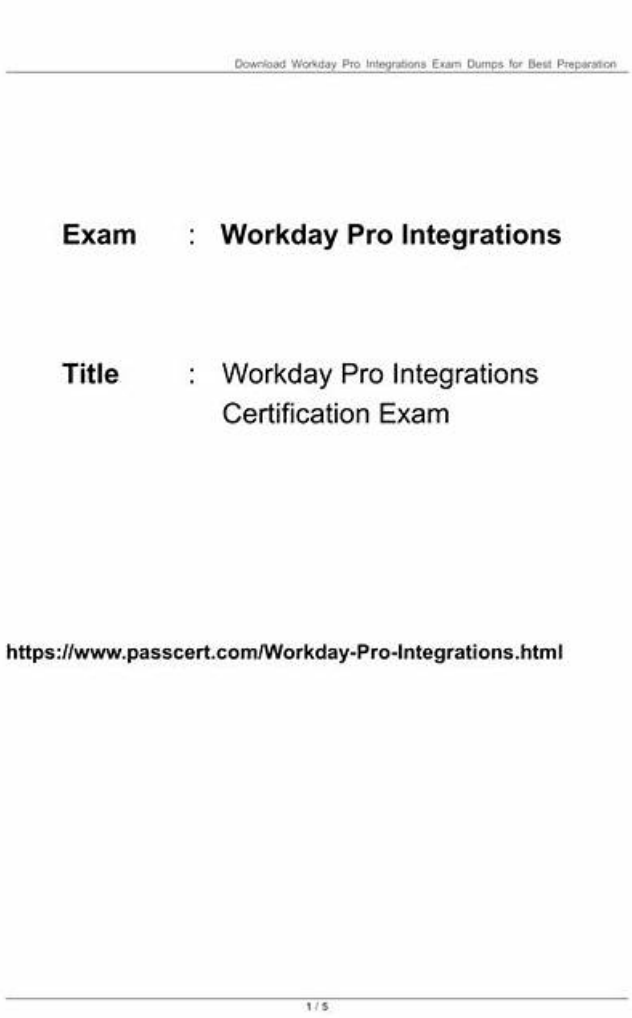
Most workable Workday-Pro-Integrations guide materials: Workday Pro Integrations Certification Exam Provide you wonderful Exam Braindumps - Lead1Pass



2025 Latest Lead1Pass Workday-Pro-Integrations PDF Dumps and Workday-Pro-Integrations Exam Engine Free Share: https://drive.google.com/open?id=16QCWI1cs2eH2_jnxt_uvTUG127IHgB3D

Getting the Workday Workday-Pro-Integrations certification exam is necessary in order to get a job in your desired tech company. Success in the Workday Pro Integrations Certification Exam (Workday-Pro-Integrations) certification exam gives you an edge over the others because you will have certified skills. The Workday Workday-Pro-Integrations certification exam badge will make a good impression on the interviewer. Most of the people planning to attempt the Workday-Pro-Integrations Exam are confused that how will they prepare and pass Workday-Pro-Integrations exam with good grades. Many don't find real Workday-Pro-Integrations exam questions and face loss of money and time.

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	• 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 3	• 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.

>> Latest Workday-Pro-Integrations Exam Labs <<

Workday Workday-Pro-Integrations Test Dumps Pdf, Workday-Pro-Integrations Latest Braindumps Book

Did you often feel helpless and confused during the preparation of the Workday-Pro-Integrations exam? Do you want to find an expert to help but feel bad about the expensive tutoring costs? Don't worry. Our Workday-Pro-Integrations exam questions can help you to solve all the problems. Our Workday-Pro-Integrations Study Material always regards helping students to pass the exam as it is own mission. And we have successfully helped numerous of the candidates pass their exams.

Workday Pro Integrations Certification Exam Sample Questions (Q16-Q21):

NEW QUESTION # 16
Refer to the following XML to answer the question below.

```
<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">61393c9b1094d44a73166ad39caebce</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">Economics</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>
```


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 a template against <wd:Job_Profile>.
What XPath syntax would be used to select the value of the wd:Job_Code element when the <xsl:value-of> element is placed within the template which matches on <wd:Job_Profile>?

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

Answer: A

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:
Job_Code> element. The root template of your XSLT matches on <wd:Get_Job_Profiles_Response> and applies a template to <wd:Job_Profile>. Within this template, you use the <xsl:value-of> element to extract the <wd:Job_Code> 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 are:
* <wd:Job_Profile_Reference>, which contains <wd:ID> elements (e.g., a Job_Profile_ID).
* <wd:Job_Profile_Data>, which contains <wd:Job_Code> with the value
Senior_Benefits_Analyst.
The task is to select the value of <wd:Job_Code> (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 <wd:Job_Code>.
Analysis of Options
Let's evaluate each option based on the XML structure and XPath syntax rules:
* Option A: wd:Job_Profile/wd:Job_Profile_Data/wd:Job_Code
* This XPath starts from wd:Job_Profile and navigates to wd:Job_Profile_Data/wd:Job_Code.
However, in the XML, <wd:Job_Profile> is the parent element, and <wd:Job_Profile_Data> is a direct child containing <wd:Job_Code>. The path wd:Job_Profile/wd:Job_Profile_Data/wd:
Job_Code is technically correct in terms of structure, as it follows the hierarchy:
* <wd:Job_Profile> # <wd:Job_Profile_Data> # <wd:Job_Code>.
* However, since the template matches <wd:Job_Profile>, the context node is already <wd:
Job_Profile>. You don't need to include wd:Job_Profile/ at the beginning of the XPath unless navigating from a higher level. Starting directly with wd:Job_Profile_Data/wd:Job_Code (Option C) is more concise and appropriate for the context. This option is technically valid but redundant and less efficient, making it less preferred compared to Option C.
* Option B: wd:Job_Profile_Data[@wd:Job_Code]
* This XPath uses an attribute selector ([@wd:Job_Code]) to filter <wd:Job_Profile_Data> based on an attribute named wd:Job_Code. However, examining the XML, <wd:Job_Profile_Data> does not have a wd:Job_Code attribute-it has a child element <wd:Job_Code> with the value
"Senior_Benefits_Analyst." The [@attribute] syntax is used for attributes, not child elements, so this XPath is incorrect. It would not select the <wd:Job_Code> value and would likely return no results or an error. This option is invalid.
* Option C: wd:Job_Profile_Data/wd:Job_Code
* This XPath starts from wd:Job_Profile_Data (a direct child of <wd:Job_Profile>) and navigates to wd:Job_Code. Since the template matches <wd:Job_Profile>, the context node is <wd:
Job_Profile>, and wd:Job_Profile_Data/wd:Job_Code correctly points to the <wd:Job_Code> element within <wd:Job_Profile_Data>. This path is:
* Concise and appropriate for the context.
* Directly selects the value "Senior_Benefits_Analyst" when used with <xsl:value-of>.
* Matches the XML structure, as <wd:Job_Profile_Data> contains <wd:Job_Code> as a child.
* This is the most straightforward and correct option for selecting the <wd:Job_Code> value within the <wd:Job_Profile> template.
* Option D: wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID']
* This XPath navigates to <wd:Job_Profile_Reference> (a child of <wd:Job_Profile>) and then to <wd:ID> with an attribute wd:type="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 XPath wd:Job_Profile_Reference/wd:ID[@wd:type='Job_Profile_ID'] selects the <wd:ID> element with wd:type="Job_Profile_ID", which has the value "Senior_Benefits_Analyst." However, this is not the <wd:Job_Code> value-the <wd:Job_Code> is a separate element under

<wd:Job_Profile_Data>, not <wd:Job_Profile_Reference>. The question specifically asks for the <wd:Job_Code> value, so this option is incorrect, as it selects a different piece of data (the job profile ID, not the job code).
Why Option C is Correct
Option C, wd:Job_Profile_Data/wd:Job_Code, 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_Data/wd:Job_Code>, which directly selects the <wd:Job_Code> element's value ("Senior_Benefits_Analyst").
* It is concise and aligns with standard XPath navigation in XSLT, avoiding unnecessary redundancy (unlike Option A) or incorrect attribute selectors (unlike Option B).
* It matches the XML structure, where <wd:Job_Profile_Data> is a child of <wd:Job_Profile> and contains <wd:Job_Code> as a child.
* When used with <xsl:value-of select="wd:Job_Profile_Data/wd:Job_Code"/> in the template, it outputs the job code value, fulfilling the requirement.
Practical Example in XSLT
Here's how this might look in your XSLT:
xml
WrapCopy
<xsl:template match="wd:Job_Profile">
<xsl:value-of select="wd:Job_Profile_Data/wd:Job_Code"/>
</xsl:template>
This would output "Senior_Benefits_Analyst" for the <wd:Job_Code> element 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_Data> as the container for job profile details, including <wd:Job_Code>. The guide emphasizes using relative XPath paths within templates to navigate from the matched element (e.g., <wd:Job_Profile>) to child elements like <wd:Job_Profile_Data/wd:Job_Code>.
Workday Pro Integrations Study Guide References
* Section: XSLT Transformations in EIBs - Describes using XSLT to transform web service responses, including selecting elements with XPath.
* Section: Workday Web Services - Details the Get_Job_Profiles operation and its XML output structure, including <wd:Job_Profile_Data> and <wd:Job_Code>.
* Section: XPath Syntax - Explains how to navigate XML hierarchies in Workday XSLT, using relative paths like wd:Job_Profile_Data/wd:Job_Code from a <wd:Job_Profile> context.
* Workday Community SOAP API Reference - Provides examples of XPath navigation for Workday web service responses.
Option C is the verified answer, as it correctly selects the <wd:Job_Code> value using the appropriate XPath syntax within the <wd:Job_Profile> template context.

NEW QUESTION # 17
What is the relationship between an ISU (Integration System User) and an ISSG (Integration System Security Group)?

- A. The ISU is a member of the ISSG.
- B. The ISU controls what accounts are in the ISSG.
- C. The ISU owns the ISSG.
- D. The ISU grants security policies to the ISSG.

Answer: A

Explanation:
This question explores the relationship between an Integration System User (ISU) and an Integration System Security Group (ISSG) in Workday Pro Integrations, focusing on how security is structured for integrations. Let's analyze the relationship and evaluate each option to determine the correct answer.
Understanding ISU and ISSG in Workday
Integration System User (ISU): An ISU is a dedicated user account in Workday specifically designed for integrations. It acts as a "robot account" or service account, used by integration systems to interact with Workday via APIs, web services, or other integration mechanisms (e.g., EIBs, Core Connectors). ISUs are typically configured with a username, password, and specific security settings, such as disabling UI sessions and setting session timeouts to prevent expiration (commonly set to 0 minutes). ISUs are not human users but are instead programmatic accounts for automated processes.
Integration System Security Group (ISSG): An ISSG is a security container or group in Workday that defines the permissions and access rights for integration systems. ISSGs are used to manage what data and functionalities an integration (or its associated ISU) can access or modify within Workday. There are two types of ISSGs:

Unconstrained: Allows access to all data instances secured by the group.
Constrained: Limits access to a subset of data instances based on context (e.g., specific segments or data scopes).ISSGs are configured with domain security policies, granting permissions like "Get" (read), "Put" (write), "View," or "Modify" for specific domains (e.g., Worker Data, Integration Build).
Relationship Between ISU and ISSG: In Workday, security for integrations is managed through a hierarchical structure. An ISU is associated with or assigned to an ISSG to inherit its permissions. The ISSG acts as the security policy container, defining what the ISU can do, while the ISU is the account executing those actions. This relationship ensures that integrations have controlled, audited access to Workday data and functions, adhering to the principle of least privilege.
Evaluating Each Option
Let's assess each option based on Workday's security model for integrations:
Option A: The ISU is a member of the ISSG.
Analysis: This is correct. In Workday, an ISU is assigned to or associated with an ISSG to gain the necessary permissions. The ISSG serves as a security group that contains one or more ISUs, granting them access to specific domains and functionalities. For example, when creating an ISU, you use the "Create Integration System User" task, and then assign it to an ISSG via the "Assign Integration System Security Groups" or "Maintain Permissions for Security Group" tasks. Multiple ISUs can belong to the same ISSG, inheriting its permissions. This aligns with Workday's security framework, where security groups (like ISSGs) manage user (or ISU) access.
Why It Fits: The ISU is a "member" of the ISSG in the sense that it is linked to the group to receive its permissions, enabling secure integration operations. This is a standard practice for managing integration security in Workday.
Option B: The ISU owns the ISSG.
Analysis: This is incorrect. In Workday, ISUs do not "own" ISSGs. Ownership or control of security groups is not a concept applicable to ISUs, which are service accounts for integrations, not administrative entities with authority over security structures. ISSGs are created and managed by Workday administrators or security professionals using tasks like "Create Security Group" and "Maintain Permissions for Security Group." The ISU is simply a user account assigned to the ISSG, not its owner or controller.
Why It Doesn't Fit: Ownership implies administrative control, which ISUs lack; they are designed for execution, not management of security groups.
Option C: The ISU grants security policies to the ISSG.
Analysis: This is incorrect. ISUs do not have the authority to grant or modify security policies for ISSGs. Security policies are defined and assigned to ISSGs by Workday administrators or security roles with appropriate permissions (e.g., Security Configuration domain access). ISUs are passive accounts that execute integrations based on the permissions granted by the ISSG they are assigned to. Granting permissions is an administrative function, not an ISU capability.
Why It Doesn't Fit: ISUs are integration accounts, not security administrators, so they cannot modify or grant policies to ISSGs.
Option D: The ISU controls what accounts are in the ISSG.
Analysis: This is incorrect. ISUs do not control membership or configuration of ISSGs. Adding or removing accounts (including other ISUs) from an ISSG is an administrative task performed by users with security configuration permissions, using tasks like "Maintain Permissions for Security Group." ISUs are limited to executing integration tasks based on their assigned ISSG permissions, not managing group membership.
Why It Doesn't Fit: ISUs lack the authority to manage ISSG membership or structure, as they are not administrative accounts but integration-specific service accounts.
Final Verification
Based on Workday's security model, the correct relationship is that an ISU is a member of an ISSG, inheriting its permissions to perform integration tasks. This is consistent with the principle of least privilege, where ISSGs define access, and ISUs execute within those boundaries. The other options misattribute administrative or ownership roles to ISUs, which are not supported by Workday's design.
Supporting Information
The relationship is grounded in Workday's integration security practices, including:
Creating an ISU via the "Create Integration System User" task.
Creating an ISSG via the "Create Security Group" task, selecting "Integration System Security Group (Unconstrained)" or "Constrained." Assigning the ISU to the ISSG using tasks like "Assign Integration System Security Groups" or "Maintain Permissions for Security Group." Configuring domain security policies (e.g., Get, Put) for the ISSG to control ISU access to domains like Worker Data, Integration Build, etc.
Activating security changes via "Activate Pending Security Policy Changes." This structure ensures secure, controlled access for integrations, with ISSGs acting as the permission container and ISUs as the executing accounts.
Key Reference
The explanation aligns with Workday Pro Integrations documentation and best practices, including:
Integration security overviews and training on Workday Community.
Guides for creating ISUs and ISSGs in implementation documentation (e.g., NetIQ, Microsoft Learn, Reco.ai).
Tutorials on configuring domain permissions and security groups for integrations (e.g., ServiceNow, Apideck, Surety Systems).

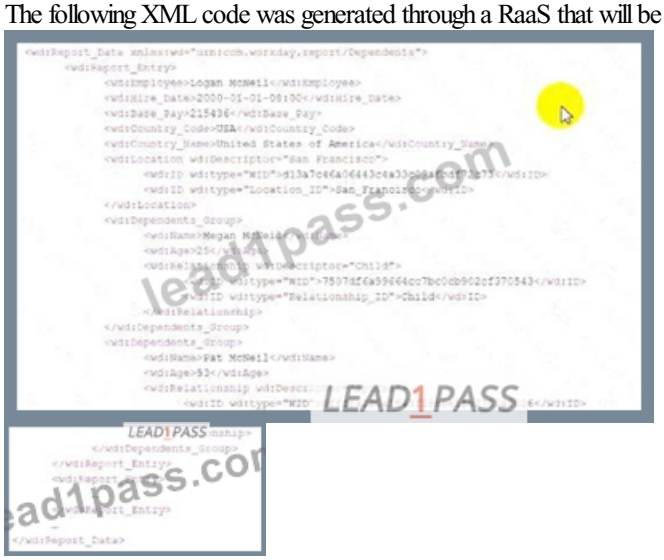
NEW QUESTION # 18
Which features must all XSLT files contain to be considered valid?

- A. A root element, namespace, and at least one template
- B. A template, a prefix, and a header
- C. A header, a footer, and a namespace
- D. A root element, namespace, and at least one transformation

Answer: A

Explanation:
A valid XSLT file must include the following key components:
Root Element: <xsl:stylesheet> or <xsl:transform>
Namespace Declaration: Usually xmlns:xsl="http://www.w3.org/1999/XSL/Transform" At Least One <xsl:template> to define transformation behavior From W3C and Workday documentation:
"A valid XSLT file must contain a stylesheet root element, a namespace, and at least one template to be considered executable."
Why others are incorrect:
A . Prefix and header are not structural requirements.
B . "Transformation" is vague; it's the template that implements it.
C . Headers and footers are not required or defined elements in XSLT.

NEW QUESTION # 19
The following XML code was generated through a RaaS that will be used in an EIB.



Within a template that matches on wd:Report_Entry, what XPath expression do you use to select the value of the Relationship_ID element?

- A. wd:Dependents_Group/wd:Relationship/wd:ID
- B. ./wd:Dependents_Group/wd:Relationship/wd:ID
- C. wd:Dependents_Group/wd:Relationship/wd:ID/wd:type='Relationship_ID'
- D. wd:Dependents_Group/wd:Relationship/wd:ID/wd:type='Relationship_ID'

Answer: B

Explanation:
The XML fragment shown follows the Report#as#a#Service (RaaS) structure typical for Workday custom report output:
<wd:Report_Entry>
<wd:Dependents_Group>
<wd:Name>Megan McNeil</wd:Name>
<wd:Age>25</wd:Age>
<wd:Relationship wd:Descriptor="Child">
<wd:ID wd:type="WID">7507df6a99664ce7bc0cb902cf370543</wd:ID>
<wd:ID wd:type="Relationship_ID">Child</wd:ID>
</wd:Relationship>
</wd:Dependents_Group>

</wd:Report_Entry>
Inside each <wd:Report_Entry>, the target value Relationship_ID resides under:
wd:Dependents_Group
wd:Relationship
wd:ID (wd:type="Relationship_ID")
When writing the template:
<xsl:template match="wd:Report_Entry">
XSLT uses a relative XPath (starting with ./) to navigate from the matched node.
Therefore, the correct XPath should be:
/wd:Dependents_Group/wd:Relationship/wd:ID
That expression selects the wd:ID element so you can then test/extract where wd:type="Relationship_ID".
Why the other options are incorrect:
Option
Why Incorrect
A & B
These use an equality test incorrectly inside the XPath expression - they would not return the node value and are syntactically invalid for value extraction.
C
Missing ./ - would still work in many cases, but Workday XSLT best practice is to use relative paths when inside a match.
Workday Pro Integration guidance for RaaS/XSLT stresses:
Always scope node selection relative to the current context tree using prefix#qualified XPath expressions.
Reference:Workday Pro: Integrations - XSLT for Workday XML (Namespaces, Context Node, Relationship ID Extraction)W3C XSLT Specification - Relative XPath from context node

NEW QUESTION # 20
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">61383c9b1d094d44a73166ad39caebce</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">Economics</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 using a web service enabled report to output worker data along with their dependents. 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:Dependents_Group elements by using an <xsl:apply-templates> element.
What XPath syntax would be used as the select for the apply templates so as to iterate over only the wd:Dependents_Group elements where the dependent relationship is Child?

- A. wd:Dependents_Group/wd:Relationship='Child'
- B. wd:Dependents_Group[@wd:Relationship='Child']
- C. wd:Dependents_Group/@wd:Relationship='Child'
- D. wd:Dependents_Group[wd:Relationship='Child']

Answer: D

Explanation:
In Workday integrations, XSLT (Extensible Stylesheet Language Transformations) is commonly used to transform XML data, such as the output from an Enterprise Interface Builder (EIB) or a web service-enabled report, into a format suitable for third-party systems. In this scenario, you are tasked with writing XSLT to process the wd:Dependents_Group elements within a report output to iterate only over those where the dependent relationship is "Child." The correct XPath syntax for the select attribute of an <xsl:apply-templates> element is critical to ensure accurate data transformation.
Here's why option B is correct:
XPath Syntax In XPath, square brackets [ ] are used to specify predicates or conditions to filter elements. The condition wd:Relationship='Child' checks if the wd:Relationship element (or attribute, depending on the XML structure) has the value "Child." When applied to wd:Dependents_Group, the expression wd:Dependents_Group[wd:Relationship='Child'] selects only those wd:Dependents_Group elements that contain a wd:Relationship child element with the value "Child." Context in XSLT: Within an <xsl:apply-templates> element, the select attribute uses XPath to specify which nodes to process. This syntax ensures that the template only applies to wd:Dependents_Group elements where the dependent is a child, aligning with the requirement to conditionally process only those specific dependents.
XML Structure Alignment: Based on the provided XML snippet, wd:Dependents_Group likely contains child elements or attributes, including wd:Relationship. The correct XPath assumes wd:Relationship is an element (not an attribute), as is common in Workday XML structures. Therefore, wd:Dependents_Group[wd:Relationship='Child'] is the appropriate syntax to filter and iterate over the desired elements.
Why not the other options?
A . wd:Dependents_Group[@wd:Relationship='Child']: This syntax uses @ to indicate that wd:Relationship is an attribute of wd:Dependents_Group, not an element. If wd:Relationship is not defined as an attribute in the XML (as is typical in Workday's XML structure, where it's often an element), this would result in no matches, making it incorrect.
C . wd:Dependents_Group/wd:Relationship='Child': This is not a valid XPath expression for a predicate. It attempts to navigate to wd:Relationship as a child but does not use square brackets [ ] to create a filtering condition. This would be interpreted as selecting wd:Relationship elements under wd:Dependents_Group, but it wouldn't filter based on the value "Child" correctly within an <xsl:apply-templates> context.
D . wd:Dependents_Group/@wd:Relationship='Child': Similar to option A, this assumes wd:Relationship is an attribute, which may not match the XML structure. Additionally, it lacks the predicate structure [ ], making it invalid for filtering in this context.
To implement this in XSLT:
You would write an <xsl:apply-templates> element within your template matching wd:Report_Entry, with the select attribute set to wd:Dependents_Group[wd:Relationship='Child']. This ensures that only wd:Dependents_Group elements with a wd:Relationship value of "Child" are processed by the corresponding templates, effectively filtering out other dependent relationships (e.g., Spouse, Parent) in the transformation.
This approach ensures the XSLT transformation aligns with Workday's XML structure and integration requirements for processing worker data and dependents in an EIB or web service-enabled report.
:
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.
Workday EIB and Web Services Guide: Chapter on "XML and XSLT for Report Data" - Explains the structure of Workday XML (e.g., wd:Dependents_Group, wd:Relationship) 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 values.

NEW QUESTION # 21
.....

Our Workday-Pro-Integrations study materials can have such a high pass rate, and it is the result of step by step that all members uphold the concept of customer first. If you use a trial version of Workday-Pro-Integrations training prep, you can find that our study materials have such a high passing rate and so many users support it. After using the trial version, we believe that you will be willing to choose Workday-Pro-Integrations Exam Questions.

Workday-Pro-Integrations Test Dumps Pdf: https://www.lead1pass.com/Workday/Workday-Pro-Integrations-practice-exam-dumps.html

BONUS!!! Download part of Lead1Pass Workday-Pro-Integrations dumps for free: https://drive.google.com/open?id=16QCWI1cs2eH2_jnxt_uvTUG127IHgB3D