DocProcessor:

Filter Conditions Syntax Reference

Filter Conditions Syntax and Construction Guide

Revised: July 5th, 2019

Table of Contents

Overview

Introduction

Many *DocProcessor* fetch commands provide flexible data filtering capability via parameters. You can specify "Filter Condition" parameters in the fetch commands so that the *DocProcessor* can return the records that satisfy the filter criteria.

What is this document about?

This document describes the syntax and usage rules for creating the filter conditions within the "condition" type of parameters of fetch commands.

Who should read this document?

The information in this document will help you customize the report and document templates that ship with *TopTeam repository*, or help you create new reports and document templates.

Specifying the Filter Text for Condition Parameters

A Filter Condition parameter consists of three parts

Field

This is the field caption of the *TopTeam/Visual Use Case* field on which you want to filter. The field caption must be enclosed in double quotes. e.g. "Priority".

Operator

This is the comparison operator.

Value

This is the value that you want to compare the field against. The value must be enclosed in double quotes. e.g. “Baselined”.

Example

Example of a Filter Parameter

\Fetch_Use_Cases_By_Condition(‘<<Filter Condition>>’, ‘<<Sort Order>>’)

This command fetches the *Use Case* records that satisfy the specified filter condition (optional).

Example

\Fetch_Use_Cases_By_Condition(‘ “Priority” = “Must Have” ‘,’’)

Field Types and Filter Condition Examples

Number/Decimal Fields

Examples

“Est Cost” = “10.5”

“Est Cost” <> “0.7”

“Est Cost” < “20”

“Est Cost” > “0.7”

“Est Effort Hrs” IS NULL

“Est Effort Hrs” IS NOT NULL

Boolean (True/False) Fields

Examples

“Locked” = “True”

“In Scope” = “False”

Text Fields

Examples

“Name” = “BLU-RAY”

“Name” <> “BLU-RAY”

“Name” STARTING WITH “BLU-RAY”

“Name” ENDING WITH “BLU-RAY”

“Name” CONTAINING “BLU-RAY”

“Title” NOT CONTAINING “BLU-RAY”

“Name” NOT STARTING WITH “BLU-RAY”

“Name” NOT ENDING WITH “BLU-RAY”

“Name” IS NULL

“Name” IS NOT NULL

List of Values Fields (drop-down list)

Examples

“Priority” = “Must Have”

“State” <> “Open”

“State” = “Baselined”

“Priority” IN LIST “High, Very High”

“Priority” NOT IN LIST “Low, Very Low”

“Level” IS NULL

“Complexity” IS NOT NULL

Multi-Value Fields

Examples

“Country” IN LIST “United States, Canada”

“Country” IS NULL

“Country” IS NOT NULL

Date Fields

NOTE: MM-DD-YYYY is the ***ONLY*** supported date format for parameter values.

Examples

“Upd Dt” = “09-18-2008”

“Crt dt” <> “10-15-2008”

“Asgnd Dt” IS NULL

“Crt dt” > “12-20-2008”

“Crt dt” < “12-30-2008”

“Crt dt” BETWEEN “09-18-2007 9-28-2008”

“Crt dt” FALLS IN LAST N DAYS “7”

“Crt dt” FALLS IN NEXT N DAYS “10”

“Crt dt” FALLS IN NEXT N MONTHS “2”

“Crt dt” FALLS BEFORE TODAY

“Crt dt” IS TODAY

“Crt dt” IS TOMORROW

“Crt dt” IS YESTERDAY

Joining Multiple Conditions with Boolean Operators

Examples

“Priority” = “Very High” AND “State” = “Open”

“Owner” = “User1” OR “Crt by” = “Me”

“Severity” = “High” AND (“Owner” = “User4” OR “Owner” = “Me”)

(“Priority” = “Very High”) OR (“State” = “All Open” AND “Owner” = “Me”)

Relational Filters

The Relational Filter has three parts

1. **Relational Filter**
2. **Record Type (Optional)**
3. **Satisfying given conditions (Optional)**

Example

\Fetch_Use_Cases_By_Condition(‘ {Having Upstream Trace Links (“Business Rules”)}’ , ‘’)

Examples of a Relational Filter

\Fetch_Use_Cases_By_Condition(‘ {Having open Tracking items} ’,‘’)

This will return *Use Case* records that have linked tracking items which are in any one of the “open” states.

\Fetch_Use_Cases_By_Condition(‘ {Having open Tracking items with (“PR”, “ISS”)} ’,‘’)

This will return *Use Case* records that have linked *Problem Reports* or *Issues* which are in any one of the “open” states.

\Fetch_Use_Cases_By_Condition(‘ {Having open Tracking items [“Priority” = “Very High” AND “Severity” = “Critical”]} ’,‘’)

This will return *Use Case* records that have linked Tracking Items that have a *Very High Priority*, their *Severity* is *Critical*, and are in any one of the “open” states.

\Fetch_Use_Cases_By_Condition(‘ {Having open Tracking items with (“PR”, “ISS”) [“Priority” = “Very High” AND “Severity” = “Critical”] } ’,‘’)

This will return *Use Case* records that have linked *Problem Reports* or *Issues* that have a *Very High Priority*, their *Severity* is *Critical*, and are in any one of the "open" states.

\Fetch_Use_Cases_By_Condition(' {Having Upstream Trace Links ("BRULE")} AND {Having Attachments ["Person" = "Me"]} ','')

This will return *Use Case* records that have upstream (incoming or reverse) trace links from "Business Rules (BRULE)" and have attachments that were added by the person running the query.

\Fetch_Use_Cases_By_Condition(' {Having Upstream Trace Links ("Business Rules")} And {Having Attachments ["Person" = "Me"]} ','')

This is an alternate format of the previous example.

\Fetch_Use_Cases_By_Condition(' "Priority" = "High" AND {Having open Tracking items with ("PR", "ISS") ["State" = "<<All Open>>" AND "Severity" = "High"] } ','')

Filter Condition and Multiple Relational Filters

\Fetch_Use_Cases_By_Condition(' "Name" containing "Rent" AND { having downstream trace links ("Business Requirements") ["Priority" = "Very High"] } AND { having attachments ["Person" = "Me"] } ','')

\Fetch_Use_Cases_By_Condition(' "Name" containing "Rent" And { having downstream trace links ("BREQ") ["Priority" = "Very High"] } And { having attachments ["Person" = "Me"] } ','')

\Fetch_Use_Cases_By_Condition(' ("Level" <> "Summary " OR "Size" <> "Large" OR "Locked" = "True") AND {having open tracking items with ("PR", "ISS") ["Priority" = "Very High" AND "Severity" = "High"] } ','')

\Fetch_Use_Cases_By_Condition(‘ NOT { Having Upstream Trace Links (“BRULE”)} ’,‘’)\

Relational Filters List

Relational Filters that can be used with Repository Object and Tracking Item Fetch commands

{Having Attachments}

This will return records that have one or more attachments.

{Having Audit Log}

This will return records for which audit log exists. All records have an audit log and therefore you may want to add some filter conditions, else all records will be fetched.

{Having Branches}

This will return records that have branches.

{Having Comments}

This will return records that have one or more comments.

{Having Discussions}

This will return records that have one or more discussion records.

{Having Todos}

This will return records that have Todos.

{Having Variants}

This will return records that have variants.

{Having Versions}

This will return records for which version history shows records. All records have version history and therefore you may want to add some filter conditions, else all records will be fetched.

{Having Workflow History}

This will return records for which workflow history exists.

{Included in any Collection}

This will return records that are included in any Collection.

{Included in any Release}

This will return records that are included as Deliverables in at least one Release.

{Not included in any Collection}

This will return records that are not included in any Collection.

{Not included in any Release}

This will return records that are not included as deliverables in any Release.

{Which I am following}

This will return records that you are following.

Relational Filters that can be used with Repository Object Fetch commands

{Having Upstream Trace Links}

This will return records that have at least one upstream (incoming or reverse) link: Traces From, Impacted By, etc.

Sample DocProcessor Commands:

1. Fetch records which have upstream links of type "included in" with Review Package as the name "pkg 1"

   { Having Upstream System Links with ("Included In") [{ Repository Objects ("RPG") ["Name" = "pkg 1"] }] }

2. Fetch records which have upstream links of type "included in", and that link was created by me, with Review Package as the name "pkg 1"

   { Having Upstream System Links with ("Included In") ["Created By" = "Me" and { Repository Objects ("RPG") ["Name" = "pkg 1"] }] }

{Not Having Upstream Trace Links}

This will return records that have NO upstream (incoming or reverse) links: Traces From, Impacted By, etc.

{Having Downstream Trace Links}

This will return records that have at least one downstream (outgoing or forward) link: Traces Into, Impacts, etc.

{Not Having Downstream Trace Links}

This will return records that NO downstream (outgoing or forward) links: Traces Into, Impacts, etc.

{Having Suspect Upstream Traces}

This will return records that have at least one suspect upstream (incoming or reverse) link: Traces From, Impacted By, etc.

{Having Suspect Downstream Traces}

This will return records that have at least one suspect downstream (outgoing or forward) link: Traces Into, Impacts, etc.

{Having Downstream Non Trace Links}

This will return records that have at least one downstream (outgoing or forward) link: Traces Into, Impacts, etc.

{Not Having Downstream Non Trace Links}

This will return records that have NO downstream (outgoing or forward) links: Traces Into, Impacts, etc.

{Having Traced Test Cases}

This will return records that Trace Into one or more Test Case records.

{Having Linked Tracking Items}

This will return records that are linked to Tracking Items/Issues: Change Requests, Defects, Issues, etc.

{Not Having Upstream Links}

This will return records that have NO upstream (incoming or reverse) Traces Into links i.e. nothing traces into these records.

{Not Having Downstream Links}

This will return records that have NO downstream (outgoing or forward) Traces Into links i.e. these records do NOT Trace Into any other records.

{Included in any Baseline}

This will return records that are included in any Baseline.

{Having Test Results of Traced Test Cases}

This will return Test Results of Traced Test Cases.

{Having Linked Repository Objects}

This will return records that are linked to Repository Objects.

{Having Upstream System Links}

This will return records that have at least one upstream (incoming or reverse) System Link: Included In Use Case, Parent, Primary Actor For, etc.

{Not Having Upstream System Links}

This will return records that have NO upstream (incoming or reverse) System Links: Included In Use Case, Parent, Primary Actor For, etc.

{Having Downstream System Links}

This will return records that have at least one downstream (outgoing or forward) System Link: Includes, Children, Primary Actors, etc.

{Not Having Downstream System Links}

This will return records that have NO downstream (outgoing or forward) System Links: Includes, Children, Primary Actors, etc.

{Having Upstream Non Trace Links}

This will return records that have at least one upstream (incoming or reverse) Non Trace Links: Approved by, Author, Reviewer, etc.

{Not Having Upstream Non Trace Links}

This will return records that have NO upstream (incoming or reverse) Non Trace Links: Approved by, Author, Reviewer, etc.

Relational Filters that can be used with Use Case Fetch commands

{Having <<extends>> Relations with Use Cases}

This will return Use Cases that have at least one <<extends>> type of link with other Use Cases.

{Having <<include>> Relations with Use Cases}

This will return Use Cases that have at least one <<include>> type of link with other Use Cases.

Relational Filters that can be used with Requirements Fetch commands

{Having Child Requirements}

This will return requirements that have children requirements.

Relational Filters that can be used with Tracking Item Fetch commands

{Having Child Items}

This will return tracking items that have children Tracking Items.

{Having no Child Items}

This will return tracking items that have no children Tracking Items.

{Having Open Child Items}

This will return tracking items that have at least one child Tracking Item in an <<open>> state.

{Having Parent}

This will return tracking items that have parent Tracking Item.

{Linked to Repository Objects}

This will return tracking items that are linked to a Repository Object.

{Linked to Repository Objects I own}

This will return Tracking Items that are linked to a Repository Object where the owner of the repository object is the current user.

{Having Approval Requests}

This will return records that have Approval Request(s) sent in the past for any version (current or older).

{Having Approval Requests for Current Version}

This will return records that have Approval Request(s) sent in the past for its current version.

{Reported while Testing}

This will return Defects that are submitted while executing Test Runs.

{Having Test Results}

Can be used with Test Cases or Traced Test Cases fetch commands. This will return Test Results for Test Cases.

{Having Test Runs}

Can be used with Test Results fetch command. This will return Test Runs for Test Results.

{Having Test Sets}

Can be used with Test Results fetch command. This will return Test Sets for Test Results.

{Having Test Run Results}

This will return Test Results of Test Cases that are executed while running a Test Run.

{Having Test Results of Traced Test Cases}

This will return Test Results of Test Cases that are traced from requirements or any other type of records.

Filter Condition Syntax & Troubleshooting

You must have a SPACE character between Field, Operator, and Value.

Correct

“Priority” = “Must Have”

Incorrect

“Priority”=“Must Have”

This is incorrect because there is *NO* space character between the Field, Operator, and Value tokens.

The operator should be compatible with the Field Type

Correct

“Crt dt” IS TODAY

Incorrect

“Name” IS TODAY

This is incorrect because **“Name”** is a text field and **IS TODAY** is a date operator that is not compatible with a text field.

Multiple values can be used as values for List (Allowed Values) type fields only.

Correct

“Priority” IN LIST “High, Very High”

Multiple values must be separated by a comma.

The Field value is not needed with some operators

Correct

“Crt dt” IS TODAY

Incorrect

“Crt dt” = IS TODAY

This is incorrect because **“=”** and **“IS TODAY”** are both operators.

Two operators cannot be used in the same condition.

The following operators do not need field values:

BEFORE TODAY

IS TODAY

IS TOMORROW

IS YESTERDAY

IS NULL

IS NOT NULL

Special characters are not allowed in Field Captions

Correct

“Est Effort Hrs” > “10”

Incorrect

“Est Effort (Hrs)” > “10”

This is incorrect because special characters such as ‘ ( ) “ ” ‘ ’ { } [ ] ’ etc. are not allowed in the field names and field values**.** Replace the special characters in field names with a space character.

Two date values are separated by the SPACE character for the "Between" operator.

Correct

"Crt dt" Between "10-28-2007 10-30-2007"

Pseudo values for certain fields types

Correct

"State" = "<<All Open>>"

"State" = "All Closed"

"Crt by" = "<<Me>>"

"Crt by" = "Me"