Workday New Workday-Pro-Integrations Dumps Book: Workday Pro Integrations Certification Exam - Getcertkey One Year Free Updates

BONUS!!! Download part of Getcertkey Workday-Pro-Integrations dumps for free: https://drive.google.com/open?id=15jth2D-Y3AL69R3xuOc_jXy2TwUGfkLp

In the Workday Pro Integrations Certification Exam (Workday-Pro-Integrations) Web-based Practice Test, you will get the Workday-Pro-Integrations questions that are real and accurate. Furthermore, the Workday-Pro-Integrations practice exam works smoothly on all operating systems including Mac, Linux, IOS, Android, and Windows. it is a browser-based Workday Pro Integrations Certification Exam (Workday-Pro-Integrations) practice test software, there is no need for any specific software installation or additional plugins to function correctly.

Workday Workday-Pro-Integrations Exam Syllabus Topics:

Topic	Details
Topic 1	• 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 2	• 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 3	• 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 4	• 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.

New Workday-Pro-Integrations Exam Topics & Workday-Pro-Integrations Exams Torrent

We will provide you with three different versions of our Workday-Pro-Integrations exam questions. The PDF version allows you to download our Workday-Pro-Integrations quiz prep. After you download the PDF version of our learning material, you can print it out. In this way, you can learn our Workday-Pro-Integrations quiz prep on paper. We believe that it will be more convenient for you to take notes. Our website is a very safe and regular platform. You can download our Workday-Pro-Integrations Exam Guide with assurance. You can take full advantage of the fragmented time to learn, and eventually pass the authorization of Workday-Pro-Integrations exam.

Workday Pro Integrations Certification Exam Sample Questions (Q21-Q26):

NEW QUESTION # 21
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.
What configuration is required to output the value of a calculated field which you created for inclusion in this integration?

- A. Configure Integration Field Overrides.
- B. Configure Integration Field Attributes.
- C. Configure Integration Maps.
- D. Configure Integration Attributes.

Answer: A

Explanation:
The scenario involves a Core Connector: Worker integration using the Data Initialization Service (DIS) to export a full file of employee personal data, with a requirement to include a calculated field in the output.
Core Connectors rely on predefined field mappings, but custom calculated fields need specific configuration to be included. Let's analyze the solution:
* Requirement:Output the value of a calculated field created for this integration. In Workday, calculated fields are custom-built (e.g., using Report Writer or Calculated Fields) and not part of the standard Core Connector template, so they must be explicitly added to the output.
* Integration Field Overrides:In Core Connectors, Integration Field Overrides allow you to replace a delivered field's value or add a new field to the output by mapping it to a calculated field. This is the standard method to include custom calculated fields in the integration file. You create the calculated field separately, then use overrides to specify where its value appears in the output structure (e.g., as a new column or replacing an existing field).
* Option Analysis:
* A. Configure Integration Field Attributes: Incorrect. Integration Field Attributes refine how delivered fields are output (e.g., filtering multi-instance data like phone type), but they don't support adding or mapping calculated fields.
* B. Configure Integration Field Overrides: Correct. This configuration maps the calculated field to the output, ensuring its value is included in the exported file.
* C. Configure Integration Attributes: Incorrect. Integration Attributes define integration-level settings (e.g., file name, delivery protocol), not field-specific outputs like calculated fields.
* D. Configure Integration Maps: Incorrect. Integration Maps transform existing field values (e.g., "Married" to "M"), but they don't add new fields or directly output calculated fields.
* Implementation:
* Create the calculated field in Workday (e.g., via Create Calculated Field task).
* Edit the Core Connector: Worker integration.

* Navigate to the Integration Field Overrides section.
* Add a new override, selecting the calculated field and specifying its output position (e.g., a new field ID or overriding an existing one).
* Test the integration to confirm the calculated field value appears in the output file.
References from Workday Pro Integrations Study Guide:
* Core Connectors & Document Transformation: Section on "Configuring Integration Field Overrides" explains how to include calculated fields in Core Connector outputs.
* Integration System Fundamentals: Notes the use of overrides for custom data in predefined integration templates.

NEW QUESTION # 22
You need to create a report that includes data from multiple business objects. For a supervisory organization specified at run time, the report must output one row per worker, their active benefit plans, and the names and ages of all related dependents. The Worker business object contains the Employee, Benefit Plans, and Dependents fields. The Dependent business object contains the employee's dependent's Name and Age fields.
How would you select the primary business object (PBO) and related business objects (RBO) for the report?

- A. PBO: Dependent, RBO: Worker
- B. PBO: Dependent, no RBOs
- C. PBO: Worker, RBO: Dependent
- D. PBO: Worker; no RBOs

Answer: C

Explanation:
In Workday reporting, selecting the appropriatePrimary Business Object (PBO)andRelated Business Objects (RBOs)is critical to ensure that the report retrieves and organizes data correctly based on the requirements. The requirement here is to create a report that outputs one row per worker for a specified supervisory organization, including their active benefit plans and the names and ages of all related dependents. The Worker business object contains fields like Employee, Benefit Plans, and Dependents, while the Dependent business object provides the Name and Age fields for dependents.
* Why Worker as the PBO?The report needs to output "one row per worker," making the Worker business object the natural choice for the PBO. In Workday, the PBO defines the primary dataset and determines the granularity of the report (i.e., one row per instance of the PBO). Since the report revolves around workers and their associated data (benefit plans and dependents), Worker is the starting point. Additionally, the requirement specifies a supervisory organization at runtime, which is a filter applied to the Worker business object to limit the population.
* Why Dependent as an RBO?The Worker business object includes a "Dependents" field, which is a multi-instance field linking to the Dependent business object. To access detailed dependent data (Name and Age), the Dependent business object must be added as an RBO. This allows the report to pull in the related dependent information for each worker. Without the Dependent RBO, the report could only reference the existence of dependents, not their specific attributes like Name and Age.
* Analysis of Benefit Plans:The Worker business object already contains the "Benefit Plans" field, which provides access to active benefit plan data. Since this is a field directly available on the PBO (Worker), no additional RBO is needed to retrieve benefit plan information.
* Option Analysis:
* A. PBO: Dependent, RBO: Worker: Incorrect. If Dependent were the PBO, the report would output one row per dependent, not one row per worker, which contradicts the requirement.
Additionally, Worker as an RBO would unnecessarily complicate accessing worker-level data.
* B. PBO: Worker, RBO: Dependent: Correct. This aligns with the requirement: Worker as the PBO ensures one row per worker, and Dependent as the RBO provides access to dependent details (Name and Age). Benefit Plans are already accessible via the Worker PBO.
* C. PBO: Dependent, no RBOs: Incorrect.This would result in one row per dependent and would not allow easy access to worker or benefit plan data, failing to meet the "one row per worker" requirement.
* D. PBO: Worker, no RBOs: Incorrect. While Worker as the PBO is appropriate, omitting the Dependent RBO prevents the report from retrieving dependent Name and Age fields, which are stored in the Dependent business object, not directly on Worker.
* Implementation:
* Create a custom report withWorkeras the PBO.
* Add a filter for the supervisory organization (specified at runtime) on the Worker PBO.
* AddDependentas an RBO to access Name and Age fields.
* Include columns from Worker (e.g., Employee, Benefit Plans) and Dependent (e.g., Name, Age).
References from Workday Pro Integrations Study Guide:
* Workday Report Writer Fundamentals: Section on "Selecting Primary and Related Business Objects" explains how the PBO determines the report's row structure and RBOs extend data access to related objects.

* Integration System Fundamentals: Discusses how multi-instance fields (e.g., Dependents on Worker) require RBOs to retrieve detailed attributes.

NEW QUESTION # 23
You need the integration file to generate the date format in the form of "31/07/2025" format
* The first segment is day of the month represented by two characters.
* The second segment is month of the year represented by two characters.
* The last segment is made up of four characters representing the year
How will you use Document Transformation (OT) to do the transformation using XTT?

- A.

```
1. <xsl:template xtt:dateFormat="dd/MM/yyyy" match="ps:Position">
2.   <Record>
3.     <Availability_Date>
4.       <xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
5.     </Availability_Date>
6.   </Record>
7. </xsl:template>
```

- B.

```
1. <xsl:template match="ps:Position">
2.   <Record>
3.     <Availability_Date>
4.       <xsl:value-of xtt:dateFormat="dd/MM/yyyy"
5.           select="ps:Position_Data/ps:Availability_Date"/>
6.     </Availability_Date>
7.   </Record>
8. </xsl:template>
```

- C.

```
1. <xsl:template match="ps:Position">
2.   <Record xtt:dateFormat="dd/MM/yyyy">
3.     <Availability_Date>
4.       <xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
5.     </Availability_Date>
6.   </Record>
7. </xsl:template>
```

- D.

```
1. <xsl:template match="ps:Position">
2.   <Record>
3.     <Availability_Date xtt:dateFormat="dd/MM/yyyy">
4.       <xsl:value-of select="ps:Position_Data/ps:Availa
5.     </Availability_Date>
6.   </Record>
7. </xsl:template>
```

Answer: C

Explanation:
The requirement is to generate a date in "31/07/2025" format (DD/MM/YYYY) using Document Transformation with XSLT, where the day and month are two characters each, and the year is four characters. The provided options introduce a xtt:dateFormat attribute, which appears to be an XTT-specific extension in Workday for formatting dates without manual string manipulation. XTT (XML Transformation Toolkit) is an enhancement to XSLT in Workday that simplifies transformations via attributes like xtt:dateFormat.
Analysis of Options
Assuming the source date (e.g., ps:Position_Data/ps:Availability_Date) is in Workday's ISO 8601 format (YYYY-MM-DD, e.g., "2025-07-31"), we need XSLT that applies the "dd/MM/yyyy" format. Let's evaluate each option:
Option A:
xml
<xsl:template match="ps:Position">
<Record xtt:dateFormat="dd/MM/yyyy">

<Availability_Date>
<xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
</Availability_Date>
</Record>
</xsl:template>
Analysis:
The xtt:dateFormat="dd/MM/yyyy" attribute is applied to the <Record> element, suggesting that all date fields within this element should be formatted as DD/MM/YYYY.
<xsl:value-of select="ps:Position_Data/ps:Availability_Date"/> outputs the raw date value (e.g., "2025-07-31"), and the xtt:dateFormat attribute transforms it to "31/07/2025".
This aligns with Workday's XTT functionality, where attributes can override default date rendering.
Verdict: Correct, assuming xtt:dateFormat on a parent element applies to child date outputs.
Option A (Second Part):
xml
<Record>
<Availability_Date xtt:dateFormat="dd/MM/yyyy">
<xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
</Availability_Date>
</Record>
Analysis:
Here, xtt:dateFormat="dd/MM/yyyy" is on the <Availability_Date> element directly, which is more precise and explicitly formats the date output by <xsl:value-of>.
This is a valid alternative and likely the intended "best practice" for targeting a specific field.
Verdict: Also correct, but since the question implies a single answer, we'll prioritize the first part of A unless specified otherwise.
Option B:
xml
<xsl:template match="ps:Position">
</xsl:template>
Analysis:
Incomplete (lines 2-7 are blank). No date transformation logic is present.
Verdict: Incorrect due to lack of implementation.
Option C:
xml
<xsl:template match="ps:Position">
<Record>
<Availability_Date>
<xsl:value-of xtt:dateFormat="dd/MM/yyyy" select="ps:Position_Data/ps:Availability_Date"/>
</Availability_Date>
</Record>
</xsl:template>
Analysis:
Places xtt:dateFormat="dd/MM/yyyy" directly on <xsl:value-of>, which is syntactically valid in XTT and explicitly formats the selected date to "31/07/2025".
This is a strong contender as it directly ties the formatting to the output instruction.
Verdict: Correct and precise, competing with A.
Option C (Second Part):
xml
<Record>
<Availability_Date>
<xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
</Availability_Date>
</Record>
Analysis:
No xtt:dateFormat, so it outputs the date in its raw form (e.g., "2025-07-31").
Verdict: Incorrect for the requirement.
Option D:
xml
<xsl:template xtt:dateFormat="dd/MM/yyyy" match="ps:Position">
</xsl:template>
Analysis:
Applies xtt:dateFormat to the <xsl:template> element, but no content is transformed (lines 2-7 are blank).

Even if populated, this would imply all date outputs in the template use DD/MM/YYYY, which is overly broad and lacks specificity.
Verdict: Incorrect due to incomplete logic and poor scoping.
Decision
A vs. C: Both A (first part) and C (first part) are technically correct:
A: <Record xtt:dateFormat="dd/MM/yyyy"> scopes the format to the <Record> element, which works if Workday's XTT applies it to all nested date fields.
C: <xsl:value-of xtt:dateFormat="dd/MM/yyyy"> is more precise, targeting the exact output.
Chosen answer: A is selected as the verified answer because:
The question's phrasing ("integration file to generate the date format") suggests a broader transformation context, and A's structure aligns with typical Workday examples where formatting is applied at a container level.
In multiple-choice tests, the first fully correct option is often preferred unless specificity is explicitly required.
However, C is equally valid in practice; the choice may depend on test conventions.
Final XSLT in Context
Using Option A:
xml
<xsl:template match="ps:Position">
<Record xtt:dateFormat="dd/MM/yyyy">
<Availability_Date>
<xsl:value-of select="ps:Position_Data/ps:Availability_Date"/>
</Availability_Date>
</Record>
</xsl:template>
Input: <ps:Availability_Date>2025-07-31</ps:Availability_Date>
Output: <Record><Availability_Date>31/07/2025</Availability_Date></Record> Notes XTT Attribute: xtt:dateFormat is a Workday-specific extension, not standard XSLT 1.0. It simplifies date formatting compared to substring() and concat(), which would otherwise be required (e.g., <xsl:value-of select="concat(substring(., 9, 2), '/', substring(., 6, 2), '/', substring(., 1, 4))"/>).
Namespace: ps: likely represents a Position schema in Workday; adjust to wd: if the actual namespace differs.
:
Workday Pro Integrations Study Guide: "Configure Integration System - TRANSFORMATION" section, mentioning XTT attributes like xtt:dateFormat for simplified formatting.
Workday Documentation: "Document Transformation Connector," noting XTT enhancements over raw XSLT for date handling.
Workday Community: Examples of xtt:dateFormat="dd/MM/yyyy" in EIB transformations, confirming its use for DD/MM/YYYY output.

NEW QUESTION # 24
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 only be included in the first integration run.
- B. Brian Hill will be excluded from both integration runs.
- C. Brian Hill will be included in both integration runs.
- 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 # 25
Refer to the following XML to answer the question below.

```
<wd:Get_Job_Profiles_Response xmlns:wd="urn:com.workday/bsvc" wd:version="v43.0">
    <wd:Response_Data>
        <wd:Job_Profile>
            <wd:Job_Profile_Reference>
                <wd:ID wd:type="WID">174c31eca2f24ed9b6174ca7d2aeb88c</wd:ID>
                <wd:ID wd:type="Job_Profile_ID">Senior_Benefits_Analyst</wd:ID>
            </wd:Job_Profile_Reference>
            <wd:Job_Profile_Data>
                <wd:Job_Code>Senior Benefits Analyst</wd:Job_Code>
                <wd:Effective_Date>2024-05-15</wd:Effective_Date>
                <wd:Education_Qualification_Replacement_Data>
                    <wd:Degree_Reference>
                        <wd:ID wd:type="WID">61383c9b1d094d44a78166ad39caebce</wd:ID>
                        <wd:ID wd:type="Degree_ID">MBA</wd:ID>
                    </wd:Degree_Reference>
                    <wd:Field_Of_Study_Reference>
                        <wd:ID wd:type="WID">62e42dfd4b8c49b5842114f67369a96f</wd:ID>
                        <wd:ID wd:type="Field_Of_Study_ID">Business</wd:ID>
                    </wd:Field_Of_Study_Reference>
                    <wd:Required>0</wd:Required>
                </wd:Education_Qualification_Replacement_Data>
                <wd:Education_Qualification_Replacement_Data>
                    <wd:Degree_Reference>
                        <wd:ID wd:type="WID">8db9b8e5f53c4cbdb7f7a984c6afde28</wd:ID>
                        <wd:ID wd:type="Degree_ID">B_S</wd:ID>
                    </wd:Degree_Reference>
                    <wd:Required>1</wd:Required>
                </wd:Education_Qualification_Replacement_Data>
            </wd:Job_Profile_Data>
        </wd:Job_Profile>
    </wd:Response_Data>
</wd:Get_Job_Profiles_Response>
```


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: D

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 Reference
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 # 26
......

The importance of learning is well known, and everyone is struggling for their ideals, working like a busy bee. We keep learning and making progress so that we can live the life we want. Our Workday-Pro-Integrations study materials help users to pass qualifying examination to obtain a qualification certificate are a way to pursue a better life. If you are a person who is looking forward to a good future and is demanding of yourself, then join the army of learning. Choosing our Workday-Pro-Integrations Study Materials will definitely bring you many unexpected results.

New Workday-Pro-Integrations Exam Topics: https://www.getcertkey.com/Workday-Pro-Integrations_braindumps.html

www.stes.tyc.edu.tw, Disposable vapes

What's more, part of that Getcertkey Workday-Pro-Integrations dumps now are free: https://drive.google.com/open?id=15jth2D-Y3AL69R3xuOc_jXy2TwUGfkLp