

ACD-301 Test Guide - Appian Certified Lead Developer Study Question & ACD-301 Exam Questions

The safer, easier way to help you pass any IT exams.

Appian ACD301 Exam

Appian Lead Developer

<https://www.passquestion.com/acd301.html>



Pass Appian ACD301 Exam with PassQuestion ACD301 questions and answers in the first attempt.

<https://www.passquestion.com/>

1 / 15

Our Software version has the advantage of simulating the real ACD-301 exam environment. Many candidates can't successfully pass their real ACD-301 exams for the reason that they are too nervous to performance rightly as they do the practices. This Software version of ACD-301 practice materials will exactly help overcome their psychological fear. Besides, the scores will show out when you finish the practice, so after a few times, you will definitely do it better and better. You will be bound to pass your ACD-301 Exam since you have perfected yourself in taking the ACD-301 exam.

The world is changing rapidly and the requirements to the employees are higher than ever before. If you want to find an ideal job and earn a high income you must boost good working abilities and profound major knowledge. Passing ACD-301 certification can help you realize your dreams. If you buy our product, we will provide you with the best Appian Certification Program study materials and it can help you obtain ACD-301 certification. Our product is of high quality and our service is perfect.

>> ACD-301 Authentic Exam Hub <<

Appian ACD-301 Review Guide & ACD-301 Intereactive Testing Engine

Our ACD-301 guide torrent has gone through strict analysis and summary according to the past exam papers and the popular trend in the industry and are revised and updated. The ACD-301 exam questions have simplified the sophisticated notions. The software boosts varied self-learning and self-assessment functions to check the learning results. The software of our ACD-301 Test Torrent provides the statistics report function and help the students find the weak links and deal with them. With this version of our ACD-

301 exam questions, you will be able to pass the exam easily.

Appian Certified Lead Developer Sample Questions (Q21-Q26):

NEW QUESTION # 21

You are taking your package from the source environment and importing it into the target environment.

Review the errors encountered during inspection:

What is the first action you should take to Investigate the issue?

- A. Check whether the object (UUID ending in 18028821) is included in this package
- B. Check whether the object (UUID ending in 18028931) is included in this package
- C. Check whether the object (UUID ending in 7t00000i4e7a) is included in this package
- D. Check whether the object (UUID ending in 25606) is included in this package

Answer: C

Explanation:

The error log provided indicates issues during the package import into the target environment, with multiple objects failing to import due to missing precedents. The key error messages highlight specific UUIDs associated with objects that cannot be resolved. The first error listed states:

"TEST_ENTITY_PROFILE_MERGE_HISTORY": The content [id=uuid-a-0000m5fc-f0e6-8000-9b01-011c48011c48, 18028821] was not imported because a required precedent is missing: entity [uuid=a-0000m5fc-f0e6-8000-9b01-011c48011c48, 18028821] cannot be found..." According to Appian's Package Deployment Best Practices, when importing a package, the first step in troubleshooting is to identify the root cause of the failure. The initial error in the log points to an entity object with a UUID ending in 18028821, which failed to import due to a missing precedent. This suggests that the object itself or one of its dependencies (e.g., a data store or related entity) is either missing from the package or not present in the target environment.

Option A (Check whether the object (UUID ending in 18028821) is included in this package): This is the correct first action. Since the first error references this UUID, verifying its inclusion in the package is the logical starting point. If it's missing, the package export from the source environment was incomplete. If it's included but still fails, the precedent issue (e.g., a missing data store) needs further investigation.

Option B (Check whether the object (UUID ending in 7t00000i4e7a) is included in this package): This appears to be a typo or corrupted UUID (likely intended as something like "7t000014e7a" or similar), and it's not referenced in the primary error. It's mentioned later in the log but is not the first issue to address.

Option C (Check whether the object (UUID ending in 25606) is included in this package): This UUID is associated with a data store error later in the log, but it's not the first reported issue.

Option D (Check whether the object (UUID ending in 18028931) is included in this package): This UUID is mentioned in a subsequent error related to a process model or expression rule, but it's not the initial failure point.

Appian recommends addressing errors in the order they appear in the log to systematically resolve dependencies. Thus, starting with the object ending in 18028821 is the priority.

NEW QUESTION # 22

You are deciding the appropriate process model data management strategy.

For each requirement, match the appropriate strategies to implement. Each strategy will be used once.

Note: To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Answer:

Explanation:

NEW QUESTION # 23

You are designing a process that is anticipated to be executed multiple times a day. This process retrieves data from an external system and then calls various utility processes as needed. The main process will not use the results of the utility processes, and there are no user forms anywhere.

Which design choice should be used to start the utility processes and minimize the load on the execution engines?

- A. Use Process Messaging to start the utility process.
- B. Start the utility processes via a subprocess synchronously.
- C. Start the utility processes via a subprocess asynchronously.

- D. Use the Start Process Smart Service to start the utility processes.

Answer: C

Explanation:

Comprehensive and Detailed In-Depth Explanation:

As an Appian Lead Developer, designing a process that executes frequently (multiple times a day) and calls utility processes without using their results requires optimizing performance and minimizing load on Appian's execution engines. The absence of user forms indicates a backend process, so user experience isn't a concern-only engine efficiency matters. Let's evaluate each option:

A . Use the Start Process Smart Service to start the utility processes:

The Start Process Smart Service launches a new process instance independently, creating a separate process in the Work Queue. While functional, it increases engine load because each utility process runs as a distinct instance, consuming engine resources and potentially clogging the Java Work Queue, especially with frequent executions. Appian's performance guidelines discourage unnecessary separate process instances for utility tasks, favoring integrated subprocesses, making this less optimal.

B . Start the utility processes via a subprocess synchronously:

Synchronous subprocesses (e.g., `a!startProcess` with `isAsync: false`) execute within the main process flow, blocking until completion. For utility processes not used by the main process, this creates unnecessary delays, increasing execution time and engine load. With frequent daily executions, synchronous subprocesses could strain engines, especially if utility processes are slow or numerous.

Appian's documentation recommends asynchronous execution for non-dependent, non-blocking tasks, ruling this out.

C . Use Process Messaging to start the utility process:

Process Messaging (e.g., `sendMessage()` in Appian) is used for inter-process communication, not for starting processes. It's designed to pass data between running processes, not initiate new ones. Attempting to use it for starting utility processes would require additional setup (e.g., a listening process) and isn't a standard or efficient method. Appian's messaging features are for coordination, not process initiation, making this inappropriate.

D . Start the utility processes via a subprocess asynchronously:

This is the best choice. Asynchronous subprocesses (e.g., `a!startProcess` with `isAsync: true`) execute independently of the main process, offloading work to the engine without blocking or delaying the parent process. Since the main process doesn't use the utility process results and there are no user forms, asynchronous execution minimizes engine load by distributing tasks across time, reducing Work Queue pressure during frequent executions. Appian's performance best practices recommend asynchronous subprocesses for non-dependent, utility tasks to optimize engine utilization, making this ideal for minimizing load.

Conclusion: Starting the utility processes via a subprocess asynchronously (D) minimizes engine load by allowing independent execution without blocking the main process, aligning with Appian's performance optimization strategies for frequent, backend processes.

Appian Documentation: "Process Model Performance" (Synchronous vs. Asynchronous Subprocesses).

Appian Lead Developer Certification: Process Design Module (Optimizing Engine Load).

Appian Best Practices: "Designing Efficient Utility Processes" (Asynchronous Execution).

NEW QUESTION # 24

Your Agile Scrum project requires you to manage two teams, with three developers per team. Both teams are to work on the same application in parallel. How should the work be divided between the teams, avoiding issues caused by cross-dependency?

- A. Group epics and stories by technical difficulty, and allocate one team the more challenging stories.
- B. Have each team choose the stories they would like to work on based on personal preference.
- **C. Group epics and stories by feature, and allocate work between each team by feature.**
- D. Allocate stories to each team based on the cumulative years of experience of the team members.

Answer: C

Explanation:

Comprehensive and Detailed In-Depth Explanation:

In an Agile Scrum environment with two teams working on the same application in parallel, effective work division is critical to avoid cross-dependency, which can lead to delays, conflicts, and inefficiencies. Appian's Agile Development Best Practices emphasize team autonomy and minimizing dependencies to ensure smooth progress.

Option B (Group epics and stories by feature, and allocate work between each team by feature):

This is the recommended approach. By dividing the application's functionality into distinct features (e.g., Team 1 handles customer management, Team 2 handles campaign tracking), each team can work independently on a specific domain. This reduces cross-dependency because teams are not reliant on each other's deliverables within a sprint. Appian's guidance on multi-team projects suggests feature-based partitioning as a best practice, allowing teams to own their backlog items, design, and testing without frequent coordination. For example, Team 1 can develop and test customer-related interfaces while Team 2 works on campaign processes, merging their work during integration phases.

Option A (Group epics and stories by technical difficulty, and allocate one team the more challenging stories):

This creates an imbalance, potentially overloading one team and underutilizing the other, which can lead to morale issues and uneven progress. It also doesn't address cross-dependency, as challenging stories might still require input from both teams (e.g., shared data models), increasing coordination needs.

Option C (Allocate stories to each team based on the cumulative years of experience of the team members):

Experience-based allocation ignores the project's functional structure and can result in mismatched skills for specific features. It also risks dependencies if experienced team members are needed across teams, complicating parallel work.

Option D (Have each team choose the stories they would like to work on based on personal preference):

This lacks structure and could lead to overlap, duplication, or neglect of critical features. It increases the risk of cross-dependency as teams might select interdependent stories without coordination, undermining parallel development.

Feature-based division aligns with Scrum principles of self-organization and minimizes dependencies, making it the most effective strategy for this scenario.

NEW QUESTION # 25

You have an active development team (Team A) building enhancements for an application (App X) and are currently using the TEST environment for User Acceptance Testing (UAT).

A separate operations team (Team B) discovers a critical error in the Production instance of App X that they must remediate.

However, Team B does not have a hotfix stream for which to accomplish this. The available environments are DEV, TEST, and PROD.

Which risk mitigation effort should both teams employ to ensure Team A's capital project is only minorly interrupted, and Team B's critical fix can be completed and deployed quickly to end users?

- A. Team B must address changes in the TEST environment. These changes can then be tested and deployed directly to PROD. Once the deployment is complete, Team B can then communicate their changes to Team A to ensure they are incorporated as part of the next release.
- B. Team B must address the changes directly in PROD. As there is no hotfix stream, and DEV and TEST are being utilized for active development, it is best to avoid a conflict of components. Once Team A has completed their enhancements work, Team B can update DEV and TEST accordingly.
- C. Team A must analyze their current codebase in DEV to merge the hotfix changes into their latest enhancements. Team B is then required to wait for the hotfix to follow regular deployment protocols from DEV to the PROD environment.
- D. Team B must communicate to Team A which component will be addressed in the hotfix to avoid overlap of changes. If overlap exists, the component must be versioned to its PROD state before being remediated and deployed, and then versioned back to its latest development state. If overlap does not exist, the component may be remediated and deployed without any version changes.

Answer: D

Explanation:

Comprehensive and Detailed In-Depth Explanation:

As an Appian Lead Developer, managing concurrent development and operations (hotfix) activities across limited environments (DEV, TEST, PROD) requires minimizing disruption to Team A's enhancements while ensuring Team B's critical fix reaches PROD quickly. The scenario highlights no hotfix stream, active UAT in TEST, and a critical PROD issue, necessitating a strategic approach. Let's evaluate each option:

A . Team B must communicate to Team A which component will be addressed in the hotfix to avoid overlap of changes. If overlap exists, the component must be versioned to its PROD state before being remediated and deployed, and then versioned back to its latest development state. If overlap does not exist, the component may be remediated and deployed without any version changes: This is the best approach. It ensures collaboration between teams to prevent conflicts, leveraging Appian's version control (e.g., object versioning in Appian Designer). Team B identifies the critical component, checks for overlap with Team A's work, and uses versioning to isolate changes. If no overlap exists, the hotfix deploys directly; if overlap occurs, versioning preserves Team A's work, allowing the hotfix to deploy and then reverting the component for Team A's continuation. This minimizes interruption to Team A's UAT, enables rapid PROD deployment, and aligns with Appian's change management best practices.

B . Team A must analyze their current codebase in DEV to merge the hotfix changes into their latest enhancements. Team B is then required to wait for the hotfix to follow regular deployment protocols from DEV to the PROD environment:

This delays Team B's critical fix, as regular deployment (DEV → TEST → PROD) could take weeks, violating the need for "quick deployment to end users." It also risks introducing Team A's untested enhancements into the hotfix, potentially destabilizing PROD. Appian's documentation discourages mixing development and hotfix workflows, favoring isolated changes for urgent fixes, making this inefficient and risky.

C . Team B must address changes in the TEST environment. These changes can then be tested and deployed directly to PROD. Once the deployment is complete, Team B can then communicate their changes to Team A to ensure they are incorporated as part of the next release:

Using TEST for hotfix development disrupts Team A's UAT, as TEST is already in use for their enhancements. Direct deployment from TEST to PROD skips DEV validation, increasing risk, and doesn't address overlap with Team A's work. Appian's deployment guidelines emphasize separate streams (e.g., hotfix streams) to avoid such conflicts, making this disruptive and unsafe.

D . Team B must address the changes directly in PROD. As there is no hotfix stream, and DEV and TEST are being utilized for active development, it is best to avoid a conflict of components. Once Team A has completed their enhancements work, Team B can update DEV and TEST accordingly:

Making changes directly in PROD is highly discouraged in Appian due to lack of testing, version control, and rollback capabilities, risking further instability. This violates Appian's Production governance and security policies, and delays Team B's updates until Team A finishes, contradicting the need for a "quick deployment." Appian's best practices mandate using lower environments for changes, ruling this out.

Conclusion: Team B communicating with Team A, versioning components if needed, and deploying the hotfix (A) is the risk mitigation effort. It ensures minimal interruption to Team A's work, rapid PROD deployment for Team B's fix, and leverages Appian's versioning for safe, controlled changes-aligning with Lead Developer standards for multi-team coordination.

Appian Documentation: "Managing Production Hotfixes" (Versioning and Change Management).

Appian Lead Developer Certification: Application Management Module (Hotfix Strategies).

Appian Best Practices: "Concurrent Development and Operations" (Minimizing Risk in Limited Environments).

NEW QUESTION # 26

.....

Test4Cram will provides the facility of online chat to all prospective customers to discuss any issue regarding, different vendors' certification tests, ACD-301 exam materials, discount offers etc. Our efficient staff is always prompt to respond you. If you need detailed answer, you send emails to our customers' care department, we will help you solve your problems as soon as possible. You will never regret to choose ACD-301 Exam Materials.

ACD-301 Review Guide: https://www.test4cram.com/ACD-301_real-exam-dumps.html

Appian ACD-301 Authentic Exam Hub We can promise that the online version will not let you down, With Test4Cram, you can access a wide range of features designed to provide you with the right resources and guidance for acing the Appian Certified Lead Developer (ACD-301) Exam, Best Appian ACD-301 Dumps PDF To Help You Out In Your Exam, Now the Appian Certified Lead Developer ACD-301 Exam Questions have become the first choice for instant and complete ACD-301 exam preparation.

It was both fascinating and fun, All Collections are private by ACD-301 default—only you can see them, unless you choose to share them, We can promise that the online version will not let you down.

Marvelous ACD-301 Learning Engine demonstrates high-effective Exam Materials - Test4Cram

With Test4Cram, you can access a wide range of features designed to provide you with the right resources and guidance for acing the Appian Certified Lead Developer (ACD-301) Exam

Best Appian ACD-301 Dumps PDF To Help You Out In Your Exam, Now the Appian Certified Lead Developer ACD-301 Exam Questions have become the first choice for instant and complete ACD-301 exam preparation.

These ACD-301 braindumps focus on the most significant portions of the Appian Certification Program certification that can be the part of the real ACD-301 exam.

- Actual ACD-301 Exam Prep 100% Valid Test Questions are The Best Products Easily obtain  ACD-301  for free download through  www.exam4labs.com Exam ACD-301 Flashcards
- Appian ACD-301 Authentic Exam Hub Reliable IT Certifications | ACD-301: Appian Certified Lead Developer Search for  ACD-301 on  www.pdfvce.com  immediately to obtain a free download ACD-301 Valid Test Bootcamp
- ACD-301 Latest Dumps Questions Reliable ACD-301 Braindumps Book Reliable ACD-301 Braindumps Book Immediately open  www.troytecdumps.com and search for  ACD-301  to obtain a free download Reliable ACD-301 Test Answers
- Actual ACD-301 Exam Prep 100% Valid Test Questions are The Best Products Copy URL  www.pdfvce.com  open and search for  ACD-301 to download for free  ACD-301 Practice Test Online
- Latest ACD-301 Test Answers ACD-301 Valid Test Bootcamp Dump ACD-301 File Search for ACD-301 and download exam materials for free through  www.examcollectionpass.com ACD-301 Latest Dumps Questions

