Free PDF 2026 Fantastic Workday Workday-Pro-Integrations Mock Exams

2026 Latest PrepPDF Workday-Pro-Integrations PDF Dumps and Workday-Pro-Integrations Exam Engine Free Share: https://drive.google.com/open?id=1ohQH5JXj8PU5g2Ki9XtKbyUz3G2IA5gJ

For candidates who need to practice the Workday-Pro-Integrations exam dumps for the exam, know the new changes of the exam center is quite necessary, it will provide you the references for the exam. We will provide you free update for 365 days after purchasing the product of us, so you will know the latest version of Workday-Pro-Integrations Exam Dumps. What's more, our system will send the latest version to your email box automatically. You just need to receive the version.

Workday Workday-Pro-Integrations Exam Syllabus Topics:

Topic	Details
Topic 1	• Enterprise Interface Builders: This section of the exam measures the skills of Integration Developers and covers the use of Workday's Enterprise Interface Builder (EIB) to design, deploy, and maintain inbound and outbound integrations. It evaluates the candidate's ability to create templates, configure transformation rules, schedule integrations, and troubleshoot EIB workflows efficiently.
Topic 2	• Calculated Fields: This section of the exam measures the skills of Workday Integration Analysts and covers the creation, configuration, and management of calculated fields used to transform, manipulate, and format data in Workday integrations. It evaluates understanding of field types, dependencies, and logical operations that enable dynamic data customization within integration workflows.
Topic 3	• XSLT: This section of the exam measures the skills of Data Integration Developers and covers the use of Extensible Stylesheet Language Transformations (XSLT) in Workday integrations. It focuses on transforming XML data structures, applying conditional logic, and formatting output for various integration use cases such as APIs and external file delivery.

>> Workday-Pro-Integrations Mock Exams <<

Workday-Pro-Integrations Instant Download - Workday-Pro-Integrations Exams

we guarantee to you that our Workday-Pro-Integrations study questions are of high quality and can help you pass the exam easily and successfully. Our Workday-Pro-Integrations exam questions boosts 99% passing rate and high hit rate so you needn't worry that you can't pass the exam. Our Workday-Pro-Integrations Exam Torrent is compiled by experts and approved by experienced professionals and updated according to the development situation in the theory and the practice. Our Workday-Pro-Integrations guide torrent can simulate the exam and boosts the timing function.

Workday Pro Integrations Certification Exam Sample Questions (Q31-Q36):

NEW QUESTION # 31
Refer to the following XML to answer the question below.
You are an integration developer and need to write XSLT to transform the output of an EIB which is making a request to the Get Job Profiles web service operation. The root template of your XSLT matches on the <wd:
Get_Job_Profiles_Response> element. This root template then applies templates against <wd:Job_Profile>.
What XPath syntax would be used to select the value of the ID element which has a wd:type attribute named Job_Profile_ID when the <xsl:value-of> element is placed within the template which matches on <wd:
Job_Profile>?

- A. wd:Job_Profile_Reference/wd:ID/wd:type='Job_Profile_ID'
- B. wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID']
- C. wd:Job_Profile_Reference/wd:ID/@wd:type='Job_Profile_ID'
- D. wd:Job_Profile_Reference/wd:ID/[@wd:type='Job_Profile_ID']

Answer: B

Explanation:
As an integration developer working with Workday, you are tasked with transforming the output of an Enterprise Interface Builder (EIB) that calls the Get_Job_Profiles web service operation. The provided XML shows the response from this operation, and you need to write XSLT to select the value of the <wd:ID> element where the wd:type attribute equals "Job_Profile_ID." The root template of your XSLT matches on
<wd:Get_Job_Profiles_Response> and applies templates to <wd:Job_Profile>. Within this template, you use the <xsl:value-of> element to extract the value. Let's analyze the XML structure, the requirement, and each option to determine the correct XPath syntax.
Understanding the XML and Requirement
The XML snippet provided is a SOAP response from the Get_Job_Profiles web service operation in Workday, using the namespace xmlns:wd="urn:com.workday/bsvc" and version wd:version="v43.0". Key elements relevant to the question include:
* The root element is <wd:Get_Job_Profiles_Response>.
* It contains <wd:Response_Data>, which includes <wd:Job_Profile> elements.
* Within <wd:Job_Profile>, there is <wd:Job_Profile_Reference>, which contains multiple <wd:ID> elements, each with a wd:type attribute:
* <wd:ID wd:type="WID">1740d3eca2f2ed9b6174ca7d2ae88c8c</wd:ID>
* <wd:ID wd:type="Job_Profile_ID">Senior_Benefits_Analyst</wd:ID>
The task is to select the value of the <wd:ID> element where wd:type="Job_Profile_ID" (e.g., "Senior_Benefits_Analyst") using XPath within an XSLT template that matches <wd:Job_Profile>. The <xsl:
value-of> element outputs the value of the selected node, so you need the correct XPath path from the <wd:
Job_Profile> context to the specific <wd:ID> element with the wd:type attribute value "Job_Profile_ID." Analysis of Options Let's evaluate each option based on the XML structure and XPath syntax rules:
* Option A: wd:Job_Profile_Reference/wd:ID/wd:type='Job_Profile_ID'
* This XPath attempts to navigate from wd:Job_Profile_Reference to wd:ID, then to wd:
type='Job_Profile_ID'. However, there are several issues:
* wd:type='Job_Profile_ID' is not valid XPath syntax. In XPath, to filter based on an attribute value, you use the attribute selector [@attribute='value'], not a direct comparison like wd:
type='Job_Profile_ID'.
* wd:type is an attribute of <wd:ID>, not a child element or node. This syntax would not select the <wd:ID> element itself but would be interpreted as trying to match a nonexistent child node or property, resulting in an error or no match.
* This option is incorrect because it misuses XPath syntax for attribute filtering.
* Option B: wd:Job_Profile_Reference/wd:ID/@wd:type='Job_Profile_ID'
* This XPath navigates to wd:Job_Profile_Reference/wd:ID and then selects the @wd:type attribute, comparing it to

"Job_Profile_ID" with =@wd:type='Job_Profile_ID'. However:
* The =@wd:type='Job_Profile_ID' syntax is invalid in XPath. To filter based on an attribute value, you use [@wd:type='Job_Profile_ID'] as a predicate, not an equality comparison in this form.
* This XPath would select the wd:type attribute itself (e.g., the string "Job_Profile_ID"), not the value of the <wd:ID> element. Since <xsl:value-of> expects a node or element value, selecting an attribute directly would not yield the desired "Senior_Benefits_Analyst" value.
* This option is incorrect due to the invalid syntax and inappropriate selection of the attribute instead of the element value.
* Option C: wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID']
* This XPath navigates from wd:Job_Profile_Reference to wd:ID and uses the predicate [@wd:
type='Job_Profile_ID'] to filter for <wd:ID> elements where the wd:type attribute equals
"Job_Profile_ID."
* In the XML, <wd:Job_Profile_Reference> contains:
* <wd:ID wd:type="WID">1740d3eca2f2ed9b6174ca7d2ae88c8c</wd:ID>
* <wd:ID wd:type="Job_Profile_ID">Senior_Benefits_Analyst</wd:ID>
* The predicate [@wd:type='Job_Profile_ID'] selects the second <wd:ID> element, whose value is "Senior_Benefits_Analyst."
* Since the template matches <wd:Job_Profile>, and <wd:Job_Profile_Reference> is a direct child of <wd:Job_Profile>, this path is correct:
* <wd:Job_Profile> # <wd:Job_Profile_Reference> # <wd:ID[@wd:
type='Job_Profile_ID']>.
* When used with <xsl:value-of select="wd:Job_Profile_Reference/wd:ID[@wd:
type='Job_Profile_ID']"/>, it outputs "Senior_Benefits_Analyst," fulfilling the requirement.
* This option is correct because it uses proper XPath syntax for attribute-based filtering and selects the desired <wd:ID> value.
* Option D: wd:Job_Profile_Reference/wd:ID/[@wd:type='Job_Profile_ID']
* This XPath is similar to Option C but includes an extra forward slash before the predicate: wd:ID/
[@wd:type='Job_Profile_ID']. In XPath, predicates like [@attribute='value'] are used directly after the node name (e.g., wd:ID[@wd:type='Job_Profile_ID']), not separated by a slash. The extra slash is syntactically incorrect and would result in an error or no match, as it implies navigating to a child node that doesn't exist.
* This option is incorrect due to the invalid syntax.
Why Option C is Correct
Option C, wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID'], is the correct XPath syntax because:
* It starts from the context node <wd:Job_Profile> (as the template matches this element) and navigates to <wd:Job_Profile_Reference/wd:ID>, using the predicate [@wd:type='Job_Profile_ID'] to filter for the <wd:ID> element with wd:type="Job_Profile_ID".
* It correctly selects the value "Senior_Benefits_Analyst," which is the content of the <wd:ID> element where wd:type="Job_Profile_ID".
* It uses standard XPath syntax for attribute-based filtering, aligning with Workday's XSLT implementation for web service responses.
* When used with <xsl:value-of>, it outputs the required value, fulfilling the question's requirement.
Practical Example in XSLT
Here's how this might look in your XSLT:
```
<xsl:template match="wd:Job_Profile">
<xsl:value-of select="wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID']"/>
</xsl:template>
```
This would output "Senior_Benefits_Analyst" for the <wd:ID> element with wd:type="Job_Profile_ID" in the XML.
Verification with Workday Documentation
The Workday Pro Integrations Study Guide and SOAP API Reference (available via Workday Community) detail the structure of the Get_Job_Profiles response and how to use XPath in XSLT for transformations. The XML structure shows <wd:Job_Profile_Reference> containing <wd:ID> elements with wd:type attributes, and the guide emphasizes using predicates like [@wd:type='value'] to filter based on attributes. This is a standard practice for navigating Workday web service responses.
Workday Pro Integrations Study Guide References
* Section: XSLT Transformations in EIBs - Describes using XSLT to transform web service responses, including selecting elements with XPath and attribute predicates.
* Section: Workday Web Services - Details the Get_Job_Profiles operation and its XML output structure, including <wd:Job_Profile_Reference> and <wd:ID> with wd:type attributes.
* Section: XPath Syntax - Explains how to use predicates like [@wd:type='Job_Profile_ID'] for attribute- based filtering in Workday XSLT.
* Workday Community SOAP API Reference - Provides examples of XPath navigation for Workday web service responses, including attribute selection.
Option C is the verified answer, as it correctly selects the <wd:ID> value with wd:type="Job_Profile_ID" using the appropriate XPath syntax within the <wd:Job_Profile> template context.

NEW QUESTION # 32
How do you initially upload the XSLT file to a Document Transformation integration system?

- A. From the Related Action on the Document Transformation, select Configure Integration Attachment Service.
- B. In the Global Workday Search bar, run the Edit Integration Service Attachment task.
- C. From the Related Action on the Document Transformation, select Configure Integration Attributes.
- D. In the Global Workday Search bar, run the Edit Integration Attachment Service task.

Answer: A

Explanation:
To upload an XSLT file to a Document Transformation integration system, you use the Configure Integration Attachment Service.
As per Workday documentation:
"The Configure Integration Attachment Service option on the Related Actions menu allows you to attach and manage XSLT files or other transformation documents used in Document Transformation integrations." This is the initial and correct method to upload the XSLT used for transforming incoming or outgoing XML.
Why the others are incorrect:
* B. Configure Integration Attributes configures integration behavior, not attachments.
* C and D reference invalid or misnamed tasks; they are not valid Workday tasks for XSLT upload.
Reference:Workday Pro: Document Transformation Integration Guide - "Uploading and managing XSLT via Configure Integration Attachment Service"

NEW QUESTION # 33
Refer to the following scenario to answer the question below. Your integration has the following runs in the integration events report (Date format of MM/DD/YYYY):
Run #1
* Core Connector: Worker Integration System was launched on May 15, 2024 at 3:00:00 AM.
* As of Entry Moment: 05/15/2024 3:00:00 AM
* Effective Date: 05/15/2024
* Last Successful As of Entry Moment: 05/01/2024 3:00:00 AM
* Last Successful Effective Date: 05/01/2024
Run #2
* Core Connector: Worker Integration System was launched on May 31, 2024 at 3:00:00 AM.
* As of Entry Moment: 05/31/2024 3:00:00 AM
* Effective Date: 05/31/2024
* Last Successful As of Entry Moment: 05/15/2024 3:00:00 AM
* Last Successful Effective Date: 05/15/2024 On May 13, 2024 Brian Hill receives a salary increase. The new salary amount is set to $90,000.00 with an effective date of April 30,2024. Which of these runs will include Brian Hill's compensation change?

- A. Brian Hill will be included in both integration runs.
- B. Brian Hill will be excluded from both integration runs.
- C. Brian Hill will only be included in the first integration run.
- D. Brian Hill will only be included in the second integration run.

Answer: B

Explanation:
The scenario involves a Core Connector: Worker integration with two runs detailed in the integration events report. The goal is to determine whether Brian Hill's compensation change, effective April 30, 2024, and entered on May 13, 2024, will be included in either of the runs based on their date launch parameters. Let's analyze each run against the change details to identify the correct answer.
In Workday, the Core Connector: Worker integration in incremental mode (as indicated by the presence of "Last Successful" parameters) processes changes based on the Transaction Log, filtering them by the Entry Moment (when the change was entered) and Effective Date (when the change takes effect). The integration captures changes where:
The Entry Moment falls between the Last Successful As of Entry Moment and the As of Entry Moment, and The Effective Date falls between the Last Successful Effective Date and the Effective Date.
Brian Hill's compensation change has:
Entry Moment: 05/13/2024 (time not specified, so we assume it occurs at some point during the day, before or up to 11:59:59 PM).
Effective Date: 04/30/2024.

Analysis of Run #1
Launch Date: 05/15/2024 at 3:00:00 AM
As of Entry Moment: 05/15/2024 3:00:00 AM - The latest point for when changes were entered.
Effective Date: 05/15/2024 - The latest effective date for changes.
Last Successful As of Entry Moment: 05/01/2024 3:00:00 AM - The starting point for entry moments.
Last Successful Effective Date: 05/01/2024 - The starting point for effective dates.
For Run #1 to include Brian's change:
The Entry Moment (05/13/2024) must be between 05/01/2024 3:00:00 AM and 05/15/2024 3:00:00 AM. Since 05/13/2024 falls within this range (assuming the change was entered before 3:00:00 AM on 05/15/2024, which is reasonable unless specified otherwise), this condition is met.
The Effective Date (04/30/2024) must be between 05/01/2024 (Last Successful Effective Date) and 05/15/2024 (Effective Date). However, 04/30/2024 is before 05/01/2024, so this condition is not met.
Since the effective date of Brian's change (04/30/2024) precedes the Last Successful Effective Date (05/01/2024), Run #1 will not include this change. In incremental mode, Workday excludes changes with effective dates prior to the last successful effective date, as those are assumed to have been processed in a prior run (before Run #1's baseline of 05/01/2024).
Analysis of Run #2
Launch Date: 05/31/2024 at 3:00:00 AM
As of Entry Moment: 05/31/2024 3:00:00 AM - The latest point for when changes were entered.
Effective Date: 05/31/2024 - The latest effective date for changes.
Last Successful As of Entry Moment: 05/15/2024 3:00:00 AM - The starting point for entry moments.
Last Successful Effective Date: 05/15/2024 - The starting point for effective dates.
For Run #2 to include Brian's change:
The Entry Moment (05/13/2024) must be between 05/15/2024 3:00:00 AM and 05/31/2024 3:00:00 AM. However, 05/13/2024 is before 05/15/2024 3:00:00 AM, so this condition is not met.
The Effective Date (04/30/2024) must be between 05/15/2024 (Last Successful Effective Date) and 05/31/2024 (Effective Date). Since 04/30/2024 is before 05/15/2024, this condition is also not met.
In Run #2, the Entry Moment (05/13/2024) precedes the Last Successful As of Entry Moment (05/15/2024 3:00:00 AM), meaning the change was entered before the starting point of this run's detection window. Additionally, the Effective Date (04/30/2024) is well before the Last Successful Effective Date (05/15/2024). Both filters exclude Brian's change from Run #2.
Conclusion
Run #1: Excluded because the effective date (04/30/2024) is before the Last Successful Effective Date (05/01/2024).
Run #2: Excluded because the entry moment (05/13/2024) is before the Last Successful As of Entry Moment (05/15/2024 3:00:00 AM) and the effective date (04/30/2024) is before the Last Successful Effective Date (05/15/2024).
Brian Hill's change would have been processed in an earlier run (prior to May 1, 2024) if the integration was running incrementally before Run #1, as its effective date (04/30/2024) predates both runs' baselines. Given the parameters provided, neither Run #1 nor Run #2 captures this change, making D. Brian Hill will be excluded from both integration runs the correct answer.
Workday Pro Integrations Study Guide Reference
Workday Integrations Study Guide: Core Connector: Worker - Section on "Incremental Processing" explains how changes are filtered based on entry moments and effective dates relative to the last successful run.
Workday Integrations Study Guide: Launch Parameters - Details how "Last Successful As of Entry Moment" and "Last Successful Effective Date" define the starting point for detecting new changes, excluding prior transactions.
Workday Integrations Study Guide: Change Detection - Notes that changes with effective dates before the last successful effective date are assumed processed in earlier runs and are skipped in incremental mode.

NEW QUESTION # 34
Refer to the following XML to answer the question below.
Within the template which matches on wd:Report_Entry, you would like to conditionally process the wd:Education_Group elements by using an <xsl:apply-templates> element. What XPath syntax would be used for the select to iterate over only the wd:Education_Group elements where the Degree is an MBA?

- A. wd:Education_Group/wd:Degree='MBA'
- B. wd:Report_Entry/wd:Education_Group/ wd:Degree='MBA' 1:Degree='MBA'
- C. wd:Report_Entry/wd:Education_Group[wd:Degree='MBA' 1:Degree='MBA']
- D. wd:Education_Group[wd:Degree='MBA']

Answer: D

Explanation:
In Workday integrations, XSLT is used to transform XML data, such as the output from a web service-enabled report or EIB, into

a desired format for third-party systems. In this scenario, you need to write XSLT to process wd:Education_Group elements within a template matching wd:Report_Entry, using an <xsl:apply-templates> element to iterate only over wd:Education_Group elements where the wd:Degree is "MBA." The correct XPath syntax for the select attribute is critical to ensure accurate filtering.
Here's why option A is correct:
XPath Syntax In XPath, square brackets [ ] are used to specify predicates or conditions to filter elements. The condition wd:Degree='MBA' checks if the wd:Degree child element has the value "MBA." When applied to wd:Education_Group, the expression wd:Education_Group[wd:Degree='MBA'] selects only those wd:Education_Group elements that contain a wd:Degree child element with the value "MBA." Context in XSLT: Within an <xsl:apply-templates> element in a template matching wd:Report_Entry, the select attribute uses XPath to specify which nodes to process. This syntax ensures that the template only applies to wd:Education_Group elements where the degree is "MBA," aligning with the requirement to conditionally process only those specific education groups.
XML Structure Alignment: Based on the provided XML snippet, wd:Education_Group contains wd:Education and wd:Degree child elements (e.g., <wd:Degree>MBA</wd:Degree>). The XPath wd:Education_Group[wd:Degree='MBA'] correctly navigates to wd:Education_Group and filters based on the wd:Degree value, matching the structure and requirement.
Why not the other options?
B . wd:Education_Group/wd:Degree='MBA': This is not a valid XPath expression for a predicate. It attempts to navigate to wd:Degree as a child but does not use square brackets [ ] to create a filtering condition. This would be interpreted as selecting wd:Degree elements under wd:Education_Group, but it wouldn't filter based on the value "MBA" correctly within an <xsl:apply-templates> context.
C . wd:Report_Entry/wd:Education_Group/wd:Degree='MBA' 1:Degree='MBA': This is syntactically incorrect and unclear. It includes a malformed condition (1:Degree='MBA') and does not use proper XPath predicate syntax. It fails to filter wd:Education_Group elements based on wd:Degree='MBA' and is not valid for use in select.
D . wd:Report_Entry/wd:Education_Group[wd:Degree='MBA' 1:Degree='MBA']: This is also syntactically incorrect due to the inclusion of 1:Degree='MBA' within the predicate. The 1: prefix is not valid XPath syntax and introduces an error. The correct predicate should only be wd:Degree='MBA' to filter the wd:Education_Group elements.
To implement this in XSLT:
Within your template matching wd:Report_Entry, you would write an <xsl:apply-templates> element with the select attribute set to wd:Education_Group[wd:Degree='MBA']. This ensures that only wd:Education_Group elements with a wd:Degree value of "MBA" are processed by the corresponding templates, effectively filtering out other degrees (e.g., B.S., B.A.) in the transformation.
This approach ensures the XSLT transformation aligns with Workday's XML structure and integration requirements for processing education data in a report output.
:
Workday Pro Integrations Study Guide: Section on "XSLT Transformations for Workday Integrations" - Details the use of XPath in XSLT for filtering XML elements, including predicates for conditional processing based on child element values.
Workday EIB and Web Services Guide: Chapter on "XML and XSLT for Report Data" - Explains the structure of Workday XML (e.g., wd:Education_Group, wd:Degree) and how to use XPath to navigate and filter data.
Workday Reporting and Analytics Guide: Section on "Web Service-Enabled Reports" - Covers integrating report outputs with XSLT for transformations, including examples of filtering elements based on specific values like degree types.

NEW QUESTION # 35
Refer to the following scenario to answer the question below. Your integration has the following runs in the integration events report (Date format of MM/DD/YYYY):
Run #1
* Core Connector: Worker Integration System was launched on May 15, 2024 at 3:00:00 AM.
* As of Entry Moment: 05/15/2024 3:00:00 AM
* Effective Date: 05/15/2024
* Last Successful As of Entry Moment: 05/01/2024 3:00:00 AM
* Last Successful Effective Date: 05/01/2024
Run #2
* Core Connector: Worker Integration System was launched on May 31, 2024 at 3:00:00 AM.
* As of Entry Moment: 05/31/2024 3:00:00 AM
* Effective Date: 05/31/2024
* Last Successful As of Entry Moment: 05/15/2024 3:00:00 AM
* Last Successful Effective Date: 05/15/2024 On May 13, 2024 Brian Hill receives a salary increase. The new salary amount is set to $90,000.00 with an effective date of April 30,2024. Which of these runs will include Brian Hill's compensation change?

- A. Brian Hill will be included in both integration runs.
- B. Brian Hill will be excluded from both integration runs.
- C. Brian Hill will only be included in the first integration run.

- D. Brian Hill will only be included in the second integration run.

Answer: B

Explanation:
The scenario involves a Core Connector: Worker integration with two runs detailed in the integration events report. The goal is to determine whether Brian Hill's compensation change, effective April 30, 2024, and entered on May 13, 2024, will be included in either of the runs based on their date launch parameters. Let's analyze each run against the change details to identify the correct answer.
In Workday, the Core Connector: Worker integration in incremental mode (as indicated by the presence of "Last Successful" parameters) processes changes based on the Transaction Log, filtering them by theEntry Moment(when the change was entered) andEffective Date(when the change takes effect). The integration captures changes where:
* TheEntry Momentfalls between theLast Successful As of Entry Momentand theAs of Entry Moment, and
* TheEffective Datefalls between theLast Successful Effective Dateand theEffective Date.
Brian Hill's compensation change has:
* Entry Moment:05/13/2024 (time not specified, so we assume it occurs at some point during the day, before or up to 11:59:59 PM).
* Effective Date:04/30/2024.
Analysis of Run #1
* Launch Date:05/15/2024 at 3:00:00 AM
* As of Entry Moment:05/15/2024 3:00:00 AM - The latest point for when changes were entered.
* Effective Date:05/15/2024 - The latest effective date for changes.
* Last Successful As of Entry Moment:05/01/2024 3:00:00 AM - The starting point for entry moments.
* Last Successful Effective Date:05/01/2024 - The starting point for effective dates.
For Run #1 to include Brian's change:
* TheEntry Moment(05/13/2024) must be between 05/01/2024 3:00:00 AM and 05/15/2024 3:00:00 AM. Since 05/13/2024 falls within this range (assuming the change was entered before 3:00:00 AM on
05/15/2024, which is reasonable unless specified otherwise), this condition is met.
* TheEffective Date(04/30/2024) must be between 05/01/2024 (Last Successful Effective Date) and 05
/15/2024 (Effective Date). However, 04/30/2024 isbefore05/01/2024, so this condition isnot met.
Since the effective date of Brian's change (04/30/2024) precedes theLast Successful Effective Date(05/01
/2024), Run #1 will not include this change. In incremental mode, Workday excludes changes with effective dates prior to the last successful effective date, as those are assumed to have been processed in a prior run (before Run #1's baseline of 05/01/2024).
Analysis of Run #2
* Launch Date:05/31/2024 at 3:00:00 AM
* As of Entry Moment:05/31/2024 3:00:00 AM - The latest point for when changes were entered.
* Effective Date:05/31/2024 - The latest effective date for changes.
* Last Successful As of Entry Moment:05/15/2024 3:00:00 AM - The starting point for entry moments.
* Last Successful Effective Date:05/15/2024 - The starting point for effective dates.
For Run #2 to include Brian's change:
* TheEntry Moment(05/13/2024) must be between 05/15/2024 3:00:00 AM and 05/31/2024 3:00:00 AM. However, 05/13/2024 isbefore05/15/2024 3:00:00 AM, so this condition isnot met.
* TheEffective Date(04/30/2024) must be between 05/15/2024 (Last Successful Effective Date) and 05
/31/2024 (Effective Date). Since 04/30/2024 isbefore05/15/2024, this condition is alsonot met.
In Run #2, theEntry Moment(05/13/2024) precedes theLast Successful As of Entry Moment(05/15/2024 3:
00:00 AM), meaning the change was entered before the starting point of this run's detection window.
Additionally, theEffective Date(04/30/2024) is well before theLast Successful Effective Date(05/15/2024).
Both filters exclude Brian's change from Run #2.
Conclusion
* Run #1:Excluded because the effective date (04/30/2024) is before the Last Successful Effective Date (05/01/2024).
* Run #2:Excluded because the entry moment (05/13/2024) is before the Last Successful As of Entry Moment (05/15/2024 3:00:00 AM) and the effective date (04/30/2024) is before the Last Successful Effective Date (05/15/2024).
Brian Hill's change would have been processed in an earlier run (prior to May 1, 2024) if the integration was running incrementally before Run #1, as its effective date (04/30/2024) predates both runs' baselines. Given the parameters provided, neither Run #1 nor Run #2 captures this change, makingD. Brian Hill will be excluded from both integration runsthe correct answer.
Workday Pro Integrations Study Guide References
* Workday Integrations Study Guide: Core Connector: Worker- Section on "Incremental Processing" explains how changes are filtered based on entry moments and effective dates relative to the last successful run.
* Workday Integrations Study Guide: Launch Parameters- Details how "Last Successful As of Entry Moment" and "Last Successful Effective Date" define the starting point for detecting new changes, excluding prior transactions.
* Workday Integrations Study Guide: Change Detection- Notes that changes with effective dates before the last successful effective

date are assumed processed in earlier runs and are skipped in incremental mode.

NEW QUESTION # 36
......

The marketplace is competitive, especially for securing a well-paid job. Moving your career one step ahead with Workday-Pro-Integrations certification will be a necessary and important thing. How to get the Workday-Pro-Integrations exam dumps with 100% pass is also important. Workday-Pro-Integrations training topics will ensure you pass at first time. The experts who involved in the edition of Workday-Pro-Integrations questions & answers all have rich hands-on experience, which guarantee you the high quality and high pass rate.