# 試験の準備方法-実際的なWorkday-Pro-Integrations試験合格攻略試験-完璧なWorkday-Pro-Integrations無料試験


2026年Jpexamの最新Workday-Pro-Integrations PDFダンプおよびWorkday-Pro-Integrations試験エンジンの無料共有：https://drive.google.com/open?id=16ohf4UdFge-jfkc5QUZh_AgZKcNjkhce

Workday-Pro-Integrations試験問題は、専門家によって編集され、長年の経験を持つ専門家によって承認されています。言語は理解しやすいため、学習者に障害はありません。WorkdayのWorkday-Pro-Integrationsガイド急流は誰にでも適しています。コンテンツは習得が容易であり、重要な情報を簡素化しました。 Workday-Pro-Integrationsテストトレントは、より重要な情報を少ない質問と回答で伝え、学習をリラックスして効率的にします。 Workday-Pro-Integrations試験問題を使用すると、Workday-Pro-Integrations試験に簡単に合格できます。

Jpexamを選択したら、成功が遠くではありません。Jpexamが提供するWorkdayのWorkday-Pro-Integrations認証試験問題集が君の試験に合格させます。テストの時に有効なツルが必要でございます。

**>> Workday-Pro-Integrations試験合格攻略 <<**

## Workday-Pro-Integrations試験の準備方法｜正確的なWorkday-Pro-Integrations試験合格攻略試験｜最新のWorkday Pro Integrations Certification Exam無料試験

さまざまな人々がさまざまな学習習慣を持っているという事実を踏まえて、3つのWorkday-Pro-Integrationsトレーニング質問バージョンをご案内します。さらに、Workday-Pro-Integrations学習教材のデモを自由にダウンロードして検討することもできます。そのような試用に追加料金は発生しないことをお約束します。逆に、Workday-Pro-Integrations試験問題のデモを試して、十分な内容を選択することを心からお勧めします。 Workday-Pro-Integrationsトレーニングガイドは、時間とお金をかけて購入する価値があります。

## Workday Workday-Pro-Integrations 認定試験の出題範囲：

| トピック | 出題範囲 |
|---|---|
| トピック 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. |
| トピック 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. |
| トピック 3 | • XSLT: This section of the exam measures the skills of Data Integration Developers and covers the use of Extensible Stylesheet Language Transformations (XSLT) in Workday integrations. It focuses on transforming XML data structures, applying conditional logic, and formatting output for various integration use cases such as APIs and external file delivery. |

## Workday Pro Integrations Certification Exam 認定 Workday-Pro-Integrations 試験問題 (Q68-Q73):

**質問 # 68**
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 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_Reference/wd:ID[@wd:type='Job_Profile_ID']
- B. wd:Job_Profile_Data/wd:Job_Code
- C. wd:Job_Profile_Data[@wd:Job_Code]
- D. wd:Job_Profile/wd:Job_Profile_Data/wd:Job_Code

**正解：B**

解説：
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 Reference
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.

**質問 # 69**
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.
- C.
- D.

**正解：D**

解説：
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.

**質問 # 70**
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: Worker; no RBOs
- B. PBO: Worker, RBO: Dependent
- C. PBO: Dependent, RBO: Worker
- D. PBO: Dependent, no RBOs

**正解：B**

解説：
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.

**質問 # 71**
What option for an outbound EIB uses a Workday-delivered transformation to output a format other than Workday XML?

- A. XSLT Attachment Transformation
- B. Custom Transformation
- C. Alternate Output Format
- D. Custom Report Transformation

**正解：C**

解説：

Overview
For an outbound Enterprise Interface Builder (EIB) in Workday, the option that uses a Workday-delivered transformation to output a format other than Workday XML isAlternate Output Format. This allows you to select formats like CSV, which Workday handles without needing custom coding.
How It Works
When setting up an outbound EIB, you can use a custom report as the data source. By choosing an alternate output format, such as CSV, Workday automatically transforms the data into that format. This is surprising because it simplifies the process, requiring no additional user effort for transformation.
Why Not the Others?
* XSL Attachment Transformation (B): This requires you to provide your own XSL file, making it a custom transformation, not delivered by Workday.
* Custom Transformation (C): This is clearly user-defined, not Workday-delivered.
* Custom Report Transformation (D): This also involves user customization, typically through XSL, and isn't a pre-built Workday option.
Comprehensive Analysis
This section provides a detailed examination of Workday's Enterprise Interface Builder (EIB) transformation options, focusing on outbound integrations and the specific question of identifying the option that uses a Workday-delivered transformation to output a format other than Workday XML. We will explore the functionality, configuration, and implications of each option, ensuring a thorough understanding based on available documentation and resources.
Understanding Workday EIB and Outbound Integrations
Workday EIB is a no-code, graphical interface tool designed for both inbound and outbound integrations, facilitating the exchange of data between Workday and external systems. For outbound EIBs, the process involves extracting data from Workday (typically via a custom report) and delivering itto an external endpoint, such as via SFTP, email, or other protocols. The integration process consists of three key steps: Get Data, Transform, and Deliver.
* Get Data: Specifies the data source, often a Workday custom report, which must be web service- enabled for EIB use.
* Transform: Optionally transforms the data into a format suitable for the external system, using various transformation types.
* Deliver: Defines the method and destination for sending the transformed data.
The question focuses on the Transform step, seeking an option that uses a Workday-delivered transformation to output a format other than Workday XML, which is typically the default format for Workday data exchanges.
Analyzing the Options
Let's evaluate each option provided in the question to determine which fits the criteria:
* Alternate Output Format (A)
* Description: This option is available when configuring the Get Data step, specifically when using a custom report as the data source. It allows selecting an alternate output format, such as CSV, Excel, or other supported formats, instead of the default Workday XML.
* Functionality: When selected, Workday handles the transformation of the report data into the chosen format. For example, setting the alternate output format to CSV means the EIB will deliver a CSV file, and this transformation is performed by Workday without requiring the user to define additional transformation logic.
* Workday-Delivered: Yes, as the transformation to the alternate format (e.g., CSV) is part of Workday's report generation capabilities, not requiring custom coding or user-provided files.
* Output Format Other Than Workday XML: Yes, formats like CSV are distinct from Workday XML, fulfilling the requirement. From resources likeWorkday HCM features | Workday EIB, it's noted that custom reports can use CSV as an alternate output format, and this is managed by Workday, supporting our conclusion.
* XSL Attachment Transformation (B)
* Description: This involves attaching an XSL (Extensible Stylesheet Language) file to the EIB for transforming the data, typically from XML to another format like CSV or a custom structure.
* Functionality: The user must create or provide the XSL file, which defines how the data is transformed. This is used in the Transform step to manipulate the XML output from the Get Data step.
* Workday-Delivered: No, as the XSL file is custom-created by the user. Resources liker/workday on Reddit: EIB xslt Transformationdiscuss users working on XSL transformations, indicating they are user-defined, not pre-built by Workday.
* Output Format Other Than Workday XML: Yes, it can output formats like CSV, but it's not Workday-delivered, so it doesn't meet the criteria.
* Custom Transformation (C)
* Description: This option allows users to define their own transformation logic, often through scripting or other custom methods, to convert the data into the desired format.
* Functionality: It is a user-defined transformation, typically used for complex scenarios where standard options are insufficient.
* Workday-Delivered: No, as it explicitly states "custom," meaning it's not provided by Workday.
* Output Format Other Than Workday XML: Yes, it can output various formats, but again, it's not Workday-delivered, so it doesn't fit.
* Custom Report Transformation (D)
* Description: This might refer to transformations specifically related to custom reports, potentially involving user-defined logic to

manipulate the report data.
* Functionality: From resources likeSpark Databox - using custom report transformation, it involves using custom XSL transformations, indicating user involvement. It seems to be a subset of custom transformations, focusing on report data.
* Workday-Delivered: No, as it involves custom XSL, which is user-provided, not pre-built by Workday.
* Output Format Other Than Workday XML: Yes, it can output formats like pipe-delimited files, but it's not Workday-delivered, so it doesn't meet the criteria.

**質問 # 72**
Refer to the following XML to answer the question below.
You are an integration developer and need to write XSLT to transform the output of an EIB which is making a request to the Get Job Profiles web service operation. The root template of your XSLT matches on the <wd:
Get_Job_Profiles_Response> element. This root template then applies templates against <wd:Job_Profile>.
What XPath syntax would be used to select the value of the ID element which has a wd:type attribute named Job_Profile_ID when the <xsl:value-of> element is placed within the template which matches on <wd:
Job_Profile>?

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

**正解：A**

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

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

**質問 # 73**
.....

多くの人々は高い難度のIT認証試験に合格するのは専門の知識が必要だと思います。それは確かにそうですが、その知識を身につけることは難しくないといわれています。IT業界ではさらに強くなるために強い専門知識が必要です。Workday Workday-Pro-Integrations認証試験に合格することが簡単ではなくて、Workday Workday-Pro-Integrations証明書は君にとってはIT業界に入るの一つの手づるになるかもしれません。しかし必ずしも大量の時間とエネルギーで復習しなくて、弊社が丹精にできあがった問題集を使って、試験なんて問題ではありません。

**Workday-Pro-Integrations無料試験**: https://www.jpexam.com/Workday-Pro-Integrations_exam.html

- 素晴らしいWorkday Workday-Pro-Integrations試験合格攻略 インタラクティブテストエンジンを使用して - 公認されたWorkday-Pro-Integrations無料試験 □ Open Webサイト✔ www.passtest.jp □✔□検索「 Workday-Pro-Integrations 」無料ダウンロードWorkday-Pro-Integrations試験問題
- Workday-Pro-Integrations日本語関連対策 □ Workday-Pro-Integrations対応資料 □ Workday-Pro-Integrations日本語関連対策 □ 検索するだけで➡ www.goshiken.com □□□から➤ Workday-Pro-Integrations □を無料でダウンロードWorkday-Pro-Integrations試験時間
- 最新-完璧なWorkday-Pro-Integrations試験合格攻略試験-試験の準備方法Workday-Pro-Integrations無料試験 □「 www.passtest.jp 」を入力して《 Workday-Pro-Integrations 》を検索し、無料でダウンロードしてください Workday-Pro-Integrations復習テキスト
- ユニークなWorkday-Pro-Integrations試験合格攻略 - 合格スムーズWorkday-Pro-Integrations無料試験 | 認定するWorkday-Pro-Integrations受験料 Workday Pro Integrations Certification Exam □ ➽ www.goshiken.com □は、「 Workday-Pro-Integrations 」を無料でダウンロードするのに最適なサイトですWorkday-Pro-Integrations PDF問題サンプル
- 効果的なWorkday-Pro-Integrations試験合格攻略 - 合格スムーズWorkday-Pro-Integrations無料試験 | 最新のWorkday-Pro-Integrations受験料 □ ウェブサイト【 www.xhs1991.com 】から□ Workday-Pro-Integrations □を開いて検索し、無料でダウンロードしてくださいWorkday-Pro-Integrations復習テキスト
- ハイパスレートのWorkday-Pro-Integrations試験合格攻略一回合格-効率的なWorkday-Pro-Integrations無料試験 □“www.goshiken.com”サイトで➡ Workday-Pro-Integrations □□□の最新問題が使えるWorkday-Pro-Integrations復習時間
- Workday-Pro-Integrations試験の準備方法 | 最高のWorkday-Pro-Integrations試験合格攻略試験 | 検証するWorkday Pro Integrations Certification Exam無料試験 □【 Workday-Pro-Integrations 】を無料でダウンロード【 www.it-passports.com 】で検索するだけWorkday-Pro-Integrations PDF問題サンプル
- Workday-Pro-Integrations資格模擬 □ Workday-Pro-Integrations資格模擬 □ Workday-Pro-Integrations真実試験 □“www.goshiken.com”にて限定無料の⇒ Workday-Pro-Integrations ⇐問題集をダウンロードせよWorkday-Pro-Integrations資格認定試験
- Workday-Pro-Integrations資格認定試験 Ⓜ Workday-Pro-Integrations出題範囲 □ Workday-Pro-Integrations勉強資料 □「 www.shikenpass.com 」に移動し、【 Workday-Pro-Integrations 】を検索して、無料でダウンロード可能な試験資料を探しますWorkday-Pro-Integrations日本語関連対策
- ハイパスレートのWorkday-Pro-Integrations試験合格攻略一回合格-効率的なWorkday-Pro-Integrations無料試験 □➤ Workday-Pro-Integrations □の試験問題は[ www.goshiken.com ]で無料配信中Workday-Pro-Integrations合格体験記
- 最新-完璧なWorkday-Pro-Integrations試験合格攻略試験-試験の準備方法Workday-Pro-Integrations無料試験 □《 www.mogiexam.com 》サイトにて最新□ Workday-Pro-Integrations □問題集をダウンロードWorkday-Pro-Integrations問題数
- nanadxmh690991.bloggerchest.com, rishixlsu229467.bloggerchest.com, deaconvrdw852301.wikipublicity.com, darrenevmn147938.myparisblog.com, www.stes.tyc.edu.tw, freeurldirectory.com, janatody323178.thenerdsblog.com, junaidexie972906.wikiconverse.com, thedeepdirectory.com, adsbookmark.com, Disposable vapes

BONUS！！！ Jpexam Workday-Pro-Integrationsダンプの一部を無料でダウンロード：https://drive.google.com/open?id=16ohf4UdFge-jfkc5QUZh_AgZKcNjkhce