# Workday

# Workday-Pro-Integrations

## Workday Pro Integrations Certification Exam

# Latest Version: 8.2

## Question: 1

You have been asked to refine a report which outputs one row per worker and is being used in an integration that sends worker data to one of your third-party systems. The integration should only send workers who have been hired in the last 30 days. Where in the custom report definition can you specify a condition that would include only workers who have been hired in the last 30 days?

A. Subfilter
B. Output
C. Columns
D. Filter

**Answer: D**

Explanation:
In Workday, when refining a custom report to include specific conditions such as limiting the output to workers hired in the last 30 days, the appropriate place to specify this condition is within the Filter tab of the custom report definition. The Filter tab allows you to define criteria that determine which instances of the primary business object (in this case, "Worker") are included in the report output. This is critical for integrations, as the filtered data ensures that only relevant records are sent to the third-party system.
The requirement here is to restrict the report to workers hired within the last 30 days. In Workday reporting, this can be achieved by adding a filter condition on the "Hire Date" field of the Worker business object. Specifically, you would configure the filter to compare the "Hire Date" against a dynamic date range, such as "Current Date minus 30 days" to "Current Date." This ensures the report dynamically adjusts to include only workers hired in the last 30 days each time it runs, which aligns with the needs of an integration sending real-time data to a third-party system.
Here's why the other options are incorrect:
A . Subfilter: Subfilters in Workday are used to further refine data within a related business object or a subset of data already filtered by the primary filter. They are not the primary mechanism for applying a condition to the main dataset (e.g., all workers). For this scenario, a subfilter would be unnecessary since the condition applies directly to the Worker business object, not a related object.
B . Output: The Output section of a custom report definition controls how the report is displayed or delivered (e.g., file format, scheduling), not the data selection criteria. It does not allow for specifying conditions like hire date ranges.
C . Columns: The Columns tab defines which fields are displayed in the report output (e.g., Worker ID, Name, Hire Date). While you can add the "Hire Date" field here for visibility, it does not control which workers are included in the report—that is the role of the Filter tab.
To implement this in practice:
In the custom report definition, go to the Filter tab.
Add a new filter condition.
Select the "Hire Date" field from the Worker business object.

Set the operator to "in the range" and define the range as "Current Date - 30 days" to "Current Date" (using dynamic date functions available in Workday).
Save and test the report to ensure it returns only workers hired within the last 30 days.
This filtered report can then be enabled as a web service (via the Advanced tab) or used in an Enterprise Interface Builder (EIB) or Workday Studio integration to send the data to the third-party system, meeting the integration requirement.
Reference from Workday Pro Integrations Study Guide:
Workday Report Writer Fundamentals: Section on "Creating and Managing Filters" explains how filters are used to limit report data based on specific conditions, such as date ranges.
Integration System Fundamentals: Discusses how custom reports serve as data sources for integrations and the importance of filters in defining the dataset.
Core Connectors & Document Transformation: Highlights the use of filtered custom reports in outbound integrations to third-party systems.

## Question: 2

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

**Answer: B**

Explanation:
In Workday reporting, selecting the appropriate Primary Business Object (PBO) and Related 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 multiinstance 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 with Worker as the PBO.
Add a filter for the supervisory organization (specified at runtime) on the Worker PBO.
Add Dependent as an RBO to access Name and Age fields.
Include columns from Worker (e.g., Employee, Benefit Plans) and Dependent (e.g., Name, Age).
Reference 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.

## Question: 3

You have a population of workers who have put multiple names in their Legal Name - First Name Workday delivered field. Your third-party vendor only accepts one-word first names. For workers that have included a middle name, the first and middle names are separated by a single space. You have been asked to implement the following logic:
* Extract the value before the single space from the Legal Name - First Name Workday delivered field.
* Count the number of characters in the extracted value.
* Identify if the number of characters is greater than.
* If the count of characters is greater than 0, use the extracted value. Otherwise, use the Legal Name - First Name Workday delivered field.
What functions are needed to achieve the end goal?

A. Extract Single Instance, Text Length, Numeric Constant, True/False Condition

B. Text Constant, Substring Text, Arithmetic Calculation, Evaluate Expression
C. Format Text, Convert Text to Number, True/False Condition, Evaluate Expression
D. Substring Text, Text Length, True/False Condition, Evaluate Expression

**Answer: D**

Explanation:
The task involves processing the "Legal Name - First Name" field in Workday to meet a third-party vendor's requirement of accepting only one-word first names. For workers with multiple names (e.g., "John Paul"), separated by a single space, the logic must:
Extract the value before the space (e.g., "John" from "John Paul").
Count the characters in the extracted value.
Check if the character count is greater than 0.
Use the extracted value if the count is greater than 0; otherwise, use the original "Legal Name - First Name" field.
This logic is typically implemented in Workday using calculated fields within a custom report or integration (e.g., EIB or Studio). Let's break down the required functions:
Substring Text:This function is needed to extract the portion of the "Legal Name - First Name" field before the space. In Workday, the Substring Text function allows you to specify a starting position (e.g., 1)
and extract text up to a delimiter (e.g., a space). For example, Substring Text("John Paul", 1, Index of " ") would return "John."
Text Length:After extracting the substring (e.g., "John"), the logic requires counting its characters to ensure it's valid. The Text Length function returns the number of characters in a text string (e.g., Text Length("John") = 4). This is critical for the condition check.
True/False Condition:The logic involves a conditional check: "Is the number of characters greater than 0?" The True/False Condition function evaluates this (e.g., Text Length(extracted value) > 0), returning True if the extracted value exists and False if it's empty (e.g., if no space exists or extraction fails).
Evaluate Expression:This function implements the if-then-else logic: if the character count is greater than
0, use the extracted value (e.g., "John"); otherwise, use the original "Legal Name - First Name" field (e.g., "John Paul"). Evaluate Expression combines the True/False Condition with the output values.
Option Analysis:
A . Extract Single Instance, Text Length, Numeric Constant, True/False Condition: Incorrect. Extract Single
Instance is used for multi-instance fields (e.g., selecting one dependent), not text parsing. Numeric Constant isn't needed here, as no fixed number is involved.
B . Text Constant, Substring Text, Arithmetic Calculation, Evaluate Expression: Incorrect. Text Constant provides a fixed string (e.g., "abc"), not dynamic extraction. Arithmetic Calculation isn't required, as this is a text length check, not a numeric operation beyond comparison.
C . Format Text, Convert Text to Number, True/False Condition, Evaluate Expression: Incorrect. Format Text adjusts text appearance (e.g., capitalization), not extraction. Convert Text to Number isn't needed, as Text Length already returns a number.
D . Substring Text, Text Length, True/False Condition, Evaluate Expression: Correct. These functions align perfectly with the requirements: extract the first name, count its length, check the condition, and choose the output.
Implementation:

Create a calculated field using Substring Text to extract text before the space.
Use Text Length to count characters in the extracted value.
Use True/False Condition to check if the length > 0.
Use Evaluate Expression to return the extracted value or the original field based on the condition.
Reference from Workday Pro Integrations Study Guide:
Workday Calculated Fields: Section on "Text Functions" details Substring Text and Text Length usage.
Integration System Fundamentals: Explains how calculated fields with conditions (True/False, Evaluate Expression) transform data for third-party systems.
Core Connectors & Document Transformation: Highlights text manipulation for outbound integration requirements.

## Question: 4

Your manager has asked for a value on their dashboard for how many days away the birthdays are of their direct reports. The format of the output should be [Worker's Name]'s birthday is in [X] days, where you must calculate the number of days until a Worker's next birthday. An example output is "Logan McNeil's birthday is in 103 days."
Which calculated field functions do you need to accomplish this?

A. Format Date, Increment or Decrement Date, Extract Single Instance, Format Text
B. Build Date, Format Date, Extract Single Instance, Format Text
C. Date Difference, Format Number, Text Constant, Concatenate Text
D. Increment or Decrement Date, Format Number, Text Constant, Concatenate Text

**Answer: C**

Explanation:
The requirement is to create a calculated field for a dashboard that displays a worker's name and the number of days until their next birthday in the format "[Worker's Name]'s birthday is in [X] days" (e.g., "Logan McNeil's birthday is in 103 days"). This involves calculating the difference between today's date and the worker's next birthday, then formatting the output as a text string. Let's break down the necessary functions:
Date Difference:To calculate the number of days until the worker's next birthday, you need to determine the difference between the current date and the worker's birthdate in the current or next year
(whichever is upcoming). The Date Difference function calculates the number of days between two dates. In this case:
Use the worker's "Date of Birth" field (from the Worker business object).
Adjust the year of the birthdate to the current year or next year (if the birthday has already passed this year) using additional logic.
Calculate the difference from today's date to this adjusted birthday date. For example, if today is February 21, 2025, and Logan's birthday is June 4 (adjusted to June 4, 2025), Date Difference returns 103 days.
Format Number:The result of Date Difference is a numeric value (e.g., 103). To ensure it displays cleanly in the output string (without decimals or unnecessary formatting), Format Number can be used to convert it to a simple integer string (e.g., "103").

Text Constant:To build the output string, static text like "'s birthday is in " and " days" is needed. The Text Constant function provides fixed text values to include in the final concatenated result.
Concatenate Text:The final step is to combine the worker's name (e.g., "Logan McNeil"), the static text, and the calculated days into one string. Concatenate Text merges multiple text values into a single output, such as "Logan McNeil" + "'s birthday is in " + "103" + " days".
Option Analysis:
A . Format Date, Increment or Decrement Date, Extract Single Instance, Format Text: Incorrect. Format Date converts dates to strings but doesn't calculate differences. Increment or Decrement Date adjusts dates but isn't suited for finding days until a future event. Extract Single Instance is for multi-instance fields, not relevant here. Format Text adjusts text appearance, not numeric calculations.
B . Build Date, Format Date, Extract Single Instance, Format Text: Incorrect. Build Date creates a date from components, useful for setting the next birthday, but lacks the difference calculation. Format Date and Extract Single Instance don't apply to the core need.
C . Date Difference, Format Number, Text Constant, Concatenate Text: Correct. These functions cover calculating the days, formatting the number, adding static text, and building the final string.
D. Increment or Decrement Date, Format Number, Text Constant, Concatenate Text: Incorrect. Increment
or Decrement Date can't directly calculate days to a future birthday without additional complexity; Date Difference is more appropriate.
Implementation:
Use Date Difference to calculate days from today to the next birthday (adjusting the year dynamically with additional logic if needed).
Apply Format Number to ensure the result is a clean integer.
Use Text Constant for static text ("'s birthday is in " and " days").
Use Concatenate Text to combine Worker Name, static text, and the formatted number.
Reference from Workday Pro Integrations Study Guide:
Workday Calculated Fields: Section on "Date Functions" explains Date Difference for calculating time spans.
Report Writer Fundamentals: Covers Concatenate Text and Text Constant for string building in reports.

## Question: 5

You need to filter a custom report to only show workers that have been terminated after a userprompted date.
How do you combine conditions in the filter to meet this requirement?

A. Worker Status is equal to the value "Terminated" OR Termination Date is greater than a value retrieved from a prompt
B. Worker Status is equal to the value retrieved from a prompt AND Termination Date is less than a value retrieved from a prompt.
C. Worker Status is equal to the value retrieved from a prompt OR Termination Date is equal to a value retrieved from a prompt.
D. Worker Status is equal to the value "Terminated" AND Termination Date is greater than a value retrieved from a prompt.

**Answer: D**

Explanation:
The requirement is to filter a custom report to show only workers terminated after a user-prompted date. In Workday, filters are defined in the Filter tab of the custom report definition, and conditions can be combined using AND/OR logic to refine the dataset. Let's analyze the requirement and options:
Key Conditions:
Workers must be terminated, so the "Worker Status" field must equal "Terminated."
The termination must occur after a user-specified date, so the "Termination Date" must be greater than the prompted value.
Both conditions must be true for a worker to appear in the report, requiring an AND combination.
Option Analysis:
A . Worker Status is equal to the value "Terminated" OR Termination Date is greater than a value retrieved from a prompt: Incorrect. Using OR means the report would include workers who are terminated (regardless of date) OR workers with a termination date after the prompt (even if not terminated), which doesn't meet the strict requirement of terminated workers after a specific date.
B . Worker Status is equal to the value retrieved from a prompt AND Termination Date is less than a value retrieved from a prompt: Incorrect. Worker Status shouldn't be a prompted value (it's fixed as "Terminated"), and "less than" would show terminations before the date, not after.
C . Worker Status is equal to the value retrieved from a prompt OR Termination Date is equal to a value retrieved from a prompt: Incorrect. Worker Status shouldn't be prompted, and "equal to" limits the filter to exact matches, not "after" the date. OR logic also broadens the scope incorrectly.
D . Worker Status is equal to the value "Terminated" AND Termination Date is greater than a value retrieved from a prompt: Correct. This ensures workers are terminated (fixed value) AND their termination date is after the user-entered date, precisely meeting the requirement.
Implementation:
In the custom report's Filter tab, add two conditions:
Field: Worker Status, Operator: equals, Value: "Terminated".
Field: Termination Date, Operator: greater than, Value: Prompt for Date (configured as a report prompt).
Set the logical operator between conditions to AND.
Test with a sample date to verify only terminated workers after that date appear.
Reference from Workday Pro Integrations Study Guide:
Workday Report Writer Fundamentals: Section on "Creating and Managing Filters" details combining conditions with AND/OR logic and using prompts.
Integration System Fundamentals: Notes how filtered reports support integration data sources with dynamic user inputs.

# Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

**15 USD Discount Coupon Code:**

**G74JA8UF**

# FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer