Record List Filters and User Filters

Question: 33

You write an expression that checks the validity of an email address.

Which three scenarios should you configure as test cases? (Choose three.)

A. An invalid email address that is missing the @-character: "john.doeexample.com".

B. An invalid email address: null.

C. A valid email address: "jane.doe@example.com".

D. The mail server is unavailable.

Answer: A, B, C

Explanation:

When testing the validity of an email address in Appian, you should consider various scenarios to ensure robust validation. This includes checking for the presence of the @-character, which is essential for a valid email format (A), testing how the expression handles null values (B), and verifying that a correctly formatted email address is validated as such (C). The availability of the mail

server (D) is not relevant to the format validation of an email address within an expression. Reference: Appian Documentation - Expression Language Syntax

Question: 34

You are creating a new interface object to display a pie chart.

The data for the chart is stored in a local variable in the parent interface object which references your child interface.

In terms of performance, what is the most efficient method to access the data required for the pie chart?

A. Create a rule input on the child interface and pass the local variable data from the parent interface.

B. Reference the local variable directly from the child Interface using a process report.

C. Query the data separately in the child interface to avoid passing it from the parent interface.

Answer: A

Explanation:

The most efficient method to access data for a pie chart in a child interface is to create a rule input on the child interface and pass the local variable data from the parent interface. This method avoids redundant data queries and takes advantage of Appian's pass-by-reference mechanism, which does not duplicate data in memory when passing it to the child interface.

Reference: Appian Documentation - Interface Design

Question: 35

When applying a default filter to a record type, what is a true statement for excluded data?

A. The data does not show up in the record list. End-users are unable to clear the filter condition to view the data.

B. The data does not show up in the record list, but can be accessed by end-users using a direct link to a record view.

C. The data does not show up in the record list, but end-users can clear the filter condition to view the data.

Answer: C

Explanation:

When a default filter is applied to a record type in Appian, the data excluded by that filter does not appear in the default record list. However, end-users with appropriate permissions can modify or clear the filter conditions to view the excluded data. This behavior allows for a flexible and user- driven data exploration experience within Appian record lists.

Reference: Appian Documentation - Record Lists and Filters

Question: 36

How can you prevent users from accessing Tempo?

A. Remove the users from the Tempo Users system group.

B. Change the default User Start Page.

C. Ensure the user is in the Application Users group, which by default does not have access to Tempo.

Answer: A

Explanation:

To prevent users from accessing Tempo, you should remove them from the Tempo Users system group. This group controls access to the Tempo interface in Appian. By removing users from this group, they will no longer have the necessary permissions to access Tempo features and content. Reference: Appian Documentation - System Groups and User Access

Question: 37

You are creating a form used to order a pizz

a. You use a radio button component for the selection.

The pizza selection labels include a list of toppings. You do not want the selection labels to be truncated.

Which layout should you choose?

A. Compact

B. Grid

C. Stacked

Answer: C

Explanation:

For a pizza ordering form where you do not want the radio button selection labels to be truncated, the Stacked layout is the most appropriate. This layout will list the options vertically, giving each one adequate space and preventing truncation, which is particularly useful when the labels include longer text, such as a list of toppings.

Reference: Appian Documentation - Interface Components

Question: 38

A user needs to navigate from a record summary to an external URL.

Which interface component can be used to support this goal?

A. Button

B. Record Link

C. Card Layout with a link

Answer: A

Explanation:

In Appian, to navigate from a record summary to an external URL, you can use a Button component configured with a 'Link' action. This approach allows you to define a URL that the button will navigate to when clicked. The Button component offers flexibility in terms of design and functionality, making it suitable for such navigation purposes within Appian interfaces. You can specify the URL directly in the Button's properties, allowing for dynamic link generation based on record data if needed. Reference:

Appian Documentation: Designing Interfaces (This section provides comprehensive details on using various interface components, including buttons, to achieve specific user interaction goals within Appian.)

Appian Documentation: Buttons (This page specifically focuses on the Button component, detailing its properties, usage, and how to configure it for different actions, including navigating to an external URL.)

Question: 39

You are developing an expression rule. You need to find information on employing an Appian function that you have not used before.

For more information on the Appian function, what should you do first?

A. Look up the function in the Appian Documentation.

B. Search the Appian Knowledge Base Articles.

C. Post a question on Appian Community.

Answer: A

Explanation:

When you need information on using a specific Appian function that you have not used before, the first step should be to consult the Appian Documentation. The documentation provides comprehensive details on each function, including syntax, parameters, usage examples, and best practices, which is essential for understanding how to correctly employ the function in an expression. Reference: Appian Documentation - Functions

Question: 40

Which code snippet calls the interface APP_RecordDashboard while following best practices for passing in values for "recordId" and "firstName"?

A)

```
rule!APP_RecordDashboard(
        1, "Kyle
```

B)

```
rule!APP_RecordOashhoard(
        recordId: 1, firstName: "Kyle"
```

C)

```
rule!APP RecordOashhoard(
        recordId » 1, firstName >> "Kyle"
```

A. Option A

B. Option B

C. Option C

Answer: B

Explanation:

The best practice in Appian for passing values into an interface is to use named parameters, which is demonstrated by Option B. Named parameters make the code more readable and maintainable by clearly specifying which parameter each value is being passed to. In this case, the recordId and firstName parameters are clearly being assigned the values 1 and "Kyle" respectively.

Reference:

Appian Documentation: Passing Parameters to Interfaces

Question: 41

You are configuring a local variable on an interface to store the date and time that the username field was last modified. The local variables are currently configured as follows:

```
local l username,
locall usernameLastModified: a!refreshVariable(
      value: now()
```

Which a!refreshVariable configuration should be added so that local!usernameLastModified stores the correct timestamp?

A. refreshInterval: 0

B. refreshOnVarChange: local!username

C. refreshAlways: true

Answer: B

Explanation:

The a!refreshVariable function should be configured to refresh when the local!username variable changes. This is achieved by setting the refreshOnVarChange parameter to local!username, which will update the local!usernameLastModified variable with the current timestamp whenever local!username is modified.

Reference:

Appian Documentation: a!refreshVariable Function

Question: 42

HOTSPOT

Review the following variables:

local'groupA: {6, 8, 10, 12} locallgroupB: {3, 6, 9, 12}

Match each expression rule to the expected output.

Note: Each output will be used once or not at all. To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Answer Area

diffsrenc s.loc all groupA, locallgraupB)

S{6. 0 '0 '2 3. 9}° S
fp, 10} ° T
{6.12}

unioni local IcrcupA locallgraupB)

{6,0 10 12 3,9}

{8,10}

{6.12}

intersectonl local IgrcupA local! groupBi

{6,3 10 '2 3,9}

{3.10}

{6.12}

Answer:

Explanation:

Answer Area

difference(local!groupA, localigroupB)

{6,8,10,12, 3, 9}

{6,12}

union(local!groupA, localigroupB)

{6,8,10,12, 3, 9}

MW

{6,12}

intersection(local!groupA, localigroupB)

{6,8,10,12, 3, 9}

MW

{6,12}

The difference function returns a list of elements that are present in the first list but not in the second. For loc!groupA and loc!groupB, the numbers 8 and 10 are present in loc!groupA but not in loc!groupB.

The union function combines the elements from both lists without duplication. Combining loc!groupA and loc!groupB includes the numbers 3, 6, 8, 9, 10, and 12.

The intersection function returns a list of elements that are present in both lists. For loc!groupA and loc!groupB, the numbers 6 and 12 are common to both.

Reference:

Appian Documentation: Functions (difference, union, intersection)

Question: 43

How do you refer to metadata of a process model object within process model expressions? For example: the creator, name, or description.

A. Process Properties, referred to in process model expressions using the pp! domain.

B. Process Variables, referred to in process model expressions using the pv! domain.

C. Process Model Properties, referred to in process model expressions using the pm! domain.

Answer: C

Explanation:

Process Model Properties within Appian are referred to using the pm! domain prefix in process model expressions. This includes metadata such as the creator, name, or description of the process model.

Reference:

Appian Documentation: Process Model Properties

Question: 44

You receive the following error message after creating a dropdown field:

A dropdown component has an Invalid value for "choicevalues".
The choiceLabels and ehoiceValues arrays must be the same length, but choice

What could be the problem?

A. The choiceLabels and choiceValues datatypes do not match.
B. The choiceValues has too few values.
C. The choiceLabels have too few labels.

Answer: B

Explanation:

The error message indicates that the choiceValues array is not the same length as the choiceLabels array. In this scenario, there are more labels than values, which means the choiceValues array needs additional values to match the number of labels.

Reference:

Appian Documentation: Dropdown Field

Question: 45

You are working on a process model "VIM Update Vehicle."

You want to call another process "VIM Get Service Date" that accepts pv!vehicleId as a process parameter and sets a value for pv!serviceDate. The next node in VIM Update Vehicle depends on the value of pv!serviceDate.

Which node should you use to execute "VIM Get Service Date" from VIM Update Vehicle?

A. Start Process smart service

B. Asynchronous subprocess with activity chaining

C. Synchronous subprocess with input and output variables configured

Answer: C

Explanation:

When a process model depends on the value of a variable being set by another process, a synchronous subprocess should be used. This ensures that the calling process (VIM Update Vehicle) will wait for the subprocess (VIM Get Service Date) to complete and set the necessary pv!serviceDate value before continuing. The subprocess node should be configured with input and output variables

to pass the pv!vehicleId and receive the pv!serviceDate. Reference:
Appian Documentation: Subprocess Node

Question: 46

Which two scenarios are ideal for using Appian Portals? (Choose two.)

A. A manager wants to obtain a view of their team's performance.

B. A retail customer wants to conduct a public survey for their recently launched product.

C. An employee who does not have an account wants to register for their company's vehicle fleet a management system.

D. A user needs to submit support requests when they are using their mobile device in areas with bad network coverage.

Answer: B, C

Explanation:

Appian Portals are designed for scenarios where users who do not have an Appian account need to interact with Appian applications. This makes them ideal for public-facing applications such as surveys (Option B) or for allowing external users to initiate processes like registrations (Option C). They are not intended for internal use by employees (Option A) or for scenarios requiring offline capabilities (Option D).

Reference:

Appian Documentation: Appian Portals

Question: 47

You have two Custom Data Types (CDT): ACME_invoice and ACME_invoiceItem that have a flat relationship. The invoice item table has a field that is a foreign key to the invoice table. You are leveraging the database to automatically generate their primary keys.

How should you structure the process model to add a new invoice and the new invoice items to the system?

A.

1. Write to Multiple Data Store Entities smart service (Writing to the ACME_invoiceItem table and ACME_invoice table).

8. 1. Write to Data Store Entity smart service (Writing to the ACME_invoiceItem table).

2. Script Task to update foreign keys.

3. Write to Data Store Entity smart service (Writing to the ACME_invoice table).

C.

1. Write to Data Store Entity smart service (Writing to the ACME_invoice table).

2. Script Task to update foreign keys.

3. Write to Data Store Entity smart service (Writing to the ACME_invoiceItem table).

Answer: C

Explanation:

When dealing with related data types where one has a foreign key to another, you must first create the record in the primary table (ACME_invoice) and then use the generated primary key to create related records in the secondary table (ACME_invoiceItem). This is why you first write to the ACME_invoice table, then update the foreign keys in a Script Task, and finally write to the ACME_invoiceItem table.

Reference:

Appian Documentation: Relational Databases

Question: 48

After selecting a record, a user wants to initiate an activity in the context of that selected record.
You start by creating the process model that implements this activity.

What should you do next?

A. Add the process model as a record list action to that record.

B. Configure a site page as an action to kick off the process model.

C. Add the process model as a record related action to that record.

Answer: C

Explanation:

a record related action. This allows the action to be performed in the context of a specific record, making it straightforward for users to initiate processes directly from the record view.

Reference:

Appian Documentation: Record Related Actions

Question: 49

An interface references an expression rule.

What are the relationships between these objects?

A. Dependents and Reliants

B. Dependents and Precedents

C. Inheritance and Association

Answer: B

Explanation:

In Appian, when an interface references an expression rule, the interface is considered a dependent

because it depends on the expression rule to function correctly. Conversely, the expression rule is a precedent because the interface relies on it.

Reference:

Appian Documentation: Managing Application Contents

Question: 50

You need to be able to define record type relationships.

What is a required prerequisite in Appian?

A. The record types must have data sync enabled.

B. The record types must be on a virtualized data source.

C. The record types must be stored in the local Appian business database.

Answer: A

Explanation:

To define record type relationships in Appian, it's required that the record types have data sync enabled. This is necessary to ensure that the data is readily available in Appian for the relationships to be established and maintained.

Reference:

Appian Documentation: Record Type Relationships

Question: 51

You need to create a record type with data sync enabled.

What are the supported data sources?

A. Web Services, Salesforce, Database

B. Web Services, Process Reports, Database

C. Salesforce, Process Models, Database

Answer: C

Explanation:

Appian's data sync feature supports synchronization from three primary data sources: Salesforce, Process Models, and Databases. This allows for the creation of record types that can efficiently access and manipulate data from these sources, leveraging Appian's capabilities to provide real-time insights and interactions with external systems and internal processes.

Reference:

Appian Documentation: Creating Record Types

Question: 52

How can you add test data into your rule inputs while editing an interface object?

A. Select the Performance tab and set test values.

B. Select the Test button and set test values.

C. Select the Gear icon, select Properties, and set test values.

Answer: B

Explanation:

While editing an interface object in Appian, you can add test data into your rule inputs by selecting the Test button. This feature allows you to simulate how the interface would behave with specific inputs, facilitating a more efficient and accurate design and debugging process.

Reference:

Appian Documentation: Testing Interfaces

Question: 53

Which two groups can be set within Application Properties? (Choose two.)

A. Administrators Groups

B. Developers Groups

C. Users Groups

D. Designers Groups

Answer: A, C

Explanation:

Within Application Properties in Appian, you can set two groups: Administrators Groups and Users Groups. The Administrators Group is responsible for managing and configuring the application, while the Users Group is designated for end-users who interact with the application's functionalities.

Reference:

Appian Documentation: Application Properties

Question: 54

What is an Appian best practice for calling interface rules on your interface?

A. Call the interface rule on a rule input.

B. Use keyword syntax.

C. Always use consistent ordering of rule parameters.

Answer: C

Explanation:

An Appian best practice for calling interface rules within your interfaces is to always use a consistent ordering of rule

parameters. This practice enhances the readability and maintainability of your interfaces, ensuring that other developers can understand and modify the interface more easily. Reference:

Appian Documentation: Designing Interfaces

Question: 55

You are running a process instance and an error occurs on an unattended node.

What happens to your process when this error occurs?

A.

- The process is not paused.
- Parallel paths in the process continue to proceed.
- An alert is sent to the appropriate recipients (usually admins).
- These nodes are not included in the num_problem_tasks process metric in process reports.

B.

- The process is paused.
- Parallel paths in the process are stopped.
- No alerts are sent.
- These nodes are included in the num_problem_tasks process metric in process reports.

C. • The process is paused.

- Parallel paths in the process are stopped.
- An alert Is sent to the process initiator.
- These nodes are included in the num_problem_tasks process metric in process reports.

Answer: A

Explanation:

When an error occurs on an unattended node in a running process instance:

The process is not paused, allowing other parallel paths to continue.

An alert is typically sent to appropriate recipients, such as administrators, to notify them of the issue.

These nodes are not counted in the num_problem_tasks process metric, which is used in process reports to track problematic tasks.

This behavior ensures that the rest of the process can continue as intended while alerting administrators to intervene and address the error.

Reference:

Appian Documentation: Error Handling in Processes

Question: 56

Which option best describes the primary purpose of the interface design object?

A. Provides a method for an Appian application to interact with an external application or service.

B. Provides a method for end users to interact with an Appian application.

C. Provides a method for an Appian application to interact with a database.

Answer: B

Explanation:

The primary purpose of the interface design object in Appian is to provide a method for end users to interact with an Appian application. Interfaces are the user-facing components that allow for the input, display, and manipulation of data within the application, facilitating user engagement and task completion.

Reference:

Appian Documentation: Interface Design

Question: 57

You received a support ticket where the user claims that nothing happens when they click the button to complete a task. You confirm that the user is assigned to the task.

What is a possible reason for this problem?

A. The user was not in the user group configured for the process model security.

B. The process report that drives the runtime database does not have security set properly.

C. The button was not configured to submit the form.

Answer: C

Explanation:

If a user claims that nothing happens when they click a button to complete a task, a possible reason could be that the button was not configured to submit the form. This configuration is essential for triggering the form submission action, which in turn can complete the task or initiate further processes. Without this setup, clicking the button will not result in any action.

Reference:

Appian Documentation: Configuring Buttons in Interfaces

Question: 58

You want to add a script task in between two User Input Tasks assigned to the same user.

What needs to be configured so that the user sees the second form immediately after submitting the first?

A. Set all process variables as parameters.

B. Enable activity chaining.

C. Run the script task as "synchronous."

Answer: B

Explanation:

To ensure that the user sees the second form immediately after submitting the first, when adding a script task between two User Input Tasks assigned to the same user, it is essential to enable activity chaining. Activity chaining in Appian allows for the seamless transition between user tasks within a process, eliminating unnecessary delays and enhancing the user experience by immediately presenting the subsequent task.

Reference:

Appian Documentation: Activity Chaining in Processes

Question: 59

What is the Production environment used for?

A. Allowing business users to test the application.

B. Allowing developers to make updates to the application.

C. Allowing business users to use the final version of the application.

Answer: C

Explanation:

The Production environment in Appian is used for allowing business users to use the final version of the application. This environment is where fully developed, tested, and approved applications are deployed for end-user interaction. It is the live environment that supports actual business operations, ensuring that users have access to stable and reliable application functionalities for their day-to-day tasks.

Reference:

Appian Documentation: Application Environments