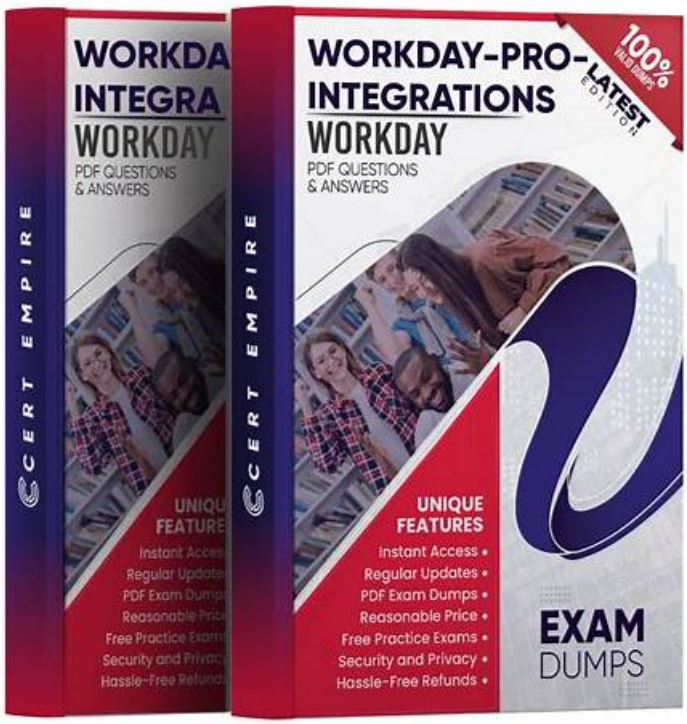
Latest Workday-Pro-Integrations Exam Camp, Workday-Pro-Integrations Free Study Material



2026 Latest Exams4sures Workday-Pro-Integrations PDF Dumps and Workday-Pro-Integrations Exam Engine Free Share: https://drive.google.com/open?id=1KuZMwCzxN2lirQ5iimcsRAK0yjTlUK4G

If you want to demonstrate your expertise in solving complex Workday real-life problems, then you need to pass the Workday Workday-Pro-Integrations certification exam. However, passing this exam is not an easy task. It requires you to master complicated subjects related to Workday Pro Integrations Certification Exam. To help you prepare for this exam, Exams4sures offers verified Workday Workday-Pro-Integrations Exam Questions that are ruling the preparation world.

Workday Workday-Pro-Integrations Exam Syllabus Topics:

Topic	Details
Topic 1	• Cloud Connect: This section of the exam measures the skills of Workday Implementation Consultants and focuses on using Workday Cloud Connect solutions for third-party integration. It includes understanding pre-built connectors, configuration settings, and how to manage data flow between Workday and external systems while ensuring security and data integrity.
Topic 2	• Reporting: This section of the exam measures the skills of Reporting Analysts and focuses on building, modifying, and managing Workday reports that support integrations. It includes working with report writer tools, custom report types, calculated fields within reports, and optimizing report performance to support automated data exchange.
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.

Topic 4	• Integrations: This section of the exam measures the skills of Integration Specialists and covers the full spectrum of integration techniques in Workday. It includes an understanding of core integration architecture, APIs, Workday Studio, and integration system user setup. The focus is on building scalable, maintainable, and secure integrations that ensure seamless system interoperability.
Topic 5	• 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.

2026 Latest Workday-Pro-Integrations Exam Camp & First-grade Workday Workday-Pro-Integrations Free Study Material 100% Pass

Workday Workday-Pro-Integrations exam certification is widely recognized IT certifications. People around the world prefer Workday-Pro-Integrations exam certification to make their careers more strengthened and successful. Speaking of Workday Workday-Pro-Integrations exam, Exams4sures Workday Workday-Pro-Integrations exam training materials have been ahead of other sites. Because Exams4sures has a strong IT elite team, they always follow the latest Workday Workday-Pro-Integrations Exam Training materials, with their professional mind to focus on Workday Workday-Pro-Integrations exam training materials.

Workday Pro Integrations Certification Exam Sample Questions (Q70-Q75):

NEW QUESTION # 70
A calculated field used as a field override in a Connector is not appearing in the output. Assuming the field has a value, what could cause this to occur?

- A. Access not provided to all instances of calculated field.
- B. Access not provided to all fields in the calculated field.
- C. Access not provided to Connector calculated field web service.
- D. Access not provided to calculated field data source.

Answer: B

Explanation:
This question addresses a troubleshooting scenario in Workday Pro Integrations, where a calculated field used as a field override in a Connector does not appear in the output, despite having a value. Let's analyze the potential causes and evaluate each option.
Understanding Calculated Fields and Connectors in Workday
Calculated Fields: In Workday, calculated fields are custom fields created using Workday's expression language to derive values based on other fields, conditions, or functions. They are often used in reports, integrations, and business processes to transform or aggregate data. Calculated fields can reference other fields (data sources) and require appropriate security permissions to access those underlying fields.
Field Override in Connectors: In a Core Connector or other integration system, a field override allows you to replace or supplement a default field with a custom value, such as a calculated field. This is configured in the integration's mapping or transformation steps, ensuring the output includes the desired data. However, for the calculated field to appear in the output, it must be accessible, have a valid value, and be properly configured in the integration.
Issue: Calculated Field Not Appearing in Output: If the calculated field has a value but doesn't appear in the Connector's output, the issue likely relates to security, configuration, or access restrictions. The question assumes the field has a value, so we focus on permissions or setup errors rather than data issues.
Evaluating Each Option
Let's assess each option based on Workday's integration and security model:
Option A: Access not provided to calculated field data source.
Analysis: This is partially related but incorrect as the primary cause. Calculated fields often rely on underlying data sources (e.g., worker data, organization data) to compute their values. If access to the data source is restricted, the calculated field might not compute correctly or appear in the output. However, the question specifies the field has a value, implying the data source is accessible. The more specific issue is likely access to the individual fields within the calculated field's expression, not just the broader data source.

Why It Doesn't Fit: While data source access is important, it's too general here. The calculated field's value exists, suggesting the data source is accessible, but the problem lies in finer-grained permissions for the fields used in the calculation.
Option B: Access not provided to all fields in the calculated field.
Analysis: This is correct. Calculated fields in Workday are expressions that reference one or more fields (e.g., Worker_ID + Position_Title). For the calculated field to be used in a Connector's output, the ISU (via its ISSG) must have access to all fields referenced in the calculation. If any field lacks "Get" or "View" permission in the relevant domain (e.g., Worker Data), the calculated field won't appear in the output, even if it has a value. This is a common security issue in integrations, as ISSGs must be configured with domain access for every field involved.
Why It Fits: Workday's security model requires granular permissions. For example, if a calculated field combines Worker_Name and Hire_Date, the ISU needs access to both fields' domains. If Hire_Date is restricted, the calculated field fails to output, even with a value. This aligns with the scenario and is a frequent troubleshooting point in Workday Pro Integrations.
Option C: Access not provided to Connector calculated field web service.
Analysis: This is incorrect. There isn't a specific "Connector calculated field web service" in Workday. Calculated fields are part of the integration's configuration, not a separate web service. The web service operation used by the Connector (e.g., Get_Workers) must have permissions, but this relates to the overall integration, not the calculated field specifically. The issue here is field-level access, not a web service restriction.
Why It Doesn't Fit: This option misinterprets Workday's architecture. Calculated fields are configured within the integration, not as standalone web services, making this irrelevant to the problem.
Option D: Access not provided to all instances of calculated field.
Analysis: This is incorrect. The concept of "instances" typically applies to data records (e.g., all worker records), not calculated fields themselves. Calculated fields are expressions, not data instances, so there's no need for "instance-level" access. The issue is about field-level permissions within the calculated field's expression, not instances of the field. This option misunderstands Workday's security model for calculated fields.
Why It Doesn't Fit: Calculated fields don't have "instances" requiring separate access; they depend on the fields they reference, making this option inaccurate.
Final Verification
The correct answer is Option B, as the calculated field's absence in the output is likely due to the ISU lacking access to all fields referenced in the calculated field's expression. For example, if the calculated field in a Core Connector: Worker Data combines Worker_ID and Department_Name, the ISSG must have "Get" access to both the Worker Data and Organization Data domains. If Department_Name is restricted, the calculated field won't output, even with a value. This is a common security configuration issue in Workday integrations, addressed by reviewing and adjusting ISSG domain permissions.
This aligns with Workday's security model, where granular permissions are required for all data elements, as seen in Questions 26 and 28. The assumption that the field has a value rules out data or configuration errors, focusing on security as the cause.
Supporting Documentation
The reasoning is based on:
Workday Community documentation on calculated fields, security domains, and integration mappings.
Tutorials on configuring Connectors and troubleshooting, such as Workday Advanced Studio Tutorial, highlighting field access issues.
Integration security guides from partners (e.g., NetIQ, Microsoft Learn, Reco.ai) detailing ISSG permissions for fields in calculated expressions.
Community discussions on Reddit and Workday forums on calculated field troubleshooting (r/workday on Reddit).

NEW QUESTION # 71
Refer to the following XML to answer the question below.
You are an integration developer and need to write X8LT to transform the output of an ElB which is using a web service enabled report to output position data along with hiring restrictions around skills. You currently have a template which matches on wd:Report Data/wd: Report .Entry for creating a record from each report entry.
Within the template which matches on wd:Report_Entry you would like to conditionally process the wd:
Job_Skills element by using a series of <xsl:if> elements so as to categorize the job skills data.
Assuming all jobs will have the wd:Job_Skills element, what XSLT syntax would be used to output the text HR Skills if the value of wd:Job_Skills contains the text HR and output NON-HR Skills if the value of wd:
Job_Skills does not contain the text HR?

- A. B.
- B. C.
- C. D.

- D.

Answer: C

Explanation:
The task is to write XSLT within a template matching wd:Report_Data/wd:Report_Entry to categorize wd:
Job_Skills data, outputting "HR Skills" if the value contains "HR" and "NON-HR Skills" if it does not, using a series of <xsl:if> elements. The correct syntax must use the contains() function to check for the substring
"HR" within wd:Job_Skills, as the question implies partial matching (e.g., "HR Specialist" or "Senior HR"), not exact equality.
Let's analyze each option:
* Option A:
xml
<job_skill>
<xsl:value-of select="wd:Hiring_Restrictions/wd:Job_Skills='HR'">
<xsl:text>HR Skills</xsl:text>
<xsl:if/>
<xsl:value-of select="not(wd:Hiring_Restrictions/wd:Job_Skills='HR')">
<xsl:text>NON-HR Skills</xsl:text>
<xsl:if/>
</job_skill>
* Issues:
* <xsl:value-of> is misused here. It outputs the result of the expression (e.g., "true" or "false" for a comparison), not the conditional text. The <xsl:text> inside won't execute as intended.
* The = operator checks for exact equality (e.g., wd:Job_Skills must be exactly "HR"), not substring presence, which contradicts the requirement to check if "HR" is contained within the value.
* <xsl:if/> is malformed (self-closing without a test attribute) and misplaced.
* Verdict: Incorrect syntax and logic.
* Option B:
xml
<job_skill>
<xsl:value-of select="contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR')">
<xsl:text>HR Skills</xsl:text>
<xsl:if/>
<xsl:value-of select="not(contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR'))">
<xsl:text>NON-HR Skills</xsl:text>
<xsl:if/>
</job_skill>
* Issues:
* Similar to A, <xsl:value-of> outputs the boolean result of contains() ("true" or "false"), not the conditional text "HR Skills" or "NON-HR Skills."
* The <xsl:text> elements are inside invalid <xsl:if/> tags (self-closing, no test), rendering them ineffective.
* While contains() is correct for substring checking, the structure fails to meet the <xsl:if> requirement.
* Verdict: Incorrect structure despite using contains().
* Option C:
xml
<job_skill>
<xsl:if test="wd:Hiring_Restrictions/wd:Job_Skills='HR'">
<xsl:text>HR Skills</xsl:text>
</xsl:if>
<xsl:if test="not(wd:Hiring_Restrictions/wd:Job_Skills='HR')">
<xsl:text>NON-HR Skills</xsl:text>
</xsl:if>
</job_skill>
* Analysis:
* Uses <xsl:if> correctly with test attributes, satisfying the "series of <xsl:if> elements" requirement.
* However, wd:Job_Skills='HR' tests for exact equality, not whether "HR" is contained within the value. For example, "HR Specialist" would fail this test, outputting "NON-HR Skills" incorrectly.
* Verdict: Semantically incorrect due to exact matching instead of substring checking.
* Option D:
xml
<job_skill>

<xsl:if test="contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR')">
<xsl:text>HR Skills</xsl:text>
</xsl:if>
<xsl:if test="not(contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR'))">
<xsl:text>NON-HR Skills</xsl:text>
</xsl:if>
</job_skill>
* Analysis:
* Correctly uses <xsl:if> with test attributes, aligning with the question's requirement.
* The contains() function properly checks if "HR" is a substring within wd:Job_Skills (e.g., "HR Manager" or "Senior HR" returns true).
* not(contains()) ensures the opposite condition, covering all cases (mutually exclusive).
* <xsl:text> outputs the exact strings "HR Skills" or "NON-HR Skills" as required.
* Note: The closing tag </xs1:if> is a typo in the option (should be </xsl:if>), but in context, it's an obvious formatting error, not a substantive issue.
* Verdict: Correct logic and syntax, making D the best answer.
Correct Implementation in Context:
xml
<xsl:template match="wd:Report_Data/wd:Report_Entry">
<job_skill>
<xsl:if test="contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR')">
<xsl:text>HR Skills</xsl:text>
</xsl:if>
<xsl:if test="not(contains(wd:Hiring_Restrictions/wd:Job_Skills, 'HR'))">
<xsl:text>NON-HR Skills</xsl:text>
</xsl:if>
</job_skill>
</xsl:template>
* Example Input: <wd:Job_Skills>Senior HR Analyst</wd:Job_Skills> # Output: <job_skill>HR Skills</job_skill>
* Example Input: <wd:Job_Skills>IT Specialist</wd:Job_Skills> # Output: <job_skill>NON-HR Skills</job_skill>
Workday Pro Integrations Study Guide: "Configure Integration System - TRANSFORMATION" section, detailing <xsl:if> and contains() for conditional XSLT logic in Workday.
Workday Documentation: "XSLT Transformations in Workday" under EIB, confirming wd: namespace usage and string functions.
W3C XSLT 1.0 Specification: Section 9.1, "Conditional Processing with <xsl:if>," and Section 11.2, "String Functions" (contains()).
Workday Community: Examples of substring-based conditionals in XSLT for report transformations.

NEW QUESTION # 72
Refer to the following XML to answer the question below.
You need the integration file to format the ps:PositionJD field to 10 characters and report any truncated values as an error.
How will you start your template match on ps:Position to use Document Transformation (DT) to do the transformation using ETV with your truncation validation?

- A.
- B.
- C.
- D.

Answer: A

Explanation:
In Workday integrations, Document Transformation (DT) using XSLT is employed to transform XML data, such as the output from a Core Connector or EIB, into a specific format for third-party systems. In this scenario, you need to transform the ps:Position_ID field within the ps:Position element to a fixed length of 10 characters and report any truncation as an error using Workday's Extension for Transformation and Validation (ETV) attributes. The template must match the ps:Position element and apply the specified formatting and validation rules.
Here's why option D is correct:
Template Matching: The <xsl:template match="ps:Position"> correctly targets the ps:Position element in the XML, as shown in the provided snippet, ensuring the transformation applies to the appropriate node.

ETV Attributes:
etv:fixedLength="10" specifies that the Pos_ID field should be formatted to a fixed length of 10 characters. This ensures the output is truncated or padded (if needed) to meet the length requirement.
etv:reportTruncation="error" instructs the transformation to raise an error if the ps:Position_ID value exceeds 10 characters and cannot be truncated without data loss, aligning with the requirement to report truncated values as errors.
XPath Selection: The <xsl:value-of select="ps:Position_Data/ps:Position_ID"/> correctly extracts the ps:Position_ID value from the ps:Position_Data child element, as shown in the XML structure (<ps:Position_ID>P-00030</ps:Position_ID>).
Output Structure: The <Position><Pos_ID>...</Pos_ID></Position> structure ensures the transformed data is wrapped in meaningful tags for the target system, maintaining consistency with Workday integration practices.
Why not the other options?
A .
xml
WrapCopy
<xsl:template match="ps:Position">
<Position>
<Pos_ID etv:fixedLength="10">
<xsl:value-of select="ps:Position_Data/ps:Position_ID"/>
</Pos_ID>
</Position>
</xsl:template>
This option includes etv:fixedLength="10" but omits etv:reportTruncation="error". Without the truncation reporting, it does not meet the requirement to report truncated values as errors, making it incorrect.
B .
xml
WrapCopy
<xsl:template match="ps:Position">
<Position etv:fixedLength="10">
<Pos_ID etv:reportTruncation="error">
<xsl:value-of select="ps:Position_Data/ps:Position_ID"/>
</Pos_ID>
</Position>
</xsl:template>
This applies etv:fixedLength="10" to the Position element instead of Pos_ID, and etv:reportTruncation="error" to Pos_ID. However, ETV attributes like fixedLength and reportTruncation should be applied to the specific field being formatted (Pos_ID), not the parent element (Position). This misplacement makes it incorrect.
C .
xml
WrapCopy
<xsl:template match="ps:Position">
<Position etv:fixedLength="10">
<Pos_ID etv:reportTruncation="error">
<xsl:value-of select="ps:Position_Data/ps:Position_ID"/>
</Pos_ID>
</Position>
</xsl:template>
Similar to option B, this applies etv:fixedLength="10" to Position and etv:reportTruncation="error" to Pos_ID, which is incorrect for the same reason: ETV attributes must be applied to the specific field (Pos_ID) requiring formatting and validation, not the parent element.
To implement this in XSLT for a Workday integration:
Use the template from option D to match ps:Position, apply etv:fixedLength="10" and etv:reportTruncation="error" to the Pos_ID element, and extract the ps:Position_ID value using the correct XPath. This ensures the ps:Position_ID (e.g., "P-00030") is formatted to 10 characters and reports any truncation as an error, meeting the integration file requirements.
:
Workday Pro Integrations Study Guide: Section on "Document Transformation (DT) and ETV" - Details the use of ETV attributes like fixedLength and reportTruncation for formatting and validating data in XSLT transformations.
Workday Core Connector and EIB Guide: Chapter on "XML Transformations" - Explains how to use XSLT templates to transform position data, including ETV attributes for length and truncation validation.
Workday Integration System Fundamentals: Section on "ETV in Integrations" - Covers the application of ETV attributes to specific fields in XML for integration outputs, ensuring compliance with formatting and error-reporting requirements.

NEW QUESTION # 73
Refer to the following scenario to answer the question below.
You have been asked to build an integration using the Core Connector: Worker template and should leverage the Data Initialization Service (DIS). The integration will be used to export a full file (no change detection) for employees only and will include personal data. The vendor receiving the file requires marital status values to be sent using a list of codes that they have provided instead of the text values that Workday uses internally and if a text value in Workday does not align with the vendors list of codes the integration should report "OTHER".
What configuration is required to output the list of codes required from by the vendor instead of Workday's values in this integration?

- A. Configure Integration Attributes with a blank Default
- B. Configure Integration Attributes with "OTHER" as a Default
- C. Configure Integration Maps with "OTHER" as a Default
- D. Configure Integration Maps with a blank Default

Answer: C

Explanation:
The scenario involves a Core Connector: Worker integration using the Data Initialization Service (DIS) to export a full file of employee personal data. The vendor requires marital status values to be transformed from Workday's internal text values (e.g., "Married," "Single") to a specific list of codes (e.g., "M," "S"), and any Workday value not matching the vendor's list should output "OTHER." Let's analyze the configuration:
Requirement:Transform the "Marital Status" field values into vendor-specific codes, with a fallback to "OTHER" for unmapped values. This is a field-level transformation, common in Core Connectors when aligning Workday data with external system requirements.
Integration Maps:In Core Connectors, Integration Maps are the primary tool for transforming field values. You create a map that defines source values (Workday's marital status text) and target values (vendor's codes). The "Default" setting in an integration map specifies what value to output if a Workday value isn't explicitly mapped. Here, setting the default to "OTHER" ensures that any marital status not in the vendor's list (e.g., a new Workday value like "Civil Union" not recognized by the vendor) is output as "OTHER." Option Analysis:
A . Configure Integration Maps with a blank Default: Incorrect. A blank default would leave the field empty or pass the original Workday value for unmapped cases, not "OTHER," failing the requirement.
B . Configure Integration Attributes with a blank Default: Incorrect. Integration Attributes define integration-level settings (e.g., file name, delivery method), not field value transformations. They don't support mapping or defaults for specific fields like marital status.
C . Configure Integration Maps with "OTHER" as a Default: Correct. This uses Integration Maps to map Workday values to vendor codes and sets "OTHER" as the default for unmapped values, meeting the requirement fully.
D . Configure Integration Attributes with "OTHER" as a Default: Incorrect. Integration Attributes don't handle field-level transformations or defaults for data values, making this option inapplicable.
Implementation:
Edit the Core Connector: Worker integration.
Use the related action Configure Integration Maps.
Create a map for the "Marital Status" field (e.g., "Married" → "M," "Single" → "S").
Set the Default Value to "OTHER" in the map configuration.
Test the output to ensure mapped values use vendor codes and unmapped values return "OTHER." Reference from Workday Pro Integrations Study Guide:
Core Connectors & Document Transformation: Section on "Configuring Integration Maps" explains mapping field values and using defaults for unmapped cases.
Integration System Fundamentals: Highlights how Core Connectors transform data to meet vendor specifications.

NEW QUESTION # 74
Refer to the following XML to answer the question below.
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>.
XPath contains a number of delivered functions such as format-date. The format-date function uses the following syntax: format-date ($value as xs: date? $picture as xs:string). Within the template which matches on <wd:Job_Profile>, what XPath syntax would you use to output the value of the <wd:Effective_Date> element formatted with the day-month-year format of "15-07-2024"?

- A. format-date (wd:Job_Profile_Data/wd:Effective_Date, '[M01]-[D01]-[Y0001]')
- B. format-date (wd:Job_Profile_Data/wd:Effective_Date, '[D01]-[M01]-[Y0001]')

- C. format-date('[D01]-[M01]-[Y0001]', wd:Job_Profile_Data/wd:Effective_Date)
- D. format-date('[M01]-[D01]-[Y0001]', wd:Job_Profile_Data/wd:Effective_Date)

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 XML provided shows the response from this operation, and you need to write XSLT to format the <wd:Effective_Date> element within the <wd:Job_Profile_Data> section. Specifically, you need to output the date "2024-05-15" (as seen in the XML) in the format "15-07-2024" (day-month-year). The root template of your XSLT matches on
<wd:Get_Job_Profiles_Response> and applies templates to <wd:Job_Profile>. You are using the format-date XPath function, which follows the syntax: format-date($value as xs:date?, $picture as xs:string). Let's analyze the XML, the requirement, and each option to determine the correct XPath syntax.
Understanding the XML and Requirement
The provided XML snippet shows a response from the Get_Job_Profiles web service operation in Workday, formatted in SOAP XML with the Workday namespace (xmlns:wd="urn:com.workday/bsvc"). 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_Data>, which contains <wd:Effective_Date> with the value 2024-05-15.
* You need to transform this date into the format "15-07-2024" (DD-MM-YYYY), where:
* "15" is the day (D01 for two digits).
* "07" is the month (M01 for two digits, noting the XML shows May, but the question specifies July for the output format-likely a hypothetical or test case adjustment).
* "2024" is the year (Y0001 for four digits).
The format-date function in XPath 2.0 (used by Workday) formats a date value according to a picture string.
The syntax is:
* First parameter: The date value (e.g., wd:Job_Profile_Data/wd:Effective_Date), which must be an xs:
date or convertible to one.
* Second parameter: The picture string (e.g., '[D01]-[M01]-[Y0001]'), specifying the format using patterns like:
* [D01] for two-digit day (01-31).
* [M01] for two-digit month (01-12).
* [Y0001] for four-digit year (e.g., 2024).
The question specifies that the root template matches <wd:Get_Job_Profiles_Response> and applies templates to <wd:Job_Profile>, so the XPath must navigate to <wd:Job_Profile_Data/wd:Effective_Date> within that context.
Analysis of Options
Let's evaluate each option based on the format-date syntax, the XML structure, and the required output format "15-07-2024":
* Option A: format-date('[D01]-[M01]-[Y0001]', wd:Job_Profile_Data/wd:Effective_Date)
* This option places the picture string ('[D01]-[M01]-[Y0001]') as the first parameter and the date value (wd:Job_Profile_Data/wd:Effective_Date) as the second. However, the format-date function requires the date value as the first parameter and the picture string as the second, per the syntax format-date($value, $picture). Reversing the parameters is incorrect and will result in an error or unexpected output, as format-date expects an xs:date? first. Thus, this option is invalid.
* Option B: format-date (wd:Job_Profile_Data/wd:Effective_Date, '[D01]-[M01]-[Y0001]')
* This option correctly follows the format-date syntax:
* First parameter: wd:Job_Profile_Data/wd:Effective_Date, which points to the <wd:
Effective_Date> element in the XML (e.g., 2024-05-15). This is an xs:date value, as Workday web services typically return dates in ISO format (YYYY-MM-DD), which format-date can process.
* Second parameter: '[D01]-[M01]-[Y0001]', which specifies the output format:
* [D01] outputs the day as two digits (e.g., "15").
* [M01] outputs the month as two digits (e.g., "05" for May, but the question requests "07" for July-assuming a test case adjustment or hypothetical transformation).
* [Y0001] outputs the year as four digits (e.g., "2024").
* The XPath wd:Job_Profile_Data/wd:Effective_Date is correctly nested under the <wd:
Job_Profile> context, as the template matches on <wd:Job_Profile>. This would transform "2024-05-15" into "15-05-2024" (or "15-07-2024" if the month is adjusted in the logic), matching the required day-month-year format. This option is valid and correct.
* Option C: format-date (wd:Job_Profile_Data/wd:Effective_Date, '[M01]-[D01]-[Y0001]')
* This option also follows the correct format-date syntax, with the date value first and the picture string second. However, the picture string '[M01]-[D01]-[Y0001]' specifies a month-day-year format:
* [M01] outputs the month first (e.g., "05" for May).

* [D01] outputs the day second (e.g., "15").
* [Y0001] outputs the year last (e.g., "2024").
* This would transform "2024-05-15" into "05-15-2024," which does not match the required "15-07-2024" (day-month-year) format. Thus, this option is incorrect for the specified output.
* Option D: format-date('[M01]-[D01]-[Y0001]', wd:Job_Profile_Data/wd:Effective_Date)
* Similar to Option A, this option reverses the parameters, placing the picture string ('[M01]-[D01]-[Y0001]') first and the date value (wd:Job_Profile_Data/wd:Effective_Date) second. As explained earlier, format-date requires the date value as the first parameter, so this syntax is incorrect and will not work as intended. This option is invalid.

Why Option B is Correct

Option B correctly uses the format-date function with the proper syntax:
* It places the date value (wd:Job_Profile_Data/wd:Effective_Date) as the first parameter, referencing the <wd:Effective_Date> element in the XML.
* It uses the picture string '[D01]-[M01]-[Y0001]' as the second parameter, which formats the date as "DD-MM-YYYY" (e.g., "15-05-2024" for the XML's "2024-05-15," or "15-07-2024" as specified, assuming a month adjustment in the transformation logic).
* The XPath is appropriate for the context, as the template matches <wd:Job_Profile>, and <wd:Job_Profile_Data/wd:Effective_Date> is a valid path within it.

The question's mention of "15-07-2024" suggests either a hypothetical adjustment (e.g., the EIB or XSLT logic modifies the month to July) or a test case variation. Since the XML shows "2024-05-15," the format- date function would output "15-05-2024" with the given picture string, but the principle of formatting day- month-year remains correct. Workday's XSLT implementation supports such transformations, and the format- date function is well-documented for this purpose.

Practical Example in XSLT

Here's how this might look in your XSLT:

```
<xsl:template match="wd:Job_Profile">
<xsl:value-of select="format-date(wd:Job_Profile_Data/wd:Effective_Date, '[D01]-[M01]-[Y0001]')"/>
</xsl:template>
```

This would process the <wd:Effective_Date> (e.g., "2024-05-15") and output "15-05-2024," aligning with the day-month-year format requested (adjusted for the hypothetical "07" if needed elsewhere in the logic).

Verification with Workday Documentation

The Workday Pro Integrations Study Guide and SOAP API Reference (available via Workday Community) detail the use of XPath functions like format-date for transforming web service responses. The Get_Job_Profiles operation returns job profile data, including effective dates, in ISO format, and XSLT transformations are commonly used in EIBs to reformat data. The format-date function's syntax and picture string patterns (e.g., [D01], [M01], [Y0001]) are standard in XPath 2.0, as implemented in Workday's integration tools.

Workday Pro Integrations Study Guide References

* Section: XSLT Transformations in EIBs - Describes using XSLT to transform web service responses, including date formatting with format-date.
* Section: Workday Web Services - Details the Get_Job_Profiles operation and its XML output structure, including <wd:Effective_Date>.
* Section: XPath Functions - Explains the syntax and usage of format-date($value, $picture), including picture string patterns like [D01], [M01], and [Y0001].
* Workday Community SOAP API Reference - Provides examples of date formatting in XSLT for Workday web services.

Option B is the verified answer, as it correctly applies the format-date function to format the <wd:Effective_Date> in the required day-month-year format.

NEW QUESTION # 75

.....

Valid Dumps Workday-Pro-Integrations Pdf □ Download ➡ Workday-Pro-Integrations □□□ for free by simply entering ▸ www.pdfvce.com ◂ website □New Workday-Pro-Integrations Study Plan

- Workday-Pro-Integrations New Soft Simulations □ Workday-Pro-Integrations New Soft Simulations □ Workday-Pro-Integrations Related Content □ Search on ⇛ www.vce4dumps.com ⇚ for ✔ Workday-Pro-Integrations □✔□ to obtain exam materials for free download □Workday-Pro-Integrations Book Pdf
- Workday-Pro-Integrations Valid Exam Experience □ Download Workday-Pro-Integrations Pdf □ Workday-Pro-Integrations Valid Exam Experience □ Download □ Workday-Pro-Integrations □ for free by simply searching on ▸ www.pdfvce.com ◂ □Valid Workday-Pro-Integrations Test Pass4sure
- Valid Workday-Pro-Integrations Test Pass4sure □ Valid Workday-Pro-Integrations Exam Papers □ Valid Dumps Workday-Pro-Integrations Sheet □ Easily obtain free download of " Workday-Pro-Integrations " by searching on ➥ www.exam4labs.com □ □Download Workday-Pro-Integrations Pdf
- New Workday-Pro-Integrations Test Fee □ Valid Workday-Pro-Integrations Test Pass4sure □ New Workday-Pro-Integrations Test Fee □ Search for " Workday-Pro-Integrations " on 【 www.pdfvce.com 】 immediately to obtain a free download □Sample Workday-Pro-Integrations Questions
- Exam Workday-Pro-Integrations Cram Questions □ Valid Workday-Pro-Integrations Exam Questions □ Workday-Pro-Integrations New Soft Simulations □ Enter [www.testkingpass.com] and search for ➠ Workday-Pro-Integrations □ □ to download for free □Valid Workday-Pro-Integrations Test Pass4sure
- 100% Pass 2026 Fantastic Workday Workday-Pro-Integrations: Latest Workday Pro Integrations Certification Exam Exam Camp □ Immediately open ➽ www.pdfvce.com □ and search for ⇛ Workday-Pro-Integrations ⇚ to obtain a free download □Workday-Pro-Integrations New Soft Simulations
- www.pdfdumps.com Workday Workday-Pro-Integrations Desktop Practice Exam □ Open 「 www.pdfdumps.com 」 and search for □ Workday-Pro-Integrations □ to download exam materials for free □Download Workday-Pro-Integrations Pdf
- New Workday-Pro-Integrations Test Fee □ Workday-Pro-Integrations Book Pdf □ Workday-Pro-Integrations Valid Dumps Ebook □ Search for （ Workday-Pro-Integrations ） and download it for free on { www.pdfvce.com } website □Workday-Pro-Integrations Valid Dumps Ebook
- Valid Workday-Pro-Integrations Test Pass4sure □ Valid Workday-Pro-Integrations Exam Papers □ New Workday-Pro-Integrations Test Fee □ Search on [www.pass4test.com] for 「 Workday-Pro-Integrations 」 to obtain exam materials for free download □New Workday-Pro-Integrations Test Fee
- www.stes.tyc.edu.tw, stevejcag988537.ziblogs.com, tegangnpu357726.estate-blog.com, rsavfsu481279.wikitelevisions.com, socialwebnotes.com, 7bookmarks.com, imogenzyhi838593.bloggactivo.com, francesxajs285354.blogaritma.com, jimkylw538952.qodsblog.com, matteohrzy666694.empirewiki.com, Disposable vapes

BONUS!!! Download part of Exams4sures Workday-Pro-Integrations dumps for free: https://drive.google.com/open?id=1KuZMwCzxN2lirQ5iimcsRAK0yjTlUK4G