DocProcessor:

Commands Reference

Revised: April 27th, 2020

This document is for informational purposes only. TECHNOSOLUTIONS MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

DocProcessor trademark is owned by TechnoSolutions Corporation.

Techno Solutions

Table of Contents

Introduction

Overview

DocProcessor™ is the document generation engine built into TopTeam platform.

Using DocProcessor you can develop custom report templates to generate documents and reports as per your needs.

What is this guide about?

This guide lists the commands available in the DocProcessor document generation engine that can be used for developing custom document and report templates.

Learning “How to build report templates”

While this document lists the commands available in DocProcessor you can learn about how to use these commands to build a workable template in the DocProcessor Report Template Customization Guide.

Command Syntax

Fetch Commands

Fetch commands get data from the application repository as *record sets*, also known as *data sets*. These record sets can then be iterated using the \scan\ and \endscan\ commands.

The Fetch commands that you use to get data from the repository also determine which fields are available for output into the document. You can view detailed information about the fields available for a specific Fetch command, in the corresponding section of this document.

Examples

\Fetch_Use_Cases_By_Condition(' "Priority" = "Very High" ', 'Priority, Name') \

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('$DEFAULT_REQUIREMENTS_DOCUMENT$', 'Crt by = Me','1') \

Fetch Command Parameters

<<ID Prefix>>	Specify the Record Type Prefix such as 'SMK', that appears in front of the identifier. For example, in the identifier SMK-2348, SMK is the ID Prefix and represents Screen Mockups.
<< Filter Name>>	Specify the Filter. The <<Filter Name>> specified here should be defined in the List/Tree editor for the current specified record type. If not specified, the command will fetch all records for the set Project or Baseline.
<< Sort Order>>	Specify one or more Field Names separated by commas along with ascending and descending indicators.
<< ID Prefix comma separated>>	Specify comma-separated Record Type ID Prefixes whose Records you want to fetch.

	For example, in the identifier BRULE-2348, BRULE is the ID Prefix and represents “Business Rules”.
<<Link Types comma separated>>	Specify comma-separated Link Type names you want to fetch.
<< Filter Condition>>	Specify conditions to fetch records based on certain criteria. You can use only the fields of the current specified record type in the filter condition. Command will fetch the records that satisfy the specified filter condition. If not specified, the command will fetch all records for the set Project or Baseline.

Case Insensitivity

Command names are not case sensitive.

Enclosing Backslash

All commands must be enclosed within starting and ending backslashes (\).

Example

\Fetch_Use_Cases_By_Condition(' “State” = “Approved” ')\

Open-Closed Parenthesis

You must have an Open-Closed Parenthesis ‘()’ after every command name. When there are no parameters to be specified, you can leave the space empty between the parenthesis.

Example

\Fetch_Use_Cases_By_Condition(' ')\

Commands Should Not be Split with a New Line or Enter

NOTE: There must not be a character (New Line or Enter) in the middle of a command.
On occasions, a command and its parameters may get very long and exceed the page width. In such cases, it is acceptable for the command to "word wrap" into the next line. However, you must not manually break the command into the next line using a "New Line" or "Enter" character.

Example

\Insert_Trace_Matrix_by_Condition('UC','Traced from', 'Rental Management',' "Priority" = "High" ', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

Command Parameters

All parameters including condition snippets must be enclosed within single quotes.

Example

\Fetch_Use_Cases('Approved Status', '')\

Multiple parameter values must be separated by commas.

Example

\Fetch_Use_Cases('', 'Priority, Name')\

If you have commands that accept multiple parameters and you want to skip one of these parameters, you must specify that parameter with an empty value. An empty value can be specified by using two successive quotes such as ('').

Example

\Fetch_Use_Cases('', 'Priority ')\

Parameters that contain single quotes must be enclosed in double quotes ("").

Example

\Set_Project("Sue's Project")\

System Field Tags

System Field Tags are system-defined tags that are used for inserting values such as Today's Date, Current User, etc. You can use these tags directly in the DocProcessor templates. These are not like field tags which can be used only after Fetch commands.

**\ FILE_NAME **	Generates the name of the output document that is being generated.
**\ PROJECT_NAME **	Generates the name of the Project which is set using the Set_Project command.
**\ PROJECT_PATH **	Generates the full path of the Project which is set using the Set_Project command.
**\ CURRENT_DATE **	Generates the system date on a user's computer. This is the date/time on the local computer from which the document is being generated.
**\ USER_NAME **	Generates a login name of the current TopTeam user who is generating the document.

System Variable Tags

Fetch_Large_Field_Data

This system variable allows you to start/stop fetching large data fields in Document Reports.

1. By default, Document Reports FETCH commands allow fetching large data fields.
2. When FETCH_LARGE_FIELD_DATA = True
   It allows FETCH commands to START fetching large data fields
3. When FETCH_LARGE_FIELD_DATA = False
   It allows FETCH commands to STOP fetching large data fields

\FETCH_LARGE_FIELD_DATA := True

Example

The following is an example to STOP fetching large data fields such as *Description*.

```
FETCH_LARGE_FIELD_DATA - OLD Value :: \ FETCH_LARGE_FIELD_DATA \
\FETCH_LARGE_FIELD_DATA := 'FALSE'\
FETCH_LARGE_FIELD_DATA - NEW Value :: \ FETCH_LARGE_FIELD_DATA \


\Fetch_Repository_Objects_by_condition('ACTR','','Name')\
\scan(a)\\if(!eof(a))\

\ a : Name\ [\ a : Id\]

\Insert_Permalink( a: ID )\
\if (Is_Field_Non_Empty(a : Description))\

Description

\Insert_Rich_Text_Using_Word(a : Description)\
\endif\
\endif\
\endscan\
```

Set and Clear Context Commands

The Set commands set the context under which the subsequent Fetch commands operate. For example, when you use the Set_Project() command, all subsequent Fetch commands will fetch Records in that Project. Another example: if you use the Set_Baseline() command, all subsequent Fetch commands will fetch only those Records that are included in that Baseline.

Set_Project

This sets the context of the Document to a specified Project. Subsequent Fetch commands will fetch Records from this Project.

NOTE: If you wish to fetch Records for more than one Project in the same Document, you can specify multiple "Set_Project()" commands and place the Fetch commands in between the Set_Project() commands.

\Set_Project('<<Project Path>>')

OR

\Set_Project('<<Project Name>>')

Parameters

<<Project Path>>	Mandatory	Specify the complete Project path (This is required when specifying sub-Projects or child Projects).
<<Project Name>>	Mandatory	Specify the Project name (This works only for root or top level Projects). Alternatively, you can specify a special parameter (as explained below).

Examples

\Set_Project('Video Rental System')\

\Set_Project('Video Rental System\Reports')\

This is an example of a full Project path that starts from the root/top level Project.

Example

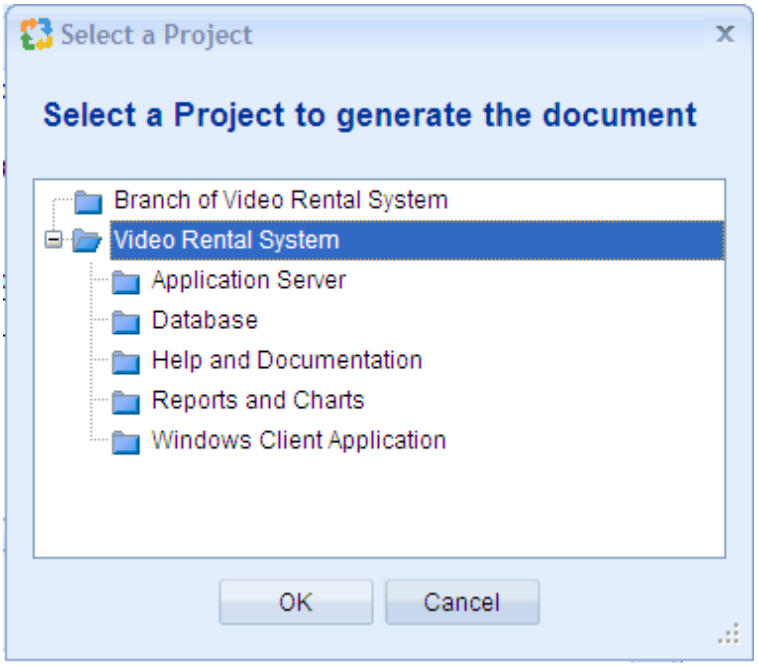
\Set_Project('$CURRENT_PROJECT$')\

This sets the context of the document to the current Project within the application. This is the most commonly used command.

Example

\Set_Project('$PROMPT_PROJECT$')\

This command prompts the user to select a Project when the document is generated. This command displays a Project selection window at runtime.



Example

\Set_Project('$PARENT_OF_CURRENT_PROJECT$')\

If you want to fetch Records from the parent Project, this command sets the context of the Document to the parent of the current Project within the application.

Example

\Set_Project('$ROOT_OF_CURRENT_PROJECT$')\

This sets the context of the document to the root or top-level of the current Project, within the application.

Set_Project_By_ID

This sets the context of the Document to the specified Project ID. Subsequent Fetch commands will fetch Records for this Project ID.

\Set_Project_By_ID('<<Project ID>>')

Parameter

<<Project ID>>	Mandatory	Specify the identifier (ID) of the Project.

Example

\Set_Project_By_ID('PRJ-1902')\

Clear_Project

This command clears the previous Set_Project context. Use this command to clear a previous Set_Project command.

\Clear_Project ()

Example

\Clear_Project()\

Set_Baseline

Compatibility: Desktop App Version 3.35 and above.

This sets the context of the Document to a specified Baseline. Subsequent Fetch commands will only fetch Records that are included in this Baseline.

NOTE: The Baseline specified as a parameter to this command MUST exist in the Project that was set previously in the template using the Set_Project() command.

\Set_Baseline('<<Baseline Name>>')

Parameter

<<Baseline Name>>	Mandatory	Specify the name of the Baseline or specify a special parameter (explained below).

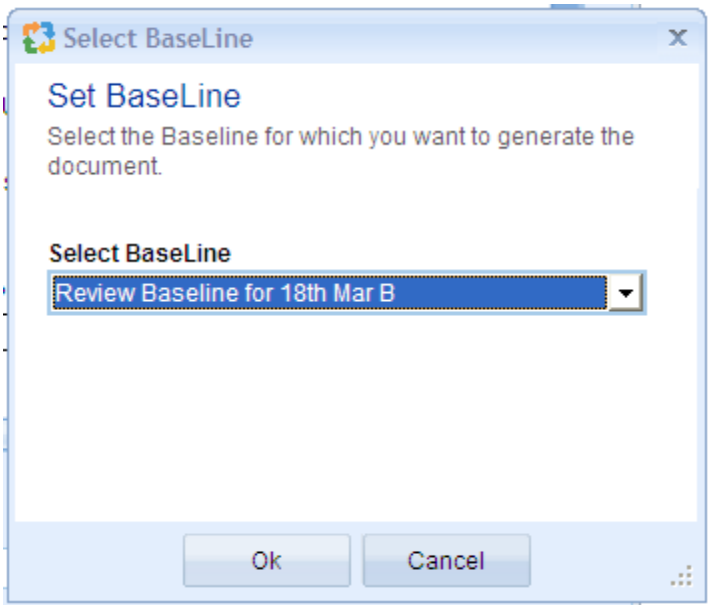
Examples

\Set_Baseline('Baseline1 on 2nd feb')\
\Set_Baseline ('$PROMPT_BASELINE$')\

Example

\Set_Baseline ('$PROMPT_BASELINE$')\

This command prompts the user to select a Baseline when the Document is generated. This command displays a Baseline selection window at runtime.



Clear_Baseline

Compatibility: Desktop App Version 3.35 and above.

This command clears any previously set Baseline context.

\Clear_Baseline()

Example

\Clear_Baseline()\

Set_Requirements_Document

Compatibility: Desktop App Version 3.35 and above.

This command is used to set the requirements Document of a user's choice.

\Set_Requirements_Document('<<Requirements Document Name>>')

Parameter

<<Requirements	Optional	

Document Name>>		

Examples

\Set_Requirements_Document('$DEFAULT_REQUIREMENTS_DOCUMENT$')\
\Set_Requirements_Document('$\Business\Req Doc 1')\
\Set_Requirements_Document('Req Doc 1')\
\Set_Requirements_Document()\
\Set_Requirements_Document('$PROMPT_REQUIREMENTS_DOCUMENT$')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Set_Requirements_Document('$PROMPT_REQUIREMENTS_DOCUMENT$')\

\PROJECT_NAME\

\Fetch_Requirements_Tree_By_Condition( )\
\scan(a) \

\a: wbs \ \ a : Title \ [\ a : Id \]

\endscan\

Example

\Set_Requirements_Document('$DEFAULT_REQUIREMENTS_DOCUMENT$')\

When the above command is used with the default parameter, you need to first use the \Set_Project('<Project>')\ command in order to set the Project in the current context. Then, to fetch the Records of the Requirements Document use:

\Fetch_Requirements_Tree_By_Condition()\

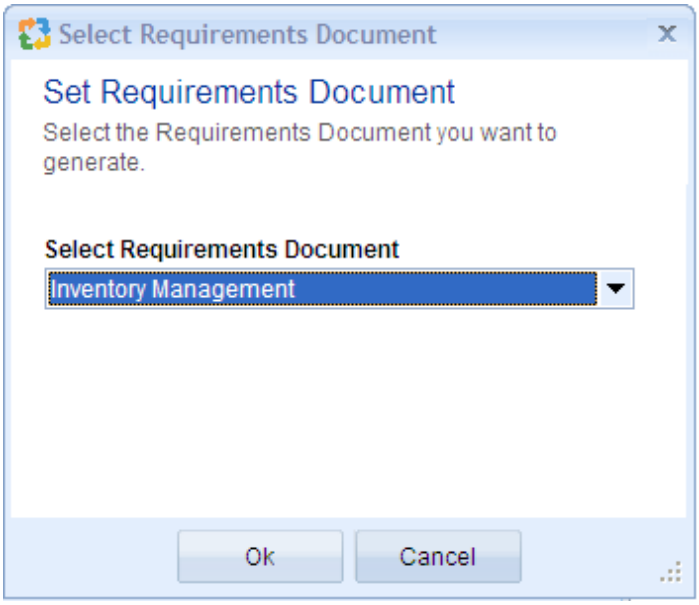
Examples

\Set_Requirements_Document()\

OR

\Set_Requirements_Document('$PROMPT_REQUIREMENTS_DOCUMENT$')\

This command, when used with "$PROMPT_REQUIREMENTS_DOCUMENT$" or no parameters, will display a pop-up window for Requirements Document selection. In this case, the user doesn't need to set the Project. If the user sets the Project, then the pop-up window will display only the Requirements Document selection option.



Example

\Set_Requirements_Document()\

Example

\Set_Requirements_Document('<<Requirement Document Path>>')\

This command will set the Requirements Document according to the Path set. The Path must contain the names of the Project and the Requirements Document.

\Set_Requirements_Document('$\Business\Req Doc 1')

In the above command "Business" is the Project name and "Req Doc 1" is the Requirements Document name.

Example

\Set_Requirements_Document('<<Requirement Document Name>>')\

NOTE: In the above command only the name of the Requirements Document and not its actual Path is specified.

This command will then search the Requirements Documents which are only in the current Project, specified by Set_Project. If the Requirements Document is found, then it will fetch the Record for this Requirements Document. If it is not found, then it will generate an error message.

\Set_Requirements_Document('Req Doc 1')\

In the above example, the "Req Doc 1" Requirements Document is set only when it is found in the current Project.

Set_Release

Compatibility: Desktop App Version 3.35 and above.

This command sets the context of the Document to a specified Release. Subsequent Fetch commands will fetch Records included in this Release only.

NOTE: The Release specified as the parameter for this command MUST exist in the Project that was set previously in the template. If the specified Release is not defined in that Project, the Release command will be ignored.

\Set_Release('<<Release Name>>')

Parameter

<<Release Name>>	Optional	Specify the name of the Release or specify a special parameter (explained below).

Example

\Set_Release('Alpha')\
\Set_Release('')\
\Set_Release('$PROMPT_RELEASE$')\

Example

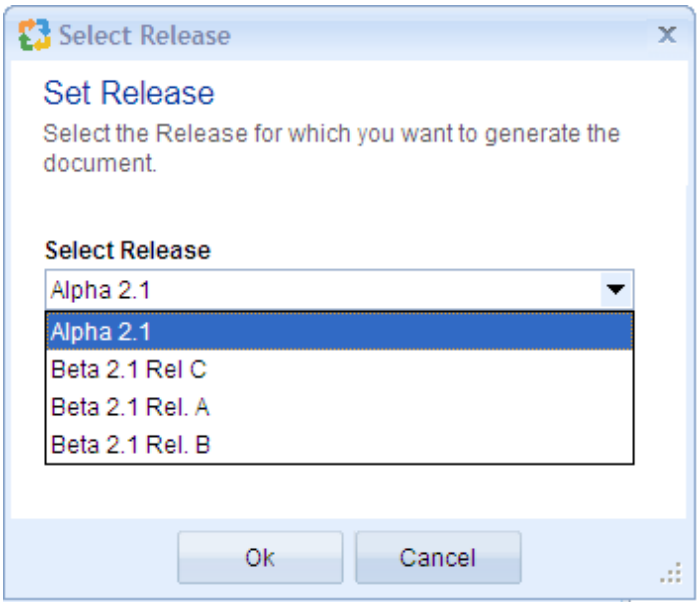
\Set_Release('')\
\Fetch_Use_Cases('' ,'Name')\
\scan(a) \
\a:Name\ \ a: Id\

\endscan\

Example

\Set_Release('$PROMPT_RELEASE$')\

If the argument passed is '$PROMPT_RELEASE$', then a pop-up window for the Release options displays. If the Project is not set, then the Project selection options are also available in that window. If the Project is already set before executing this command, then it will only display the Release selection options as seen below:



Examples

\ PROJECT_NAME \

\Set_Release('Alpha')\
\Fetch_Use_cases()\
\scan(a)\

\ a: name\ \a:id\ \a: priority\

\endscan\

Clear_Release

Compatibility: Desktop App Version 4.20 and above.

This is a miscellaneous command. It can be used in a template if you want to clear the Release which was set previously in the template. The command does not accept any parameters.

If you set the Release to fetch the records by using the Set_Release() command, then the Fetch commands will fetch only those Records that are included in the current set Release. If however, you want to fetch all the Records independent of the Release and you want to do it after setting the Release, then you can use the Clear_Release() command to fetch all the Records of the selected Project.

Clear_Release()

Example

\Clear_Release()\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME \

\Set_Release()\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

\a:Name\ [\ a: Id\]

\endscan\

\Clear_Release()\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

\a:Name\ [\ a: Id\]

\endscan\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

\a:Name\ [\ a: Id\]

\endscan\

Use Case Commands

Fetch_Use_Cases_By_Condition

NOTE: This command "Fetch_Use_Cases_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This is the primary command to fetch Use Cases which satisfy the specified filter conditions (optional).

\Fetch_Use_Cases_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Example

This command will fetch all Use Cases in the current Project.

\ Fetch_Use_Cases_By_Condition ()\

Examples

\Fetch_Use_Cases_By_Condition(' "Priority" = "Very High" ', 'Priority, Name') \
\ scan(a) \

\ a : Name\ [\ a : Id\]

\endscan\

Fetch_Use_Case_By_ID

NOTE: This command "Fetch_Use_Case_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.
This primary command fetches single Use Case for the specified Use Case ID. . If The Version Numberis also specified, and it fetches that specific version of the Use Case.

\Fetch_Use_Case_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory	
<< Version Num>>	Optional	

Examples

\ Fetch_Use_Case_By_ID('UC-1143')\
\ Fetch_Use_Case_By_ID ('1143')\
\ Fetch_Use_Case_By_ID ('UC-1143', '1.24')\

Fetch_Use_Cases

NOTE: This command "Fetch_Use_Cases" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This is the primary command to fetch Use Cases which satisfy the specified filters (optional).

\Fetch_Use_Cases('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the name of the filter that should be applied to the Fetch command. The filter you specify in this

		command should already be defined on the "Use Case Grid/List" interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Use_Cases()\
\Fetch_Use_Cases('Use Cases with status Approved', 'Priority, Name') \

This command will return all Use Cases that satisfy the filter Use Cases with status Approved sorted by Priority and then by Name in an ascending order.

Commands for Use Case Flow-of-Events

The following commands are used to insert Use Case Flow-of-Events into a document.

InsertFlows

Compatibility: Desktop App Version 3.35 and above.

This command inserts the entire Use Case Flow-of-Events section of the current Use Case into a document.

InsertFlows is a secondary command. It cannot be used independently. It must always be used inside the **\scan\-\endscan** of a Use Case Fetch command such as**: Fetch_Use_Cases_By_Condition, Fetch_Use_Case_by_ID, Fetch_Traced_Use_Cases** etc**.**

\ InsertFlows(<<Recordset Identifier>> : <<Caption for Use Case flow-of-events field>>)

Parameter

<< Caption for Use Case Flows Field>>	Mandatory	The default caption for the Use Case Flow-of-Events field is Flows.

Examples

\Fetch_Use_Cases()\

\ scan(a) \]\

\ a : Name\ [\ a : Id\]
\InsertFlows(a : Flows)

\endscan\

Fetch_Main_Success_Scenario

Compatibility: Desktop App Version 4.20 and above.

This command fetches the Header Record for the Main Flow, Main Path or Main Success Scenario of a Use Case Flow-of-Events.

\ Fetch_Main_Success_Scenario()

Fetch_Extensions

Compatibility: Desktop App Version 4.20 and above.

This command fetches the Header Records for the Alternate Flows, Alternate Paths or Extensions present in a Use Case Flow-of-Events..

\ Fetch_Extensions()

Fetch_Variations

Compatibility: Desktop App Version 4.20 and above.

This command fetches the Header Records for the Variations present in a Use Case Flow-of-Events.

\ Fetch_Variations()

Fields available

Field	Description
FLOW_ID	This is the unique identifier assigned to each Flow of a Use Case.
FLOW_NAME	This is the name of the Flow – e.g. 3a. Invalid Username or Password.
FLOW_NAME_RICH_TEXT	This is the name of the Flow in rich text format: e.g. **6a. Manually enter bar code** This needs to be outputted using the FRtf () command of DocProcessor. e.g. \ FRtf (b : FLOW_NAME_RICH_TEXT) \
FLOW_INDEX	This is the sequence number of this Flow in relation to other Flows of the Use Case i.e. the order in which they appear in the Flow-of-Events.
FLOW_TYPE	Flow Types: M – Main Flow-of-Events (or Main Path or Main Success Scenario) E – Extensions (or Alternate Flows or Alternate Paths) V – Variations
FLOW_STEPS	This field contains all the Steps of this Flow in a single block of rich text format. This needs to be outputted using the FRtf () command of DocProcessor. e.g. \ FRtf (b : FLOW_STEPS) \

Fetch_Main_Flow_of_Events

Compatibility: Desktop App Version 4.20 and above.

This command is the Use Cases Sub-report command. It is used to output the Main Flow in the document. This command will work the same as the **Fetch_Main_Success_Scenario()** command. **Fetch_Main_Flow_Of_Events** is the second name of the **Fetch_Main_Success_Scenario()**.

\Fetch_Main_Flow_of_Events(<<Flows Field>>)

Parameter

<<Flows Field>>	Mandatory	Specify the Flows field to fetch the Main Success Scenario or Main Path.

Example

\Fetch_Main_Flow_of_Events (a : Flows)\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Use_Cases_By_Condition(' "State" = "Casual" ')\
\scan(a) \
\a:id\ \a:state\

\Fetch_Main_Flow_of_Events (a : Flows)\

\scan(b)\\if (! Eof(b))\
\FRtf(b: FLOW_NAME_Rich_Text)\
\endif\
\ Fetch_Steps_For_Flow(a : Flows, b : FLOW_ID)\
\scan(c) \

\C: STEP_WBS_CODE\. \FRtf(c: Step_Data)\

\endscan\
\endscan \
\endscan \

Fetch_Alternate_Flow

Compatibility: Desktop App Version 4.20 and above.

This command is the Use Cases Sub-report command. It is used to output the Alternate Flows in the document. This command will work the same as the **Fetch_Extensions()** command.

The **Fetch_Alternate_Flow()** command is the second name of the **Fetch_Extensions()** command.

\Fetch_Alternate_Flow(<<Flows Field>>)\

Parameter

<<Flows Field>>	Mandatory	Specify the Flows field to fetch the Alternate Flows of the Main Flow.

Example

\Fetch_Alternate_Flow(a : Flows)\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

\Fetch_Use_Cases_By_Condition(' "State" = "Casual" ')\
\scan(a) \

\a: Name\ [\a: Id\]
State: \a:state\

\Fetch_Alternate_Flow(a:Flows)\
\scan(b)\\if (!Eof(b))\

\FRtf(b: Flow_Name_Rich_Text)\

\endif\
\Fetch_Steps_For_Flow(a : Flows, b : Flow_Id)\
\scan(c)\

\c: Step_Wbs_Code\\FRtf(c: Step_Data)\

\endscan\
\endscan\
\endscan\

Commands to Fetch Steps within each Flow

These commands fetch Steps within a Flow as individual records. You can either use the FLOW_STEPS field of the Header Record to insert all Steps of that flow using the InsertRTF command, or you can fetch each Step of the Flow individually using the commands below and then iterate through the Steps and insert the desired fields into the output document.

Fetch_Steps_For_Flow

Compatibility: Desktop App Version 4.20 and above.

This command fetches the Steps within a Flow as individual records. You MUST use this command after one of the Fetch Flow Header commands such as Fetch_Extensions, Fetch_Main_Success_Scenario, etc.

\ Fetch_Steps_For_Flow()

Fields available

Field	Description
STEP_ID	This is the unique identifier assigned to each Step.
STEP_PARENT_ID	This is the unique identifier of the Parent Step. If the Step is at the first level, this field contains the identifier of the Flow header. If the Step is a child Step i.e. indented to the 2nd or 3rd levels, this field will have the value of the immediate Parent Step's Step Parent ID.
STEP_FLOW_ID	This is the identifier of the Flow header record for this Step.
STEP_INDEX	This is the sequence number of the Step in this Flow. Within each Flow, the Step Index starts from 1.
STEP_WBS_CODE	This is the bullet number assigned to each Step in the Use Case Flows editor e.g. 3.2.
STEP_INDENTATION_LEVE	This is the indentation level of the Step. The indentation

L	level is 1 for the top-level Step.
STEP_ASCII_TEXT	The Text of the Step is in plain text without rich text formatting.
STEP_RICH_TEXT	The Text of the Step is in rich text format, exactly as it is displayed in the Flows editor. This needs to be outputted using the FRtf () command of DocProcessor. E.g. FRtf (c : STEP_RICH_TEXT)
STEP_SYSTEM_RICH_TEXT	This needs to be outputted using the FRtf () command of DocProcessor. E.g. FRtf (c : STEP_SYSTEM_RICH_TEXT)
STEP_ACTOR_RICH_TEXT	This needs to be outputted using the FRtf () command of DocProcessor. E.g. FRtf (c : STEP_ACTOR_RICH_TEXT)
SCREEN_ID	This is the ID of the first Screen associated with the Step.
SYSTEM_STEP_DATA	This is the Step executed by the System (Step where a systEm actor is used).
ACTOR_STEP_DATA	This is the Step executed by an Actor (Step where an Actor is used).
EXTENSIONS	Extensions for the Step.
VARIATIONS	Variations for the Step.

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ Fetch_Main_Success_Scenario(a : Flows)\

Flow

\scan(b)\\if (! Eof(b))\

\b: FLOW_NAME \

\endif\
\ Fetch_Steps_For_Flow(a : Flows, b : FLOW_ID)\
\scan(c) \

\C: STEP_WBS_CODE\ \ FRtf(c: Step_Data)\

\endscan\ \endscan\ \endscan\

Examples

\Fetch_Use_Cases ( )\
\scan(a) \

\a:Name\ [\ a: Id\]

\ Fetch_Extensions(a : Flows)\

Flow

\scan(b)\\if (! Eof(b))\

\b: FLOW_NAME \

\endif\
\ Fetch_Steps_For_Flow(a : Flows, b : FLOW_ID)\
\scan(c) \

\C: STEP_WBS_CODE\ \ FRtf(c: Step_Data)\

\endscan\ \endscan\ \endscan\

Examples

\Fetch_Use_Cases ( )\
\scan(a) \

\a:Name\ [\ a: Id\]

\ Fetch_Variations(a : Flows)\

Flow

\scan(b)\\if (! Eof(b))\

\b: FLOW_NAME \

\endif\
\ Fetch_Steps_For_Flow(a : Flows, b : FLOW_ID)\
\scan(c) \

\C: STEP_WBS_CODE\ \ FRtf(c: Step_Data)\

\endscan\ \endscan\ \endscan\

Insert_Flow_Diagram

Compatibility: Desktop App Version 3.35 and above.

Use this command to automatically insert a generated Flow Diagram of the current use case into the document. By default, the Flow Diagram is inserted in EMF image format. This is a secondary command; you must use it between the **\scan\...\endscan** sections of a Use Case Fetch command.
\Insert_Flow_Diagram()\

Examples

\Fetch_Use_Cases()\
\ scan(a) \

\ a : Name\ [\ a : ID\]

\InsertFlows(a : Flows)\
\Insert_Flow_Diagram()\
\endscan\

Examples

This command fetches the Flow Diagram at the first level.

\Fetch_Use_Cases_By_Condition('')\
\scan(a)\

Id: \ a: Id\

Traced records of type

\Fetch_Traced_Records_Of_Type_By_Condition('UC','Traces into')\
\scan(b)\

\ b : Name \ [\ b : id \]

\ Insert_Flow_Diagram(b:diagram)\
\endscan\
\endscan\

Insert_Flow_Diagram_As_Jpeg

Use this command to insert the automatically generated Flow Diagram of the current Use Case into the document in JPEG format.

\Insert_Flow_Diagram_As_Jpeg()

Insert_Flow_Diagram_As_EMF

This command inserts the Flow Diagram for the current Use Case in EMF format.

\Insert_Flow_Diagram_As_EMF()

Insert_Flow_Diagram_As_WMF

This command inserts the Flow Diagram for the current use case in WMF format.

\Insert_Flow_Diagram_As_WMF()

Insert_Flow_Diagram_Custom_In_CMs
Insert_Flow_Diagram_Custom_In_Inches
Insert_Flow_Diagram_Custom_In_Pixels

Compatibility: Desktop App Version 4.20 and above.

NOTE: The following commands are generated at any level from version 4.63 onwards.

These commands are used to output a Use Case Flow Diagram of the desired Width and Height and in the desired image format. These commands are the modified versions of the

\Insert_Flow_Diagram\, having the capability of re-sizing the Flow Diagram. These commands are used with Use Case Fetch commands only.

Insert_Flow_Diagram_Custom_In_CMs('<<Width>>', '<<Height>>', '<<Image Format>>', '<<Diagram field name>>')
Insert_Flow_Diagram_Custom_In_Inches('<<Width>>', '<<Height>>', '<<Image Format>>', '<<Diagram field name>>')

Parameters

<<Width>>	Mandatory	Specify the Width of the Diagram to be outputted into the document.
<<Height>>	Optional	Specify the Height of the Diagram to be outputted into the document.
<<Image Format>>	Optional	Image formats supported by this command are: • EMF • WMF • JPEG • JPG Specify one of the above values in this parameter. This command will then output the Diagram in the specified format. If the parameter is not specified, the Diagram will be outputted in EMF format.
<<Diagram field name>>	Optional	This is the Field Name where the Diagram is stored. If the field is empty, this command will output the Diagram in the first scan loop. If the specified field contains invalid data, this command will then return an error message.

Examples

\Insert_Flow_Diagram_Custom_In_Inches()\
\Insert_Flow_Diagram_Custom_In_CMs('', '20', 'WMF')\
\Insert_Flow_Diagram_Custom_In_Inches('12', '', 'JPEG', a : Diagram)\
\Insert_Flow_Diagram_Custom_In_Pixels('200', '', 'EMF')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Use_Cases()\
\ scan(a) \

\a:Name\ [\ a: Id\]

\Insert_flow_diagram_Custom_In_CMs('20','','EMF')\
\endscan\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Use_Cases()\
\scan(a) \

\a:Name\ [\ a: Id\]

\Insert_Flow_Diagram_Custom_In_Pixels('', '250', 'EMF')\
\endscan\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Use_Cases()\
\scan(a) \

\a:Name\ [\ a: Id\]

\Insert_Flow_Diagram_Custom_In_Inches('', '12', 'JPEG')\
\endscan\

Examples

\Fetch_Use_Cases_By_Condition()\
\scan(a) \

\a:Name\ [\ a : Id\]

\Fetch_Traced_Records_of_Type_by_condition('UC')\
\if (! eof(b))\

Traced Use Cases

\scan(b)\
\if (! Eof(b))\

\ b : Name \ [\ b : Id\]

\Insert_flow_diagram_Custom_In_CMs('20','','EMF' ,b : Diagram)\
\endif\
\endscan\
\endif\
\endscan\

Insert_Activity_Diagram_Custom_In_CMs
Insert_Activity_Diagram_Custom_In_Inches

Compatibility: Desktop App Version 4.50 and above.

This command is used along with Fetch Use Case commands. These commands are used to output the Use Case Flow's Activity Diagram in a vertical layout.

\Insert_Activity_Diagram_Custom_In_CMs(<<Flows field of Use Case>>, <<Image format>>, <<Width of the diagram>>, <<Height of the diagram>>)

\Insert_Activity_Diagram_Custom_In_Inches(<<Flows field of Use Case>>, <<Image format>>, <<Width of the diagram>>, <<Height of the diagram>>)

Parameters

<<Flows field of Use Case>>	Mandatory	The Activity Diagram is generated for Use Case Flows only. So the field passed to the command should be Use Case Flows. For fields other than Use Case Flows, this command will raise an error.
<<Image format>>	Optional	Image formats supported by this command are: GIF PNG BMP JPEG JPG EMF WMF If the parameter is not specified, the image will be displayed in EMF format.
<<Width of the diagram>>	Optional	This sets the Width of the Diagram to be displayed in the output.
<<Height of the diagram>>	Optional	This sets the Height of the Diagram to be displayed in the output.

Examples

\Insert_Activity_Diagram_Custom_In_CMs(a: flows, 'JPEG')\
\Insert_Activity_Diagram_Custom_In_Inches(a: flows, 'PNG', '7', '')\
\Insert_Activity_Diagram_Custom_In_CMs(a: flows, 'GIF', '', '20')\

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ InsertFlows(a : Flows)\

\Insert_Activity_Diagram_Custom_In_CMs(a: Flows, 'EMF', '', '20')\
\endscan\

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ InsertFlows(a : Flows)\

\Insert_Activity_Diagram_Custom_In_Inches(a: flows, 'PNG', '7', '12')\
\endscan\

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ InsertFlows(a : Flows)\

\Insert_Activity_Diagram_Custom_In_CMs(a: flows, 'GIF', '', '20')\
\endscan\

Insert_Swimlane_Diagram_Custom_In_CMs
Insert_Swimlane_Diagram_Custom_In_Inches

Compatibility: Desktop App Version 4.50 and above.

This command is used along with Fetch Use Case commands. These commands are used to output the Use Case Flow's Swimlane Diagram in a vertical or horizontal layout.

\Insert_Swimlane_Diagram_Custom_In_CMs(<<Flows field of Use Case>>, '<<Diagram Orientation>>', '<<Image format>>', '<<Width of the diagram>>', '<<Height of the diagram>>')

\Insert_Swimlane_Diagram_Custom_In_Inches(<<Flows field of Use Case>>, '<< Diagram Orientation >>', '<<Image format>>', '<<Width of the diagram>>', '<<Height of the diagram>>')

Parameters

<<Flows field of Use Case>>	Mandatory	The Activity Diagram is generated for Use Case Flows. So the field passed to the command should be Use Case Flows field. For fields other than Use Case Flows, this command will raise an error.
<<Diagram Orientation>>	Optional	This command displays the Diagram in a horizontal or vertical manner, depending upon the value of this parameter. If the value specified is "**Horizontal**", the Diagram will be displayed in a horizontal orientation. If the value specified is "**Vertical**", then the Diagram will be displayed vertically. If the parameter is not specified, the Diagram will be generated in a vertical orientation.
<<Image format>>	Optional	Image formats supported by this command are: • GIF • PNG • BMP • JPEG • JPG • EMF • WMF If the parameter is not specified, images will be outputted in EMF format.
<<Width of the diagram>>	Optional	This sets the Width of the Diagram to be displayed in the output.

<<Height of the diagram>>	Optional	This sets the Height of the Diagram to be displayed in the output.

Examples

\Insert_Swimlane_Diagram_Custom_In_CMs(a: Flows,'Horizontal','JPEG', '', '')\
\Insert_Swimlane_Diagram_Custom_In_Inches(a: Flows,'Horizontal','JPEG', '7', '')\

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ InsertFlows(a : Flows)\

\Insert_Swimlane_Diagram_Custom_In_CMs(a: Flows, 'Horizontal', 'EMF', '18', '')\
\endscan\

Examples

\scan(a) \

\a:Name\ [\ a: Id\]

\ InsertFlows(a : Flows)\
\Insert_Swimlane_Diagram_Custom_In_CMs(a: Flows, 'Vertical', 'EMF', '', '22')\
\endscan\

Use Case Sub-reports

Fetch_Traced_Requirements

NOTE: This command "Fetch_Traced_Requirements" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches the Trace Links of the Requirements Type records for the current primary record. It allows you to filter Trace Requirements Records by Link Types, specific

Requirements Types and Pseudo States. You can generate any Requirements field once you fetch Records using this command.

\Fetch_Traced_Requirements ('<<Link Types comma separated>>', '<< ID Prefix comma separated>>', '<<State filter>>','<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify the comma-separated Link Type names.
<< ID Prefix comma separated>>	Optional	Specify the Record Type Prefixes such as 'BRULE', 'FREQ' that appear in front of the identifier. For example, in the identifier BRULE-2348, BRULE is the ID Prefix and represents Business Rules.
<<State filter>>	Optional	This is the filter on Pseudo States such as, <<All Closed>>, <<All Open>>, <<Any>>. To get All Closed state records, you can supply the following combinations: '<<All Closed>>' '<All Closed>' 'All Closed' '<AllClosed' 'All Closed'
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas, along with ascending and descending indicators.

Fields Available

Field	Description
Trace Type	
Note	

Is Suspect	
Record Type specific fields	All fields of the Requirements Type Records including custom fields, may be inserted into the document.

Fetch_Traced_Requirements_By_Condition

NOTE: This command "Fetch_Traced_Requirements_By_Condition" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches the Trace Links of the Requirements Type Records for the current primary record. It allows you to filter the Trace Requirements Records by Link Types, specific Requirements Types and Pseudo States. You can generate any Requirements field once you fetch Records using this command.

\Fetch_Traced_Requirements_By_Condition('<<Link Types comma separated>>', '<<ID Prefix comma separated>>','<<Filter Condition>>','<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify the comma-separated Link Type names.
<< ID Prefix comma separated>>	Optional	Specify the Record Type Prefixes such as 'BRULE', 'FREQ' that appear in front of the identifier. For example, in the identifier BRULE-2348, BRULE is the ID Prefix and represents Business Rules.
<< Filter Condition>>	Optional	Specify the filter condition where you can use any Requirements field.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with ascending and descending indicators.

Examples

\Fetch_Use_Cases()\
\scan(a) \
\ a : Name \ [\ a : Id \ ]
\Fetch_Traced_Requirements_By_Condition('Traces Into', 'BREQ', ' "Priority" = "High" ')\
Traced Records
\scan(b)\
\b : trace type \ \ b : type \ \ b : Title \ \ b : Id \
\endscan\
\endscan\

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of the Requirements Type Records including custom fields, may be inserted into the document.

Fetch_Linked_Requirements

NOTE: This command "Fetch_Linked_Requirements" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches the Linked Requirements Records for primary use case records. You can generate any Requirements field once you fetch Records using this command.

\Fetch_Linked_Requirements('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	Specify one or more Requirements field names separated by commas along with ascending and descending indicators. If the sort order is not specified, the Linked Requirements are sorted by Requirements Type and Requirements ID.

Example

\Fetch_Linked_Requirements()\

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of Requirements Type Records including custom fields, may be inserted into the document.

Fetch_Use_Case_Associations

NOTE: This command "Fetch_Use_Case_Associations" has been. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Links for the current primary use case records. It fetches Links that are created for use cases such as <<include>> and <<extend>>. You can use any use case fields to generate the report once you've fetched the Records using this command.

\ Fetch_Use_Case_Associations ('<<Trace Direction>>', '<< Link Types comma separated>>', '<<Sort Order>>')

Parameters

<<Trace Direction>>	Optional	The values are '**Forward'** or '**Reverse'**. If you specify '**Forward'**, it will fetch only "<<include>>" links and *NOT* "included in" Links.
<< Link Types comma separated>>	Optional	Specify the comma-separated Link Type names.
<<Sort Order>>	Optional	Specify one or more Use Case Field Names separated by commas, along with ascending and descending indicators.

Examples

```
\Fetch_Use_Cases ()\
\ scan(a) \
\ a : name \ \ a: Id \
\ Fetch_Use_Case_Associations() \
\if (! eof(b))\
Use Case Links
\scan(b)\
\ b : trace type \ \ b : type \ \ b : name \ \ b : Id \
\InsertRtf(b : Description)\
\endscan\
\endif\
\endscan\
```

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of Use Case Type records including custom fields, may be inserted into the document.

Fetch_Use_Case_Actors

NOTE: This command "Fetch_Use_Case_Actors" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Actors added as Primary or Supporting Actors for the current primary use case record. You can generate any Actor field once you fetch Records with this command.

\Fetch_Use_Case_Actors('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	Specify one or more Actor field names separated by commas along with ascending and descending indicators. By default, the Actor records are first sorted by Actor Type, then by Actor Name.

Examples

\Fetch_Use_Cases('' ,'Name')\
\scan(a) \

\a:Name\ [\ a: Id\]

\ Fetch_Use_Case_Actors('Name')\

Primary Actors

\scan(b)\
\if (b: AssociationId = prmPrimaryActorId)\

\ b : Name\

\endif\
\endscan\

Supporting Actors

\scan(b)\

\if (b: AssociationId = prmSecondaryActorId)\

\ b : Name\

\endif\
\endscan\

\endscan\

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of Actor Type records including custom fields, may be inserted into the document.

Fetch_Use_Case_Scenarios

NOTE: This command "Fetch_Use_Case_Scenarios" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches a list of Use Case Scenarios for the current primary use case. It must be used inside primary commands which fetches the list of Use Cases. For e.g. **Fetch_Use_Cases, Fetch_Use_Case_By_ID, Fetch_Use_Cases_By_Condition**.

\Fetch_Use_Case_Scenarios()

Example

\Fetch_Use_Cases('' ,'Name')\
\scan(a) \

\a:Name\ [\ a: Id\]

\Fetch_Use_Case_Scenarios()\

Use Case Scenarios

\scan(b)\

\ b : Name \

\ InsertRtf(b : Description Goal in Context) \
\endscan\
\endscan\

Fields Available

All fields of the Use Case Scenarios Type records including custom fields, may be inserted into the document.

Actor Commands

Fetch_Actors_By_Condition

NOTE: This command "Fetch_Actors_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.
Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Actor records based on the specified filter conditions (optional).

\ Fetch_Actors_By_Condition ('<<Filter Condition >>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Examples

\ Fetch_Actors_By_Condition (' "State" = "Defined" ') \

\ Fetch_Actors_By_Condition ()\ - This command will fetch all Actor records in the current Project.

Fetch_Actor_By_ID

NOTE: This command "Fetch_Actor_By_ID" has been. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Actor record based on the Actor ID. If the Version Number is also specified, it fetches that specific version of the Actor.

\Fetch_Actor_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory	
<< Version Num>>	Optional	

Examples

\Fetch_Actor_By_ID('ACTR-1323')\
\Fetch_Actor_By_ID('1323')\
\Fetch_Actor_By_ID('ACTR-1323', '1.24')\

Fetch_Actors

NOTE: This command "Fetch_Actors" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Actor records based on the specified filters (optional).

\Fetch_Actors('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the names of the filters that should be applied to the Fetch command. The filters you specify in this command should already be defined in the Repository Objects Grid/List interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Actors()\ - This command will fetch all Actor records in the current Project.
\Fetch_Actors('Actors created in the last 3 days','Name') \

Use Case Diagram Commands

Fetch_Use_Case_Diagrams_By_Condition

NOTE: This command "Fetch_Use_Case_Diagrams_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Use Case Diagram records based on the specified filter conditions (optional).

\ Fetch_Use_Case_Diagrams_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Examples

\ Fetch_Use_Case_Diagrams_By_Condition (' "State" = "Approved" ','Name') \
\ Fetch_Use_Case_Diagrams_By_Condition ()\ - This command will fetch all Use Case Diagram records in the current Project.

Fetch_Use_Case_Diagram_By_ID

NOTE: This command "Fetch_Use_Case_Diagram_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Use Case Diagram record based on the specified ID. If the Version Number is also specified, it fetches that specific version of the Use Case Diagram.

\Fetch_Use_Case_Diagram_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory
<< Version Num>>	Optional

Examples

\Fetch_Use_Case_Diagram_By_ID('UCD-1234')\
\Fetch_Use_Case_Diagram_By_ID('1234')\
\Fetch_Use_Case_Diagram_By_ID('UCD-1234', '1.3')\

Fetch_Use_Case_Diagrams

NOTE: This command "Fetch_Use_Case_Diagrams" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Use Case Diagram records based on the specified filters (optional).

\Fetch_Use_Case_Diagrams('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the names of the filters that should be applied to the Fetch command. The filters you specify in this command should already be defined in the Repository Objects Grid/List interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Use_Case_Diagrams('Use Case diagrams created by me', 'Name') \
\Fetch_Use_Case_Diagrams()\ (Fetches all Use Case diagrams in the Project).

Special Fields Available

Refer to the InsertDiagram command described in this document.

Sub-reports

All common Repository Objects Sub-reports are available for this Record Type.

Diagram Commands

Fetch_Diagrams_By_Condition

NOTE: This command "Fetch_Diagrams_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Diagram records based on the specified filter conditions (optional). You can fetch records for different types of Diagrams, i.e., Screen Navigation Diagrams, Context Diagrams, Screen Prototypes, etc.

\Fetch_Diagrams_By_Condition('<<ID Prefix>>', '<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Mandatory	You must specify the Record Type Prefix for the type of Diagram Records that you want to fetch. For example, SCR, CTX, UCD, etc.
<<Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Examples

\Fetch_Diagrams_By_Condition('CTX', ' "Priority" = "High" ','Priority') \

This fetches "High" priority Context Diagram records in the Project in ascending order of Priority.

\Fetch_Diagrams_By_Condition('CTX', '', '') \

This fetches all Context Diagram records in the Project.

\ Fetch_Diagrams_By_Condition('CTX',' "Crt by" = "me" ', 'Crt dt')\

This fetches Context Diagram records for the specified filters, sorted by Create Date.

\ Fetch_Diagrams_By_Condition('DIA') \

This fetches all Diagrams (Free Format) records in the Project.

Fetch_Diagram_By_ID

NOTE: This command "Fetch_Diagram_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Diagram record based on the specified Diagram ID. If the Version Number is also specified, it fetches that specific version of the Diagram. You can fetch different types of Diagrams, i.e., Context Diagrams, Diagrams (Free Format), Screen Navigation Diagrams, Screen Prototypes, etc.

\Fetch_Diagram_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory
<< Version Num>>	Optional

Example

\ Fetch_Diagram_By_ID('DIA-1127')\

Fetch_Diagrams

NOTE: This command "Fetch_Diagrams" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Diagram records based on the specified filters (optional). You can fetch any type of Diagram records, i.e., Screen Navigation Diagrams, Context Diagrams, etc.

\Fetch_Diagrams('<<ID Prefix>>', '<<Filter Name>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Mandatory	You must specify the Record Type Prefix for the type of Diagram records that you want to fetch. For example, SCR, CTX, UCD, etc.

<< Filter Name>>	Optional	Specify the names of the filters that should be applied to the Fetch command. The filters you specify in this command should already be defined in the Repository Objects Grid/List interface.
<< Sort Order>>	Optional	

Example

\Fetch_Diagrams('SCR', 'Approved Screens', 'Priority') \

This command fetches Screen Prototype records (as specified by the ID Prefix 'SCR') sorted by Priority, for the specified filter Approved Screens.

Example

\Fetch_Diagrams('CTX','','') \

This command fetches all Context Diagram records in the current Project.

Fetch_Diagram_In_Multiple_Parts

Compatibility: Desktop App Version 8.30 and above.

Use this command to output a Diagram in multiple pages of desired Width and Height, as well as image type.
Fetch_Diagram_In_Multiple_Parts('<<Diagram field name>>', '<<Width>>', '<<Height>>','<<Image_Unit>>','<<Image Type>>','<<PageSortOrder>>')

Parameters

<<Diagramfieldname>>	Mandatory	
<< Width>>	Mandatory	
<< Height>>	Mandatory	

<<Image_Unit>>	Optional	Specify one of the following formats: a) CMs for Centimeters. b) PIXELS (default value) for Pixels. c) INCHES for Inches. If this value is set to blank, it displays the page with the default value PIXELS.
<<Image Type>>	Optional	Specify one of the following image types: a) EMF b) WMF c) JPEG (default value) If this value is set to blank, it displays the page with the default value JPEG.
<<PageSortOrder>>	Optional	Specify one of the following sort orders: a) Down_Then_Over (default value) b) Over_Then_DOWN If this value is set to blank, it displays the page with the default value Down_Then_Over.

Examples

Fetch_Diagram_In_Multiple_Parts(a:Diagram',865, 515,'','','')
Fetch_Diagram_In_Multiple_Parts(a:Diagram','5', '8','Inches','JPEG','OVER_THEN_DOWN')

Examples

\scan(a) \

\ a : Id \ \ a : Name \

\Fetch_Diagram_In_Multiple_Parts(a: Diagram,6,8,'Inches','JPEG')\

\scan(b) \
\ Insert_Diagram_Custom(b : Diagram)\
\endscan\

\endscan\

NOTE: Only the Fetch_Page_Size_Images_for_Diagram specified parameter is used to generate a Diagram output. When you Insert_Diagram_Custom specified Height, Width and other parameters are not considered.

Fields Available

Field	Description
Record Type Specific Field	All Record Type fields specified using the ID Prefix including custom fields, may be inserted into the document.

Fetch_Compare_Diagram_Records

Compatibility: Desktop App Version V8 and above.

This primary command is used to compare two records of Diagrams which support detail comparisons for Diagrams. This is currently implemented for BPD, Screen Mockups and ERD only.

Assumptions:
(a) Record Type should be the same.
(b) Comparison is based on UIDs of shapes. Hence, only records which are cloned, should be compared using this UDF.

\Fetch_Compare_Diagram_Records(<<ID>>,'<<Version1>>', '<<Version2>>')

Parameters

<<ID1>>	Mandatory	ID field/string for the first diagram record whose comparison will be fetched.
<<ID2>>	Mandatory	ID field/string for the second Diagram record whose comparison will be fetched.
<< Compare Type>>	Optional	'All'/'Difference' – Specify comparison fields to be fetched.

<<Property Type>>	Optional	Propery types to be fetched for comparison: a. "/'All' – All properties are shown in comparison b. 'Custom' – Only custom properties are shown in the comparison c. 'UI' – Only UI properties are shown in the comparison d. 'Significant' – Only significant properties are shown in the comparison e. 'Style' – Only style related properties are shown in the comparison
<<Show Difference on Diagram>>	Optional	Specify whether to show differences on the shapes graphically on the diagram – True/False – Default: False.
<<Shape List>>	Optional	Specify comma list of shape types to be fetched. E.g. 'Diagram': will fetch only the image of the diagrams. 'Task, Gateaway': will fetch all tasks and gateway type of shapes on a Business Process Diagram.

Fields Available

Field	Description
Version 1	First Version to compare
Version 2	Second Version to compare
Control Name	<<Display Text Identifier>> (<<Shape Type>>) E.g. for a Task shape, it returns **Process request (Task)**
Name	Property Name
Type	Property Type
Group	Group of the Property
Value 1	Property Value for version 1

Value 2	Property value for version 2
Merged	Field containing merged data for rich text fields
Is Rich Text	Specifies whether the property is rich text
Is Image	Specifies whether the property is image
Is Custom	Specifies whether this is a custom property
Is Color	Specifies whether this is a color property
Is Modified	Specifies whether the property has been modified
Control State	Specifies if the control is added, deleted or modified. Possible output: 'Added', 'Modified' and 'Deleted'
Control Type	Specifies type of the shape
Is Pinned	Specified if the property is pinned. E.g. properties such as name, text are the main properties to identify a shape and are termed as pinned properties.

Examples

\ Fetch_Compare_Diagram_Records (a: ID, b: ID, 'Difference', 'Significant', variable_Show_Diff_On_Image,'Diagram')\
\ Fetch_Compare_Diagram_Records (a: ID, b: ID, 'Difference', 'Significant', variable_Show_Diff_On_Image,'Task')\

Examples

Inserted

Modified

Deleted

Compare Diagram

\scan(a) [,page] \

\a: Name

\ Fetch_Compare_Diagram_Records (a: ID, b: ID, 'Difference', 'Significant', variable_Show_Diff_On_Image,'Task')\

Property	\b:Version1\	\b:Version2\

\endif\
\if(b : Is Rich Text = 'True')\

\b: Name\	\Insert_Rich_Text_Using_Word (b: Value1)\	\Insert_Rich_Text_Using_Word (b: Value2)\

\elsif(b : Is Image = 'True')\

\b: Name\	\Insert_Diagram_Custom (b:Value1)\	\Insert_Diagram_Custom (b:Value2)\

\else\

\b: Name\	\if (b : Is Modified = 'Y')\\b: Value1\\else\\b: Value1\\endif\	\if (b : Is Modified = 'Y')\\b: Value2\\else\\b: Value2\\endif\

\endif\\endscan\\endscan\

Fetch_Diagram_Compare

Compatibility: Desktop App Version 7.10 and above.

This primary command fetches the comparisons between two versions of Diagrams. Currently, this is implemented for BPD, Screen Mockups and ERD only.

\Fetch_Diagram_Compare(<<ID>>,'<<Version1>>', '<<Version2>>')

Parameters

<<ID>>	Mandatory	ID field/string for the first diagram record whose comparison will be fetched
<< Version1>>	Mandatory	First version field/string to compare

<< Version2>>	Mandatory	Second version field/string to compare
<< Compare Type>>	Optional	'All'/'Difference' – Specify comparison fields to be fetched.
<<Property Type>>	Optional	Propery types to be fetched for comparison: a. "/'All' – All properties are shown in comparison b. 'Custom' – Only custom properties are shown in the comparison c. 'UI' – Only UI properties are shown in the comparison d. 'Significant' – Only significant properties are shown in the comparison e. 'Style' – Only style related properties are shown in the comparison
<<Show Difference on Diagram>>	Optional	Specify whether to show differences on the shapes graphically on the diagram – True/False – Default: False.

Fields Available

Field	Description
Version 1	First Version to compare
Version 2	Second Version to compare
Control Name	<<Display Text Identifier>> (<<Shape Type>>) E.g. for a Task shape, it returns **Process request (Task)**
Name	Property Name
Type	Property Type
Group	Group of the Property
Value 1	Property Value for version 1

Value 2	Property value for version 2
Merged	Field containing merged data for rich text fields
Is Rich Text	Specifies whether the property is rich text
Is Image	Specifies whether the property is image
Is Custom	Specifies whether this is a custom property
Is Color	Specifies whether this is a color property
Is Modified	Specifies whether the property has been modified
Control State	Specifies if the control is added, deleted or modified. Possible output: 'Added', 'Modified' and 'Deleted'
Control Type	Specifies type of the shape
Is Pinned	Specified if the property is pinned. E.g. properties such as name, text are the main properties to identify a shape and are termed as pinned properties.

Examples

\Fetch_Diagram_Compare(a:ID,a: VersionId1, a:VersionId2,'Difference')\
\Fetch_Diagram_Compare(a:ID,a: VersionId1, a:VersionId2,'Difference','Custom')\
\Fetch_Diagram_Compare(a:ID,a: VersionId1, a:VersionId2,'Difference','All',True)\

Examples

Inserted

Modified

Deleted

Compare Diagram

\scan(a) [,page] \

\a: Name

\Fetch_Diagram_Compare(a: Id, a: VersionId1, a:VersionId2, 'Difference')\\scan(b)\\if(Bof(b))\

Property	\b:Version1\	\b:Version2\

\endif\
\if(b : Is Rich Text = 'True')\

\b: Name\	\Insert_Rich_Text_Using_Word (b: Value1)\	\Insert_Rich_Text_Using_Word (b: Valu

\elsif(b : Is Image = 'True')\

\b: Name\	\Insert_Diagram_Custom (b:Value1)\	\Insert_Diagram_Custom (b:Value2)\

\else\

\b: Name\	\if (b : Is Modified = 'Y')\\b: Value1\\else\\b: Value1\\endif\	\if (b : Is Modified = 'Y')\\b: Value2\

\endif\\endscan\\endscan\

Commands for Inserting Diagrams into a Document

Insert_Diagram_Custom

Compatibility: Desktop App Version 5.0 and above.

This command is used to output a Diagram in the desired Width and Height, as well as required image type format. You can put size constraints on the Diagram no matter how big the Diagram may be.

This command is used to output a Diagram and is used for diagramming Record Type commands only.

Insert_Diagram_Custom('<<Diagram field name>>, '<<Width>>', '<<Height>>', '<<Unit of Size>>', '<<Image Format>>', '<<Enforce Specified Size>>')

Parameters

<<Diagram field name>>	Optional	If the field passed is empty, then it will generate the default diagram field of the current record. If the passed field contains invalid data, then this command will return a proper error message.
<<Width>>	Optional	Set the Width of the Diagram to be displayed in the output.
<<Height>>	Optional	Set the Height of the Diagram to be displayed in the output.
<<Unit of Size>>	Optional	This parameter contains one of the values from the following: • CMs - If this is specified, the sizing dimensions i.e. the Height and the Width of the Diagram, will be measured in Centimeters. • INCHES - If this is specified, the sizing dimensions i.e. the Height and the Width of the Diagram, will be measured in Inches. • PIXELS - If this is specified, the sizing dimensions i.e. the Height and the Width of the Diagram, will be measured in Pixels. By default, the value for this parameter is PIXELS.
<<Image Format>>	Optional	The following image formats are available for this command: • EMF • WMF • JPG • PNG (this option is available in Desktop App version 7.1 or above)

		• GIF (this option is available in Desktop App version 7.1 or above) By default, the Diagram will be outputted in EMF format.
<<Enforce Specified Size>>	Optional	This parameter is available in Desktop App 7.1 and above. Indicates whether or not the image should be generated with the same size that is specified in the Width and Height parameters or it should maintain the current proportions of Width and Height. The default value of this parameter is **FALSE**. When the value of the parameter is **FALSE**, the image will be generated in the following manner: • If the size of image is smaller than the specified parameters, then it will not stretch the image to the specified size. • If one parameter (Width or Height) is specified, and the image is of a higher size, then the other parameter will be reduced to a proportionate scale. • If both the parameters are specified, the image will still be proportionate. In this case, whichever is smaller (Width or Height), will be taken as a reference to for scaling edown the image.

NOTE:

- If the Width and Height parameters are not specified, the image will be generated as it was originally defined.
- If any of the parameters for Width and Height is not specified, this command automatically calculates the other parameter in proportion to the supplied value.
- If Width and Height are set to blank, then the original size of the image will be generated in the document.

Examples

\Insert_Diagram_Custom(a: Screen Prototype) \

\Insert_Diagram_Custom(a: Screen Prototype,'', '8','Inches','JPG') \

\Insert_Diagram_Custom(a: Flow Diagram)\
\Insert_Diagram_Custom(a: Activity Diagram)\
\Insert_Diagram_Custom(a: Swimlane Diagram Horizontal)\
\Insert_Diagram_Custom(a: Swimlane Diagram Vertical)\
\Insert_Diagram_Custom(a: Diagram) \

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Diagrams ('SCR','' ,'Name')\
\scan(a)\

\ a : Type \ [\ a: Id\]
\ a : Name\

\Insert_Diagram_Custom(a: Screen Prototype,'6', '','Inches','JPG') \
\endscan\

InsertDiagram

Compatibility: Desktop App Version 3.35 and above.

This command inserts a Diagram image into the output document.. The default format of the Diagram image is EMF format.

\InsertDiagram(<<Recordset Identifier>> : <<Caption of Diagram Field>>)

Parameter

<<Caption of Diagram Field>>	Mandatory	The default caption of Diagram field is Diagram.

Example

\InsertDiagram(a : Diagram)\

Examples

\Fetch_Diagrams ('CTX','' ,'Name')\

Context Diagrams

\ scan(a) \

\ a : Name\ [\ a : Id\]

\ InsertDiagram (a : Diagram)\
\endscan\

Examples

\Fetch_Use_Cases()\

Use Cases

\ scan(a) \

\ a : Name\ [\ a : Id\]

\fetch_linked_screens()\
\if (! eof(b))\

Screens

\scan(b)\
\if (! Eof(b))\

\ b : Name \ [\ b : Id\]

\InsertDiagram(b: Diagram)\
\endif\
\endscan\
\endif\
\endscan\

Insert_Diagram_As_JPEG

Compatibility: Desktop App Version 4.20 and above.

This command inserts the diagram in JPEG format into the output document.

\Insert_Diagram_As_JPEG(<<Caption of Diagram Field>>)

Insert_Diagram_As_EMF

Compatibility: Desktop App Version 4.20 and above.

This command inserts the diagram in EMF format into the output document.

\Insert_Diagram_As_EMF(<<Caption of Diagram Field>>)

Insert_Diagram_As_WMF

Compatibility: Desktop App Version 4.20 and above.

This command inserts the diagram in WMF format into the output document.

\Insert_Diagram_As_WMF(<<Caption of Diagram Field>>)

Insert_diagram_Custom_In_CMs

Compatibility: Desktop App Version 4.20 and above.

NOTE: The fourth parameter **<<Diagram field name>>** is available from Desktop App version 4.63 and above.

This command inserts the Diagram in the specified Width and Height format in Centimeters.

\Insert_diagram_Custom_In_CMs(' <<Width >>, '<<Height>>', '<<Image format>>',' <<Diagram field name>>')

Parameters

<<Width>>	Optional	This is the Width of the image in Centimeters.
<<Height>>	Optional	This is the Height of the image in Centimeters.
<<Image format>>	Optional	This is the format of the image file. Allowed values are: • JPEG • JPG

		• EMF • WMF • BMP If image format is not supplied, the default image format is EMF.
<<Diagram field name>>	Optional	If the field passed is empty, then it will generate the default diagram field of the current record. If the passed field contains invalid data, then this command will return a proper error message.

Example

\ Insert_diagram_Custom_In_CMs('', '20', 'EMF')\

Examples

\Fetch_Use_Cases_By_Condition()\
\scan(a) \

\ a: Name\ [\ a: Id\]

\Insert_diagram_Custom_In_CMs('15', '','WMF')\
\endscan\

Examples

NOTE: This example is compatible with Desktop App Version 4.63 and above.

\Fetch_Use_Cases_By_Condition()\
Use Cases
\scan(a) \

\a:Name\ [\ a : Id\]

\fetch_linked_screens()\
\if (! eof(b))\

Screens

\scan(b)\
\if (! Eof(b))\

\ b : Name \ [\ b : Id\]

\Insert_diagram_Custom_In_CMs('15', '', 'WMF', b: Diagram)\
\endif\
\endscan\
\endif\
\endscan\

Insert_diagram_Custom_In_Inches

Compatibility: Desktop App Version 4.20 and above.

This command inserts a Diagram in the specified Width and Height in Inches.

\Insert_diagram_Custom_In_Inches('<<Width >>', '<<Height>>', '<<Image format>>', '<<Diagram field name>>')

NOTE: The fourth parameter **<<Diagram field name>>** is available from Desktop App version 4.63 and above.

Parameters

<<Width>>	Optional	This is the Width of the image in Inches.
<<Height>>	Optional	This is the Height of the image in Inches.
<<Image format>>	Optional	This is the format of the image file. Allowed values are: • JPEG • JPG • EMF • WMF • BMP If image format is not supplied, the default image format

		is EMF.
<<Diagram field name>>	Optional	If the field passed is empty, then it will generate the default diagram field of the current record. If the passed field contains invalid data, then this command will return a proper error message.

Examples

\ Insert_diagram_Custom_In_Inches('', '10', 'EMF')\

Insert_diagram_Custom_In_Pixels

Compatibility: Desktop App Version 4.20 and above.
This command inserts a Diagram in the specified Width and Height format in Pixels.

\ Insert_diagram_Custom_In_Pixels('<<Width >>', '<<Height>>', '<<Image format>>', '<<Diagram field name>>')

NOTE: The fourth parameter **<<Diagram field name>>** is available from Desktop App version 4.63 and above.

Parameters

<<Width>>	Optional	This is the Width of the image in Pixels.
<<Height>>	Optional	This is the Height of the image in Pixels.
<<Image format>>	Optional	This is the format of the image file. Allowed values are: • JPEG • JPG • EMF • WMF • BMP If image format is not supplied, the default image format

		is EMF.
<<Diagram field name>>	Optional	If the field passed is empty, then it will generate the default diagram field of the current record. If the passed field contains invalid data, then this command will return a proper error message.

Examples

\ Insert_diagram_Custom_In_Pixels('', '500', 'EMF')\

Requirements Commands

Fetch Requirements Commands

Fetch_Requirements_Branch_Starting_With_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Requirements records in a hierarchical format. It fetches the hierarchical list of Requirements records starting from the specified Requirements records. The Requirements records are also filtered according to the specified filter conditions (optional).

They are fetched in the same sequence and indentation levels as they are listed in the Requirements Tree or Requirements Document View interface.

\Fetch_Requirements_Branch_Starting_With_By_Condition('<<Starting Req ID>>', '<<Filter Condition>>', '<<WBS Code>>', '<<Show Parent>>')

Parameters

<<Starting Req ID>>	Mandatory	The identifier (ID) of the starting Requirements record from which the Requirements hierarchy has to be fetched from the Repository (i.e. DocProcessor will fetch this Record and all its child Records).

<< Filter Condition>>	Optional	
<<Starting WBS Code>>	Optional	Specify a starting WBS (Work Breakdown Structure) Code such as 2.2, 3.0, etc. This parameter specifies the starting bullet number for the hierarchical records. WBS is another name for the Outlined Numbering System**.** If the starting WBS Code is not specified, the system picks up the WBS Code specified for the corresponding Record Type of the current Project. The WBS Code for a Record Type can be specified using the Edit Project Wizard. Example: If the Starting WBS Code is specified as 5, then the generated records will be numbered from 5.1 and so on.
<<Show Parent>>	Optional	This is the Boolean parameter which takes the value as **TRUE** and **FALSE.** By Default the value of this parameter is **FALSE.** If the value is **FALSE**, it will ignore the header Requirements statements and therefore, it will not be inserted in the report. If the value is set to **TRUE**, this command also fetches the header Requirements and will not be inserted in the report. **NOTE**: If filter name is specified and if the Show Parent parameter is set to **TRUE** ,then this command will fetch the Parent Requirement of those child Records, which satisfy the filter condition, even if the Parent Requirement is not satisfying the filter condition.

		If the parameter is set to **FALSE**, it will ignore the Parent Requirement of those child Records that satisfy the filter condition.

Example

\Fetch_Requirements_Branch_Starting_With('REQ-102', ' "State" = "Approved" ','1.5')\

This fetches the Requirements branch starting from Requirements ID "REQ-102" for the specified filter conditions. These records will start from the WBS Code 1.5.

Examples

\Fetch_Requirements_Branch_Starting_With_By_Condition('BREQ-3142', ' "State" = "Approved" ')\
\scan(a)\
\a:wbs\ \a:title\ \a:id\
\endscan\

Fetch_Requirements_Branch_Starting_With

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Requirements records in a hierarchical format. It fetches the hierarchical list of Requirements records starting from the specified Requirements records. The Requirements records are also filtered according to the specified filters (optional).

The Requirements records are fetched in the same sequence and indentation levels as they are listed in the Requirements Tree or Requirements Document View interface.

\Fetch_Requirements_Branch_Starting_With('<<Starting Req ID>>', '<<Filter Name>>', '<<WBS Code>>', '<<Show Parent>>')

Parameters

<<Starting Req ID>>	Optional	The identifier (ID) of the starting Requirements record from which the Requirements hierarchy has to be fetched from the Repository (i.e. DocProcessor will fetch this Record and all its child Records).

<<Filter Name>>	Optional	The filters specified here should already be defined in the Requirements Tree or Requirements Document View interface. If Filter Name is not specified, all Records are fetched.
<<Starting WBS Code>>	Optional	Specify a starting WBS Code such as 2.2, 3.0, etc. This parameter specifies the starting bullet number for the hierarchical records. WBS is another name for the Outlined Numbering System. If the starting WBS Code is not specified, the system picks up the WBS Code specified for the corresponding Record Type of the current Project. The WBS Code for a Record Type can be specified using the Edit Project Wizard. Example: If the Starting WBS Code is specified as 5, then the generated records will be numbered from 5.1 and so on.
<<Show Parent>>	Optional	This is the Boolean parameter which takes the value as **TRUE** and **FALSE.** By Default the value of this parameter is **FALSE.** If the value is **FALSE**, it will ignore the header Requirements statements and therefore, it will not be inserted in the report. If the value is set to **TRUE**, this command also fetches the header Requirements and will not be inserted in the report. **NOTE**: If filter name is specified and if the Show Parent parameter is set to **TRUE** ,then this command will fetch the Parent Requirement of those child Records, which satisfy the filter condition, even if the Parent Requirement is not satisfying the filter condition.

		If the parameter is set to **FALSE**, it will ignore the Parent Requirement of those child Records that satisfy the filter condition.

Example

\Fetch_Requirements_Branch_Starting_With('REQ-102', 'Requirements with no outgoing Traces','1.5') \

This command fetches the hierarchy of requirements starting from Requirement ID REQ-102. The Requirements records are filtered using the specified filter Requirements with no outgoing Trace.

The Outlined Number will be generated starting from the starting WBS (bullet) Code 1.5.

Fetch_Requirements_Tree_By_Document_Id

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches the Requirements records in a hierarchical format for the specified Requirements document. If the Requirements document contains multiple Requirements hierarchies, all of them are fetched.

The Requirements records are returned in the same sequence and indentation levels as they are listed in the Requirements Tree or Requirements Document View interface. If a filter is specified, the Records will be filtered accordingly.

\Fetch_Requirements_Tree_By_Document_Id('<<Requirements Document Id>>', '<<Filter Name>>','<<Starting WBS Code>>', '<<Show Parent>>')

Parameters

<<Requirements Document ID>>	Mandatory	The identifier (ID) of the Requirements Document for which the Requirements hierarchy has to be fetched from the Repository.

		Instead of specifying a specific Requirements Document ID, you can also tell the system to pick up the Default Requirements Document for the current Project. Specify '$DEFAULT_REQUIREMENTS_DOCUMENT$' as the parameter that will pick up the Default Requirements Document for the current Project.
<<Filter Name>>	Optional	The filters specified here should already be defined in the Requirements Tree or Requirements Document View interface. If Filter Name is not specified, all Records are fetched.
<<Starting WBS Code>>	Optional	Specify a starting WBS Code such as 2.2, 3.0, etc. This parameter specifies the starting bullet number for the hierarchical records. WBS is another name for the Outlined Numbering System. If the starting WBS Code is not specified, the system picks up the WBS Code specified for the corresponding Record Type of the current Project. The WBS Code for a Record Type can be specified using the Edit Project Wizard. Example: If the Starting WBS Code is specified as 5, the generated records will be numbered from 5.1 and so on.
***<<Show Parent>>**	Optional	This is the Boolean parameter which takes values as **TRUE** or **FALSE**. By default, the value of this parameter is **FALSE**. If the value is **FALSE**, it will ignore the header Requirements statements which will not be outputted in the report.

		If the value is set to **TRUE**, this command also fetches the header Requirements which are outputted in the report. **NOTE**: If the filter name is specified and if the Show Parent parameter is set to **TRUE** ,then this command will fetch the Parent Requirement of those child Records which satisfy the filter condition, even if the Parent Requirement is not satisfying the filter condition. If the parameter is set to **FALSE**, it will ignore the parent Requirement of those child Records that satisfy the filter condition.

Example

\Fetch_Requirements_Tree_By_Document_Id('RDOC-1142', '','', '') \

This command fetches the entire Requirements hierarchy for a Requirements Document "RDOC-1142".

Examples

\Fetch_Requirements_Tree_By_Document_Id('121', ' Status Approved ', '1.1', 'True')\
\scan(a) \
\a: wbs \ \ a : Title \ \ a : Id \
\ InsertRtf(a : Description)\
\endscan\

Example

To use the Default Requirements Document of the Project as the first parameter, you can use '$DEFAULT_REQUIREMENTS_DOCUMENT$' in place of the Requirements Document ID.

\Fetch_Requirements_Tree_By_Document_Id('$DEFAULT_REQUIREMENTS_DOCUMENT$',
'Requirements with no outgoing Traces','1')\

This command fetches the hierarchy of Requirements records for the Default Requirements Document defined for the current Project. The Requirements records are filtered using the

specified filter Requirements with no outgoing Traces. The Outlined Number will be generated starting from Starting WBS Code 1.

NOTE: In the above command, the Record Type header Requirements records are not fetched in the output.

Fetch_Requirements_Tree_By_Document_ID_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Requirements records in a hierarchical format for the specified Requirements Document. If the Requirements Document contains multiple Requirements hierarchies, all of them are fetched.

The Requirements records are returned in the same sequence and indentation levels as they are listed in the Requirements Tree or Requirements Document View interface. If a filter condition is specified, the records are filtered according to that specification.

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('<<Requirements Document Id>>', '<<Filter Condition>>','<<Starting WBS Code>>', '<<Show Parent>>')

Parameters

<<Requirements Document ID>>	Mandatory	This is the identifier (ID) of the Requirements Document for which the Requirements is fetched from the Repository. Instead of specifying a specific Requirements Document ID, you can also tell the system to pick up the Default Requirements Document for the current Project. To do this, specify '$DEFAULT_REQUIREMENTS_DOCUMENT$' as the parameter.
<< Filter Condition>>	Optional	
<< Starting WBS	Optional	This is the starting WBS Code. This parameter specifies

Code>>		the starting indentation number for the hierarchical records. WBS is another name for a bulleted or Outlined Numbering System. If the Starting WBS Code is not specified, the system picks up the WBS Code specified for the corresponding Record Type of the current Project. The WBS Code for a Record Type can be specified using the Edit Project Wizard. For example, if the Starting WBS Code is specified as 5, then the generated records will be numbered from 5.1.
***<<Show Parent>>**	Optional	This is the Boolean parameter which takes the value as **TRUE** and **FALSE.** By Default the value of this parameter is **FALSE.** If the value is **FALSE**, it will ignore the header Requirements statements and therefore, it will not be inserted in the report. If the value is set to **TRUE**, this command also fetches the header Requirements and will not be inserted in the report. **NOTE**: If filter name is specified and if the Show Parent parameter is set to **TRUE** ,then this command will fetch the Parent Requirement of those child Records, which satisfy the filter condition, even if the Parent Requirement is not satisfying the filter condition. If the parameter is set to **FALSE**, it will ignore the Parent Requirement of those child Records that satisfy the filter condition.

Example

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('RDOC-1142', '','') \

This command fetches the entire Requirements hierarchy for a Requirements document "RDOC-1142".

Examples

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('RDOC-2435',' "State" = "Approved" ')\
\scan(a)\
\a:wbs\ \a:title\ \a:id\
\endscan\

Examples

To specify the Default Requirements Document of the Project, you can use '$DEFAULT_REQUIREMENTS_DOCUMENT$' in place of the Requirements Document ID.

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('$DEFAULT_REQUIREMENTS_DOCUMENT$',
'Crt by = Me','5') \
\scan(a) \
\a: wbs \ \ a : Title \ \ a : Id \
\endscan\

Fetch_Requirements_Tree_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches the Requirements records in a hierarchical format for the Default Requirements Document of the current Project. If the Requirements Document contains multiple Requirements hierarchies, all of them are accessed and returned.

The Requirements records are returned in the same sequence as they are listed in the Requirements Tree or Requirements Document View interface. You can generate the Requirements into a tabular format if desired.

**\Fetch_Requirements_Tree_By_Condition('<Filter condition>' , '<WBS Code>') **

Parameters

<< Filter Condition>>	Optional	
<<Starting WBS Code>>	Optional	If Starting WBS Code is not specified, the system picks up the WBS Code specified for the corresponding Record Type of the current Project. The WBS Code for a Record Type can be specified using the Edit Project Wizard. For example, if the Starting WBS Code is specified as 5, then the generated records will be numbered from 5.1 onwards.

Example

\Fetch_Requirements_Tree_By_Condition(' "Crt by" = "Me" ','5') \

This fetches Requirements records for the Default Requirements Document defined for the current Project based on the specified filter conditions. The records fetched will be shown starting from WBS Code 5

NOTE: The above command does not generate those Requirements which do not satisfy the filter conditions.

To insert Requirements fetched in an indented fashion in the report, you need to format the Requirements fields in the format as shown in the functions below:
Insert_Indented_Rtf(a: Description).

Examples

\Fetch_Requirements_Tree_By_Condition((' "Crt by" = "Me" ','5')\
\scan(a) \
\a: wbs \ \ a : Title \ \ a : Id \
\endscan\

Fetch_Requirement_By_ID

NOTE: This command "Fetch_Requirement_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Requirement record based on the specified Requirement ID. If you also specify the Version Number, it fetches that specific version of the Requirement.

\Fetch_Requirement_By_ID('<<ID>>', '<<Version Num>>')

Parameters

<< ID>>	Mandatory
<< Version Num>>	Optional

Examples

\Fetch_Requirement_By_ID('REQ-1123')\
\Fetch_Requirement_By_ID('1123')\
\Fetch_Requirement_By_ID('REQ-1123', '1.08')\

Fetch_Requirements_By_Condition

NOTE: This command "Fetch_Requirements_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above

This primary command fetches Requirements records in a Flat list format based on the specified filter conditions (optional). If you wish to fetch Requirements in a parent-child structure i.e. hierarchical, you need to use one of the below commands described in this document.

\ Fetch_Requirements_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Examples

\ Fetch_Requirements_By_Condition(‘ “Type” = “Business Requirement” ’) \
\ Fetch_Requirements_By_Condition(‘ “Type” <> “Functional Requirement” ’, ‘Priority, User Need’) \

Examples

\Fetch_Requirements_By_Condition(‘ “owner” = “John Doe” ’, ‘title’)\
\scan(a)\
\a:Title\ \a:id\ \a:owner\
\endscan\

Insert_Requirements_Hierarchy_Starting_with_ID

Compatibility: Desktop App Version 3.35 and above.

Use this command to directly insert the Requirements hierarchy into the generated document. This command directly inserts the hierarchical list of Requirements records, starting from the specified Starting Requirements ID into the output document in the same format as they appear in the Requirements Document View editor.

Data Field tags do not need to be placed in the output document.

The Requirements records are returned in the same sequence as they are listed in the Requirements Tree or Requirements Document View interface. You cannot access individual Requirements records using this command.

\Insert_Requirements_Hierarchy_Starting_with_ID(‘<<Starting Requirement ID>>’, ‘<<Filter Name>>’)

Parameters

<<Starting Requirement ID>>	Mandatory	This is the identifier (ID) of the Starting Requirements Record from which the Requirements hierarchy has to be fetched from the Repository (i.e. the system will fetch this Record and all its child Records).
<<Filter Name>>	Optional	The filters specified here should be defined in the Requirements Tree interface. If Filter Name is not specified, all Records are fetched.

Example

\Insert_Requirements_Hierarchy_Starting_with_ID('REQ-8392','Requirements with no incoming Traces')\

This command inserts the Requirements hierarchy starting from Requirements ID "REQ-8392" into the output document. The Requirements records use the specified filter Requirements with no Incoming Traces.

Insert_Requirements_Hierarchy_By_Document_ID

Compatibility: Desktop App Version 3.35 and above.

This command directly inserts the Requirements hierarchy into the generated document in the same format as they appear in the Requirements Document View editor. Data Field tags do not need to be placed in the document template.

The Requirements records are returned in the same sequence as they are listed in the Requirements Tree or Requirements Document View interface.

\Insert_Requirements_Hierarchy_By_Document_ID('<<Requirements Document Id>>', '<<Filter Name>>')

Parameters

<<Requirements Document ID>>	Mandatory	This is the identifier (ID) of the Requirements document for which the Requirements hierarchy has to be fetched from the Repository.

		Instead of specifying a specific Requirements Document ID, you can also tell the system to pick up the Default Requirements Document for the current Project.
<<Filter Name>>	Optional	The filters specified here should be defined in the Requirements Tree interface. If Filter_Name is not specified, all records are fetched.

Example

\Insert_Requirements_Hierarchy_By_Document_ID('RDOC-2333','') \

This command gets the complete hierarchy of Requirements for the specified Requirements Document. Requirements are automatically inserted into the output document in the same format as they appear in the Requirements Document View editor.

Fetch_Requirements_Tree

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches the hierarchical list of Requirements records for the Default Requirements Document of the current Project. If the Default Requirements Document contains multiple Requirements hierarchies, all of them are fetched.

The Requirements records are returned in the same sequence as they are listed in the Requirements Tree or Requirements Document View interface. The fetched records will be generated using the WBS Code specified for that type in the Project. You can generate the Requirements into a tabular format, if desired.

\Fetch_Requirements_Tree()

Examples

```
\Set_Project ($CURRENT_PROJECT$)\
\Fetch_Requirements_Tree() \
\scan(a)  \
\a: wbs \ \ a : Title \ \ a : Id \
\endscan\
```

Fetch_Requirements

NOTE: This command "Fetch_Requirements" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This is the primary command to fetch Requirements records based on the specified filters (optional), in a non-hierarchical manner.

\Fetch_Requirements('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the filters. The Filter Name specified here should be defined in the Requirements Tree interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Requirements('Requirements included in next Release') \
\Fetch_Requirements('Requirements included in next Release', 'User Need, Priority') \

Special Fields Available

The following special fields are available for Requirements Type records:

\ a : Indentation Level \
\ a : WBS \

These fields are not available in the editor and are available only in hierarchy Fetch commands.

Examples

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('$DEFAULT_REQUIREMENTS_DOCUMENT$', '
"Crt by" = "Me" ','5') \
\scan(a) \

\a: wbs \ \ a : Title \ \ a : Id \

\endscan\

Examples

\ Fetch_Repository_Objects_By_Hierarchy_By_Condition('','DE-346',' "Priority" = "High" ', '5', 'True')\
\scan(a) \
\if (! eof(a))\
\if (a : Indentation Level = '1')\

\ a : Title \ [\ a : Id \]

\elsif (a : Indentation Level = '2')\

\ a : Title \ [\ a : Id \]

\elsif (a : Indentation Level = '3')\

\ a : Title \ [\ a : Id \]

\else\

\ a : Title \ [\ a : Id \]

\endif\
\endif\
\endscan\

Fetch_Requirements_Document_Baselines

Compatibility: Desktop App Version V8 and above.

This command fetches Baselines created for the specified Requirements Document.

\Fetch_Requirements_Document_Baselines()

Parameters

There are no parameters for this command.

Fields Available

Field	Description
Baseline	Baseline name
Crt dt	Baseline Create date
Crt by	Baseline Created by
Upd dt	Baseline Update date
Upd by	Baseline Updated by
Baseline Type	Baseline Type – Complete or Selective
Baseline for	Baseline for – Project or Requirements Document
Baseline Date	Baseline Created for Date

Project Name	Project Name

Example

\Fetch_Requirements_Document_Baselines()\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Generates Requirements Document Baselines\

Requirements Document Baselines

\Set_Requirements_Document('$DEFAULT_REQUIREMENTS_DOCUMENT$')\
\Fetch_Requirements_Document_Baselines()\
\if (! eof(a))\

Baseline	Crt Dt	Crt by

\endif\
\scan(a)\

\a:Baseline\	\a:Crt dt\	\a:Crt by\

\endscan\

OneView Commands

Fetch_One_View_Section_By_Name

Compatibility: Desktop App Version 8 and above.

This primary command fetches a section from a specific OneView Document record. This command can be used within a Scan Loop.

\Fetch_One_View_Section_By_Name('<<ID>>', '<<SectionName>>')

Parameters

<<ID>>	Mandatory	Record identifier of the OneView Document or a Section Record. It can be ID_With or Without_Prefix.

		This parameter can be a Field or String. If the specified <<ID>> record does not present, this command will generate an error.
<<SectionName>>	Mandatory	Name of the Section Record. It can be Name of a Section Record in the OneView Document. This parameter must be a String. If the specified Section Name Record does not present in specified OneView Document, then this command will generate an error.

Fields Available

All fields for collection record type will be available.

Examples

\Fetch_One_View_Section_By_Name(a:ID, 'Preface')\
\Fetch_One_View_Section_By_Name(a:ID, 'Vision')\
\Fetch_One_View_Section_By_Name('OneV-321', 'Preface')\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\scan(a)\\if (! eof(a))\
\Fetch_One_View_Section_By_Name(a:ID, 'UC Folder Query ', 7)\\scan(b)\
\if (! eof(b))\
\Fetch_One_View_Pages(b:ID)\
\scan(c)\\if (! eof(c))\
\Fetch_Records_For_One_View_Page(c:ID)\
\scan(d)\\if (! eof(d))\
\if (d : Indentation Level = 1)\

\d: wbs \ [\Insert_Permalink(d: ID)\] \ d : Name \

\elsif (d : Indentation Level = 2)\

\d: wbs \ [\Insert_Permalink(d: ID)\] \ d : Name \

\elsif (d: Indentation Level = 3)\

\d: wbs \ [\Insert_Permalink(d: ID)\] \ d : Name \

\else\

\d: wbs \ [\Insert_Permalink(d: ID)\] \d : Name \

\endif\\endif\\endscan\
\endif\\endscan\
\endif\\endscan\
\endif\\endscan\

Fetch_One_View_Pages and Records

Compatibility: Desktop App Version 8 and above.

Primary command to fetch OneView Document record.

Fetch_One_View_Pages

This command fetches Pages (Record Type and ID List) for One View record.

Fetch_Records_For_One_View_Page

This command fetches Records for each One View record page. It must be used with command Fetch_Records_For_One_View_Page.

\Fetch_One_View_Pages('<<ID>>')\

Parameter

<<ID>>	Mandatory	Record IDENTIFIER of the record. It can be ID_With or Without_Prefix. This parameter can be a Field or String. If the specified <<ID>> record does not present, then this command will generate an error.

Fields Available

No fields are available for this command.

Examples

\Fetch_One_View_Pages(a : ID)\
\Fetch_One_View_Pages('OneV-621')\
\Fetch_One_View_Pages('621')\

Systemwide Template:

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Repository_Object_By_Id('OVACLL-298')\
\scan(a)\\if (! eof(a))\
\Fetch_One_View_Pages(a:ID)\
\scan(b)\\if (! eof(b))\
\Fetch_Records_For_One_View_Page(b:ID)\
\scan(c)\\if (! eof(c))\
\if (c : Indentation Level = 1)\

\c: wbs \ [\Insert_Permalink(c: ID)\] \ c : Name \

\elsif (c : Indentation Level = 2)\

\c: wbs \ [\Insert_Permalink(c: ID)\] \ c : Name \

\elsif (c: Indentation Level = 3)\

\c: wbs \ [\Insert_Permalink(c: ID)\] \ c : Name \

\else\

\c: wbs \ [\Insert_Permalink(c: ID)\] \c : Name \

\endif\\endif\\endscan\
\endif\\endscan\
\endif\\endscan\

Variants

Fetch_Variants_By_Condition

Compatibility: Desktop App Version 9.0 and above.
(Filter condition is not supported)

This secondary command fetches Variants that are created based on the conditions specified by you. It accesses and returns all Variant (branch) records for the primary record.

\Fetch_Variants_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	Specify the filter condition as per the required values of the Record. If not specified, all records will be fetched for the set context.
<< Sort Order>>	Optional	Specify the field names by which you want the results to be sorted. If the sort order is not specified, data will not be sorted.

Fields Available

Field	Description
Project	
ID	
Name	
Version	
Different	
State	
Priority	
Owner	

Create Date	
Modified By	
Modified Date	
Type	

Examples

\Fetch_Variants_by_condition (‘’,’Type, Id’)\

\Fetch_Variants_by_condition (‘’,’Project, ID’)\

\Fetch_Variants _By_Condition(‘’,'Project')\

Sample Template

\scan(a)\

\a:Name\

\Insert_Permalink(a: Id)\

\Fetch_Variants_by_condition (‘’,’Project, Id’)\\if (! eof(a))\

Variants

Project	Version	Different?	Name	Global ID	ID

\scan(b)\

\if (! eof(b))\

\ b : Project \	\ b : Version\	\ b : Different\	\ b : Name \	\ b : IBR Id\	\ b : Id \

\endif\

\endscan\

\endif\

\endscan\

Minutes of Meeting Commands

Fetch_Minutes_Of_Meeting_By_Condition

NOTE: This command "Fetch_Minutes_Of_Meeting_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Minutes of Meeting records based on the specified filter conditions (optional).

\Fetch_Minutes_Of_Meeting_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Examples

This command fetches all Minutes of Meeting records in a Project.

\ Fetch_Minutes_Of_Meeting_By_Condition()\

\ Fetch_Minutes_Of_Meeting_By_Condition(' "Meeting Date" = "12-17-2009" ') \

Fetch_Minutes_Of_Meeting_By_ID

NOTE: This command "Fetch_Minutes_Of_Meeting_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Minutes of Meeting record based on the specified Meeting ID. If the Version Number is also specified, it fetches that specific version of the record.

\Fetch_Minutes_Of_Meeting_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory
<< Version Num>>	Optional

Examples

\Fetch_Minutes_Of_Meeting_By_ID('MTG-5123')\
\Fetch_Minutes_Of_Meeting_By_ID('5123')\
\Fetch_Minutes_Of_Meeting_By_ID('MTG-5123', '1.05')\

Fetch_Minutes_Of_Meeting_By_Date

NOTE: This command "Fetch_Minutes_Of_Meeting_By_Date" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Minutes of Meeting records based on where the meeting date falls between From Date and To Date.

\Fetch_Minutes_Of_Meeting_By_Date('<<From Date>>', '<<To Date>>', '<<Sort Order>>')

Parameters

<<From Date>>	Optional	Date format should be "MM-DD-YYYY"
<<To Date>>	Optional	Date format should be should be "MM-DD-YYYY"

<< Sort Order>>	Optional	

Examples

\Fetch_Minutes_Of_Meeting_By_Date('01-26-05', '08-15-05', 'Meeting Date Desc') \

This command fetches all Minutes of Meeting records scheduled between the Jan 26 2005 to Aug 15 2005, sorted by the Meeting Date, in descending order.

\Fetch_Minutes_Of_Meeting_By_Date('', '11-14-05', 'Meeting Date Desc') \

This command fetches all Minutes of Meeting records scheduled before the Nov 14 2005, sorted by the Meeting Date in descending order.

\Fetch_Minutes_Of_Meeting_By_Date('01-26-05', '', 'Meeting Date Desc') \

This command fetches all Minutes of Meeting records scheduled after the Jan 26 2005, sorted by the Meeting Date in descending order.

Fetch_Minutes_Of_Meeting

NOTE: This command "Fetch_Minutes_Of_Meeting" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Minutes of Meeting records based on the specified filters (optional).

\Fetch_Minutes_Of_Meeting('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	The Filter Name specified here should be defined in the Minutes of Meeting List interface.
<< Sort Order>>	Optional	

Example

\Fetch_Minutes_Of_Meeting ()\ - This command will fetch all Minutes of Meeting records in a Project.

\Fetch_Minutes_Of_Meeting ('Meetings scheduled for next week') \

Online Document Commands

Fetch_Online_Documents_By_Condition

NOTE: This command "Fetch_Online_Documents_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Online Document records based on the specified filter conditions (optional).

\Fetch_Online_Documents_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Example

\ Fetch_Online_Documents_By_Condition() \ - This command will fetch all Online Document records in the current Project.

\ Fetch_Online_Documents_By_Condition(' "Name" = "Scope and Vision" ', 'Name') \

Fetch_Online_Document_By_ID

NOTE: This command "Fetch_Online_Document_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Online Document record based on the specified Online Document ID. If the Version Number is specified, it fetches that specific version of the Online Document.

\Fetch_Online_Document_By_ID('<<ID>>', '<<Version Num >>')

Parameters

<< ID>>	Mandatory
<< Version Num>>	Optional

Examples

\Fetch_Online_Document_By_ID('ODOC-1128')\
\Fetch_Online_Document_By_ID('1128')\
\Fetch_Online_Document_By_ID('ODOC-1128', '2.02')\

Fetch_Online_Documents

NOTE: This command "Fetch_Online_Documents" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Online Documents records based on the specified filters (optional).

\Fetch_Online_Documents('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the names of the filters that should be applied to the Fetch command. The filters you specify in this command should already be defined in the Repository Objects List/Grid interface.
<< Sort Order>>	Optional	

Example

\Fetch_Online_Documents() \ - This command will fetch all Online Documents records.

\Fetch_Online_Documents('Documents created by me', 'Name') \

Release Commands

Fetch Release Commands

Fetch_Release_By_ID

NOTE: This command "Fetch_Release_By_ID" has been deprecated. As an alternative, use command Fetch_Repository_Object_By_Id.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Release record based on the specified Release ID. If the Version Number is also specified, it fetches that specific version of the Release.

\Fetch_Release_By_ID('<<Release ID>>')

Parameters

<< ID>>	Mandatory	Specify the Release identifier.

Example

\Fetch_Release_By_ID('RLS-5011')\

Fetch_Releases

NOTE: This command "Fetch_Releases" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

This primary command fetches Release records based on the specified filters (optional).

\Fetch_Releases('<<Filter Name>>', '<<Sort Order>>')

Parameters

<< Filter Name>>	Optional	Specify the filters. The Filter Names specified here should be defined in the Repository Objects List/Grid interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Releases('','')\

This command fetches the list of all Releases defined for the Project set in the Project Context environment variable.

\Fetch_Releases('','End dt Desc, State') \

This command fetches the list of all Releases defined for the value of the Project set in the Project Context environment variable. They are sorted in a descending order by the field End Date and ascending order by the field State.

Fetch_Release_Tracking_Items

This command fetches all Tracking Items included in the specified Release ID and is sorted by Tracking Items.

\Fetch_Release_Tracking_Items('<<Release ID>>', '<<Sort Order>>')

NOTE: This command will be available in a future release of DocProcessor.

Example

\Fetch_Release_Tracking_Items('RLS-1244','Title Name, Priority, State') \

Fetch_Release_Repository_Objects

This command fetches all Repository Objects included in the specified Release ID and is sorted by Repository Objects.

\Fetch_Release_Repository_Objects('<<Release ID>>', '<<Sort Order>>')

NOTE: This command will be available in a future release of DocProcessor.

Example

\Fetch_Release_Repository_Objects('RLS-1244','Title Name, Priority, State') \

Fields Available

Field	Description
Record Type Specific Field	All fields of Release Record Type including custom fields, may be inserted into the document.

Release Sub-reports

Fetch_Items_In_Release

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Tracking Items records that have been included in the current primary Release. You have the ability to use all fields defined for Tracking Items for formatting as well as sorting.

\Fetch_Items_In_Release('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	

Example

\ Fetch_Items_In_Release('State') \

Fetch_Items_In_Release_By_Condition

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Tracking Items records that have been included in the current primary Release. All fields defined for Tracking Items may be used for formatting, creating conditions, and sorting.

\Fetch_Items_In_Release_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	Specify the Tracking Items fields to create the filter conditions.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Example

\Fetch_Items_In_Release_By_Condition (' "State" = "Open" ' ,'State') \

Fetch_Objects_In_Release

NOTE: This command "Fetch_Objects_In_Release" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Repository Objects records that have been included in the current primary Release. You can use all fields defined for Repository Objects for formatting, as well as sorting.

\Fetch_Objects_In_Release('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator. The default sort order for the fetched Records is by Record Type then by the Record Title/Name.

Example

\Fetch_Objects_In_Release('State') \

Fetch_Objects_In_Release_By_Condition

NOTE: This command "Fetch_Objects_In_Release_By_Condition" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Repository Objects records that have been included in the current primary Release. You have the ability to use all fields defined for Repository Objects for formatting, creating conditions, and sorting.

\Fetch_Objects_In_Release_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	Specify the Tracking Items fields to create the filter conditions.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator. The default sort order for the fetched Records is by Record

		Type and then by the Record Title/Name.

Example

\Fetch_Objects_In_Release_By_Condition (' "State" = "Open" ' ,'State') \

You can use all fields defined for Repository Objects for formatting as well as sorting.

Repository Objects Commands

Fetch_Repository_Objects_By_Conditions

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Repository Objects records of any type based on the specified filter conditions (optional). If ID Prefix (optional) is specified, only those types of records will be fetched.

This is a generic function and can be used to fetch all types of Repository Objects. It is useful in fetching Custom Record Types created as Descendant Record Types in the Repository.

\Fetch_Repository_Objects_By_Condition('<<ID Prefix>>', '<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Optional	Specify the Record Type Prefixes such as 'MOD', 'DOC' that appear in front of the identifiers.
<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Examples

\ Fetch_Repository_Objects_By_Condition('MOD', ' "Priority" = "High" and "Target Release" = "Beta" ', 'Functional Area') \

This command fetches all Module Types of records that have the Target Release set as Beta, the Priority set to High, and sorted by Functional Area.

\ Fetch_Repository_Objects_By_Condition('DOC', ' "Crt by" = "Me" ') \

This command fetches all Documents for the specified filter conditions.

Fetch_Repository_Object_By_ID

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Repository Object record of any type based on the specified ID. If the Version Number is also specified, it fetches that specific version of the Repository Object.

\Fetch_Repository_Object_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Optional	Specify the Repository Object identifier.
<< Version Num>>	Optional	Specify the Version Number you wish to fetch.

Example

\Fetch_Repository_Object_By_ID('MOD-1023')\

Fetch_Repository_Objects

NOTE: This command "Fetch_Repository_Objects" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Repository Objects records of any type based on the specified filters (optional) and Record Type (if ID Prefix is passed).

This is a generic function and can be used to fetch different types of Repository Objects. (E.g. Modules, Database Tables, Use Cases).

\Fetch_Repository_Objects('<<ID Prefix>>', '<<Filter Name>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Optional	Specify the Record Type Prefix such as 'MOD', 'DOC' that appears in front of the identifier. For example, in the identifier MOD-2348, MOD is the ID Prefix and represents Modules.
<< Filter Name>>	Optional	Specify the filters. The Filter Name specified here should be defined in the Repository Objects List interface.
<< Sort Order>>	Optional	

Examples

\Fetch_Repository_Objects('MOD', 'Modules in Beta Release', 'Functional Area') \

This command fetches all Module records for the specified filter Modules in Beta Release, sorted by Functional Area in an ascending order.

\Fetch_Repository_Objects('DOC', 'My Documents') \

This command fetches all Documents records for the specified filter My Documents.

Fetch_Records_By_Folder

Compatibility: Desktop App Version 4.20 and above.

This primary command fetches Records by their folder hierarchy. It fetches Repository Objects of the specified ID Prefix type with its folder hierarchy.

\Fetch_Records_By_Folder('<<ID Prefix>>', '<< Folder Path>>')

Parameters

<<ID Prefix>>	Mandatory	Specify the ID Prefix of the Record Type you want to fetch. E.g. 'UC', 'DOC', 'CTX', etc.
<<Folder Path>>	Optional	Specify the Path of the Folder for which you want to generate the Records. The Folder Path must contain its Project path.

Example

This template generates all use cases under the folder Video Rentals of the Project, Video Rentals System.

```
\Fetch_Records_By_Folder('UC' ,'Video Rentals System\Video Rentals')\
\scan(a)  \
\if(a : Type= 'Folder')\
    \a:Folder path \

\else\
    \ a : Name \  [\a: Id\]

\endif\
\endscan\
```

Fetch_Records_By_Folder_By_Condition

Compatibility: Desktop App Version 4.20 and above.

This primary command fetches Records that are contained in Folders based upon the filter conditions specified in the parameters. If the filter condition is not specified, then it fetches all Records of the specified Folder.

\Fetch_Records_By_Folder_By_Condition('<<ID Prefix>>', '<< Folder or Project Path>>', '<<Filter Condition>>')

Parameters

<<ID Prefix>>	Mandatory	This is the Prefix of the record identifier of the Record Type. e.g.: 'UC', 'TC' etc.
<< Folder or Project Path>>	Optional	Specify Folder Name / Complete Folder Path / Project Path / Project Name, to fetch the Records for that Project or Folder. Once specified, the Records are fetched based on the filter conditions. If the Project Path is specified, all Records are fetched for that Project as per the filter conditions. Examples: 1) Video Rentals System\Test Case Folder 2) Video Rentals System\Database It will work if there are spaces before or after '\', but Project and Folder Names must be correct. Project and Folder Names are case insensitive. If only the Folder Name is specified, then the DocProcessor will try to search for that Folder in the current Project.
<<Filter Condition>>	Optional	This filter condition fetches only those Records in the Folder which satisfy the condition. If the filter condition is not specified, then it fetches all Records from the specified path. Examples: Priority = High State = Approved

Example

\Fetch_Records_By_Folder_By_Condition('TC' ,'Video Rental System\ Test Cases Folder', ' "Priority" = "High" ')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \
\Fetch_Records_By_Folder_By_Condition('UC' ,'Video Rental System\Rental Management', ' "Priority" = "High" and "State" = "Brief" ')\

\scan(a) \
\if (! eof(a))\
\if(a : Type= 'Folder')\

\a:Folder Path\

\else\

\ a: Id\	\a:Name\	\a:Priority\	\a: State\

\endif\
\endif\
\endscan\

Fields Available

Field	Description
Type	
Folder Path	
Indentation Level	
All Record Type fields	All the fields defined for the specified Record Type using the ID Prefix.

Fetch_Repository_Objects_By_Hierarchy_By_Condition

Compatibility: Desktop App Version 5.0 and above.

This command fetches the Records hierarchy for the specified Record Type, Prefix or ID. It is used to fetch hierarchical and/or ordered Repository Objects.

\Fetch_Repository_Objects_By_Hierarchy_By_Condition('<<ID Prefix>>', '<<Starting Record ID >>','<<Filter Name/Condition>>','<<WBS Code>>', ' <<Show Parent>>')

Parameters

<<ID Prefix>>	Mandatory	Specify the ID Prefix of Record Type you want to fetch. E.g. 'UC', 'TC', etc. This is mandatory if second parameter Starting Record ID is not provided. This is optional if second parameter Starting Record ID is specified.
<<Starting Record ID >>	Optional	This fetches the Records hierarchy for the Starting Record ID specified. This is mandatory if first parameter ID Prefix is not provided. This is optional if first parameter ID Prefix is provided.
<<Filter Name/Condition>>	Optional	The filter condition can be used to fetch desired Records such as Priority = High. **From *TopTeam* version 13.41 or higher**: You can also add predefined filter conditions that are available in *Repository Object list/tree* editor of a record type.
<<WBS Code>>	Optional	The WBS Code for the generated Records will start with the specified value.

		For example, if WBS Code is specified as 5, then the generated Records will start from 5.1. If this parameter is not specified, the WBS Code for each Record Type will be picked up from the Project settings.
<<Show Parent>>	Optional	This is the Boolean parameter which takes the values of **TRUE** and **FALSE.** By default, the value of this parameter is **FALSE**. - If the filter condition is not specified: o And the value is **FALSE,** then it will ignore the Header Records and they will not be generated in the Report. o If the value is set to **TRUE**, then it will fetch the Header Records in the Report. - If the filter condition is specified: o And the value is **FALSE,** then it will ignore the Parent Records that do not satisfy the filter condition of the Child Records which do satisfy the filter condition. o If the value is set to **TRUE**, then this command will fetch the Parent Records of those Child Records, which satisfy the filter condition, even if the Parent Records are not satisfying the filter condition.

NOTE: This command fetches the Records hierarchy for the specified Record Type, Prefix or ID from the current Project set in the template.

Fields Available

Field	Description
ID Prefix	ID, Prefix, Record Type specific columns, including custom fields, can be generated i.e. Fields displayed in the editor for that Record Type.

Special Fields Available

The following Requirements Type records special fields are available:

\ a : Indentation Level \

\ a : WBS \

These fields are not available in the editor. They are available only in the hierarchy Fetch commands.

Examples

\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ', '', ' "Crt by" = "Me" ') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ', '', ' "Crt by" = "Me" ','3') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ', '', ' "Crt by" = "Me" ','5', 'True') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ', '', '','', 'True') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ', '', '','', 'False') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '343', '','') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '343', ' "Priority" = "High" ') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '343', ' "Priority" = "High" ','3') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '343', ' "Priority" = "High" ','5', 'True') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '346', '','', 'True') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('', '346', '','', 'False') \
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOB','', 'High Priority records')\ **[Applicable from v13.41 and above Desktop App]**

Example

\Set_Project('$CURRENT_PROJECT$')\

```
\ Comments("Fetch record hierarchy for the specified record type ")\
\Fetch_Repository_Objects_By_Hierarchy_By_Condition('GOBJ','', ' "Crt by" = "Me" ','2') \
\scan(a)\

\a:Wbs\. \a: Name\ [\a:Id\]

\endscan\


\ Comments("Fetch record hierarchy for the specified record type ")\
\ Fetch_Repository_Objects_By_Hierarchy_By_Condition('','DE-346',' "Priority" = "High" ', '5', 'True')\
\scan(a)  \
\if (! eof(a))\
\if (a : Indentation Level = '1')\

\a: wbs \. \ a : Name \ [\ a : Id \]

\elsif (a : Indentation Level = '2')\

        \a: wbs \. \ a : Name \ [\ a : Id \]

\elsif (a : Indentation Level = '3')\

                \a: wbs \. \ a : Name \ [\ a : Id \]

\else\

                        \a: wbs \. \ a : Name \ [\ a : Id \]

\endif\
\endif\
\endscan\
```

Fetch Sub-report Commands for Repository Objects and Tracking Items

Comments Sub-report Commands

Fetch_Comments

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Comments records for the current primary record.

\Fetch_Comments('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional

Example

\Fetch_Comments('Date desc') \

Fetch_Comments_By_Condition

Compatibility: Desktop App Version 5.0 and above.
This secondary command fetches Comments records for the current primary record based on the specified filter conditions (optional).

\Fetch_Comments_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Compatibility: Desktop App Version 8.60 and above.

\Fetch_Comments_By_Condition('<<Filter Condition>>' , '<<Sort Order>>',<<ShowContextInfo>>)

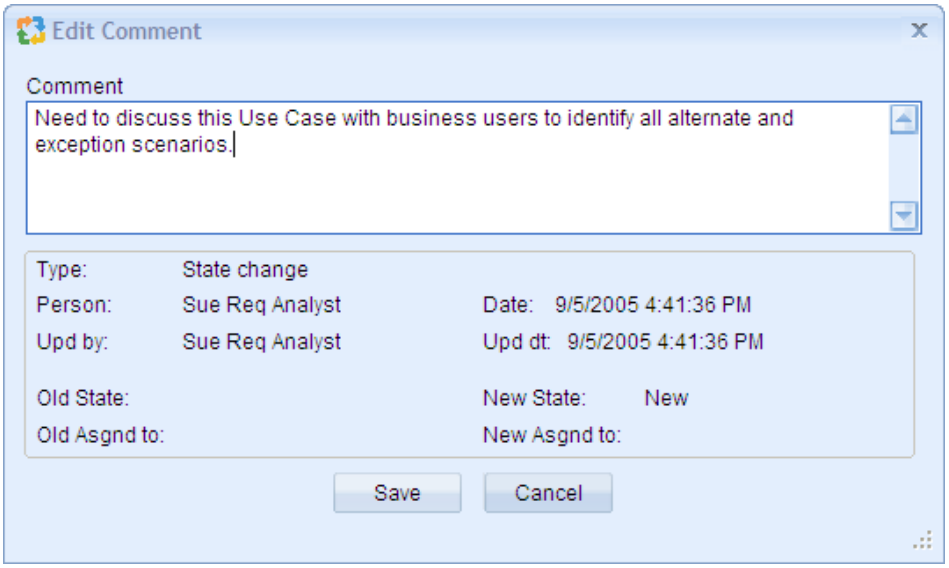
Parameters

<< Filter Condition>>	Optional	Specify the filter condition as per the required values of the Record. If not

		specified, all records will be fetched for the set context.
<< Sort Order>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is not sorted.
<<ShowContextInfo>>	Optional	This flag is by default set to **FALSE**. If user wants to load context of comment then set this Flag to **TRUE.**

Fields Available

All Comments records fields may be inserted into the document.
Double-click a Comment to open the Comments Detail editor. This editor lists all the fields that are available.



Field	Description
Comment	This is text entered as a Comment.
Type	This is a simple Comment, a state change event with optional Comments.
Person	This is the user who added the Comment.
Date	This is the Date (and Time) when the Comment was added.

Upd by	This is the user who last updated the Comment.
Upd dt	This is the last updated Date and Time.
Old State	In case of a state change Comment, this field displays the State (Status) of the Record before the Comment was entered. Otherwise, this field is empty.
New State	In case of a state change Comment, this field displays the new State (Status) of the Record when the Comment was entered.
Old Asgnd To	In case of a State or an Owner change Comment, this field displays the Owner (or Assigned To Person) of the Record before the Comment was entered.
New Asgnd To	In case of a State or Owner an change Comment, this field displays the New Owner (or Assigned To Person) of the Record when the Comment was entered.
Context	This field displays Contextual comment info. (This field is available for Desktop App 8.60 and above.)

Example

\Fetch_Comments_By_Condition(‘“Type” = “State Change” ’, ‘Date desc’) \

Examples

\Fetch_Use_Cases ()\
\scan(a) \

\ a : Name \ [\ a : Id \ ]
Comments (Sub-report)

\Fetch_comments_by_condition(‘ “Person” = “User1” ’)\
\scan(b)\

\ b : date \ \ b : person \
\ b : comment \

\endscan\

\endscan\

Examples to Show Context Info of Comment

\Fetch_Use_Cases ()\
\scan(a) \

\ a : Name \ [\ a : Id \ ]
Comments (Sub-report)

\Fetch_comments_by_condition(' "Person" = "User1" ' ,'Date',True)\
\scan(b)\

\ b : date \ \ b : person \
\ b : comment \
\ Insert_Diagram_Custom(b: Context)\

\endscan\
\endscan\

Attachments Sub-report Commands

Fetch_Attachments

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Attachments records for the current primary record.

\Fetch_Attachments('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional

Example

\Fetch_Attachments('Date desc')\

Fetch_Attachments_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Attachments based on specified filter conditions (optional).

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

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Example

\Fetch_Attachments_By_Condition(‘ “added on” = “1-13-2008” ’,’Size kb desc’)\

Fields Available

Field	Description
File Name	This is the name of the attached file.
File Date	This is the last modified Date of the attached File.
Person	This is the user who attached the File.
Added On	This is the Date (and Time) when the File was attached.
Note	This is text associated with the attached File.
Size KB	This is the Size of the attached File in Kilo Bytes (KB).
Upd by	This is the user who last updated the attached File.
Upd dt	This is the last update Date and Time of the attached File.

Insert_Attachment

Compatibility: Desktop App Version 4.20 and above.

This command fetches Attachment File Contents.
Only the following file type attachments can be inserted into a document:

- BMP

- GIF
- PNG
- WMF
- EMF
- JPEG
- JPG
- RTF
- TXT
- DOC
- DOCX

This is used under the Fetch_Attachments() Sub-report command.

\ Insert_Attachment ('<<Field Name of the file/attachment attached>>','<<Insert As Icon>>')

Parameter

<<Field Name of the file/attachment attached>>	Mandatory	Specify the Field Names for which you want the results to be displayed.
<<Insert As Icon>>	Optional	This is the Boolean parameter which takes the value as **TRUE** and **FALSE**. By default, the value of this parmeter is **FALSE**. If the value is **FALSE**, attachments will be embedded in the report. If the value is **TRUE**, attachments will be embedded as OLE icons.

Example

\Insert_Attachment(b : File Name)\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Use_Cases_By_Condition(‘ “State” = “All Open” ’ ,‘Name’)\
\scan(a)\

\a : Name\ [\ a : Id \]

\Fetch_Attachments(‘File Name’)\
\if (! eof(b))\

Attachments

\scan(b)\

\ b : File name \ \ b : Person \ \ b : added on \

\ Insert_Attachment (b : File Name) \
\endscan\
\endif\
\endscan\

Workflow Comments Sub-report Commands

Fetch_WorkFlow_Comments

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Workflow Comments records for the current primary record. Workflow Comments records are automatically created whenever the State (Status) or Owner/Assigned To fields of a Repository Objects or Tracking Items record is changed.

\Fetch_WorkFlow_Comments(‘<<Sort Order>>’)

Parameter

<< Sort Order>>	Optional

Example

\Fetch_WorkFlow_Comments()\

Fetch_WorkFlow_Comments_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Workflow Comments records for the current primary record based on the specified filter conditions (optional). Workflow Comments records are automatically created whenever the State (status) or Owner/Assigned To fields of a Repository Objects or Tracking Items record is changed.

\Fetch_WorkFlow_Comments_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional
<< Sort Order>>	Optional

Example

\Fetch_WorkFlow_Comments_By_Condition(' "Person" = "Me" ', 'Date desc') \

All fields of the Comments record are also available in the Workflow Comments record.

Fields Available

Field	Description
Comment	This is a Comment field.
Type	This is a simple Comment, state change event with optional Comments.
Person	This is the user who added the Comment.
Date	This is the Date and Time when the Comment was added.
Upd by	This is the user who last updated the Comment.
Upd dt	This is the last update date and time.

Old State	In case of a state change Comment, this field displays the State (Status) of the Record before the Comment was entered. Otherwise, this field is empty.
New State	In case of a state change Comment, this field displays the New State (Status) of the Record when the Comment was entered.
Old Asgnd To	In case of a State or Owner change Comment, this field displays the Owner (or Assigned To Person) of the Record before the Comment was entered.
New Asgnd To	In case of a State or Owner change Comment, this field displays the New Owner (or Assigned To Person) of the Record when the Comment was entered.

Version History Sub-report Commands

Fetch_Versions

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Version History records for the current primary record.

\Fetch_Versions('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional

Example

\Fetch_Versions ()\

Fetch_Versions_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Version History records for the current primary record based on the specified filter conditions (optional).

\Fetch_Versions_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Example

\Fetch_Versions_By_Condition(' "Comment" IS NOT NULL ')\

All fields of the Version History record may be inserted into the document.

Fields Available

Field	Description
Version	Version number .e.g. 1.72.
Date	This is the Date and Time when this Version of the Record was created.
Person	This is the user who created this Version.
Comment	This is the Comment associated with this Version of the record.
Keep	This is the Keep Flag. If the system is set to '**Y**' it will not allow the user to purge this Version of the Record.

Audit Log Sub-report Commands

Fetch_Audit_Log

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Audit Log records for the current primary record.

\Fetch_Audit_Log('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	

Example

\Fetch_Audit_Log()\

All fields of the Audit Log records may be inserted into the document.

Fields Available

Field	Description
Action	This is the Action performed by the user.
Person	This is the user who updated this Record.
Date	This is the Date and Time when this Record was updated.
Version	This is the Version of the record on which this update was performed. If this update resulted in the creation of a New Version, that New Version is listed here.
Keep	This is the Keep Flag. If the system is set to '**Y**' it will not allow the user to purge this Version of the Record.
Log Memo	This is the description of the aCtion performed.
Old Value	This is the Old Value of fields changed in this update.
New Value	This is the New Value of fields changed in this update.

Fetch_Audit_Log_Detail

Compatibility: Desktop App Version 6.20 and above.

This command displays the Audit Log details such as Old Values and its corresponding changed New Values in a record.

\Fetch_Audit_Log_Detail(<<Old Values Field>>,<<New Values Field>>)

Parameters

<<Old Values Field>>	Mandatory	Old Values field from the Audit Log Sub-report.
<<New Values Field>>	Mandatory	New Values field from the Audit Log Sub-report.

Fields Available

Field	Description
Old Value	
New Value	

Example

\Fetch_Audit_log_detail(b:Old Values ,b:New Values)\

Examples

\scan(a)\

\a:Name\ [\ a: Id\]

Fetch_Audit_Logs_By_Condition

Compatibility: Desktop App Version 8.50 and above.

This primary command fetches Audit Logs based on a filter condition. This is a generic function and can be used to fetch all types of tracking items. If the sort order is not specified, the 'Audit Log' records are sorted by 'Date'.

\Fetch_Audit_Logs_By_Condition('<<Filter Condition>>', '<<Sort_Order>>')

Parameters

<<Filter Condition>>	Optional	Specify the Filter condition as per the required values of the record. If the filter condition is not specified, all records will be fetched.
<<Sort Order>>	Optional	Specify the field names by which you want the results to be sorted. If the sort order is not specified, the data is sorted by 'Date'.

Fields Available

Field	Description
Person	
Action	
Old Values	
New Values	
Date	
Version	
Log Memo	

Examples

\Fetch_Audit_Logs_By_Condition(' "Person" = "User_A" ' ,'Date')\
\Fetch_Audit_Logs_By_Condition()\
\Fetch_Audit_Logs_By_Condition('' ,'Action')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Audit_Logs_By_Condition(' **"Person" = "User_A"** ' ,'Date')\
\scan(a)\

\a:Person\

Audit Logs

Person	Action	Old Values	New Values	Date	Version	Log Memo

\ a : Person \ \ a : Action \ \ a : Old Values\ \ a : New Values\ \ a : Date\ \ a : Version \ \a : Log Memo\

\endscan\

Audit Log

\Fetch_Audit_Log()\
\if (! eof(b))\
\scan(b)\

Action : \b: Action
Log : \b: Log memo**\Fetch_Audit_log_detail(b:Old Values, b:New Values)\\if (! eof(c))**

Field	Old Value	New Value

\scan(c)\

\ c : Field\	\ c : Old Value \	\ c : New Value \

\endscan\\endif\
\endscan\\endif\
\endscan\

Discussions Sub-report Commands

Fetch_Linked_Discussions

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Discussions records for the current primary record. By default the Discussions records are sorted in chronological order and display below the original post i.e. sequenced as threaded Discussions.

\Fetch_Linked_Discussions(<<Sort Order>>)

Parameter

<< Sort Order>>	Optional	

Example

\Fetch_Linked_Discussions()\

Fetch_Linked_Discussions_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Discussions records for the current primary record. By default, Discussions records are sorted in a chronological order and display below the original post i.e. sequenced as threaded Discussions.

\Fetch_Linked_Discussions(<< Filter Condition >>, << Sort Order >>)

Parameters

<< Filter Condition >>	Optional	
<< Sort Order>>	Optional	

Example

\Fetch_Linked_Discussions_By_Condition (' "Date" FALLS IN CURRENT MONTH ')\

All fields of Discussions records may be inserted into the document.

Fields Available

Field	Description
Date	This is the date of the Discussion Message.
From	This is the user who posted the Message.

Subject	This is the subject of the Message.
Message	This is the text of the Message. This should be inserted into the document using the InsertRTF command.
Priority	This is the Priority set for the Message.
Project	This is the Project in which the Message was posted.
Attachment ID	This is the identifier of the attached Record.
Attachment Name	This is the name of the attached Record.
Attachment Type	This is the Record Type of the attached Record.
To	
Cc	

Link Records Sub-report Commands

Fetch_Links

Compatibility: Desktop App Version 3.35 and above.

Secondary command fetches non-directional Link Records for the current primary record.

\Fetch_Links('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	

Example

\Fetch_Links('Type') \

All fields of Link Records may be inserted into the document.

Fields Available

Field	Description

Title Name	This is the title or name of the associated Record.
ID	
Record Type	e.g. Use Case, Issue, etc.
Type	Whether it is Link to or Duplicate of.
Project	
Note	
Upd dt	
Upd by	

Linked Screens Sub-report Commands

Checklist Sub-report Commands

Fetch_CheckList

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Checklist Records defined for the current primary record.

\Fetch_CheckList('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	Specify one or more field names separated by commas along with an ascending and descending indicator. By default, Checklist Items are ordered by Owner Checklist, Display Sequence, and then by Check Item.

Example

\Fetch_CheckList()\

All fields of a Checklist record may be inserted into the document.

Fields Available

Field	Description
Attribute Name	
Attribute Value	
Upd by	
Upd dt	
Visible	
Usage Hint	
Disp Seq	
Check Item	
Checklist	

Traceability Sub-report Commands

Traceability Sub-report Commands

These commands fetch outgoing or incoming Link Traceability Records for the current primary record.

Fetch_Traced_Records_of_Type

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Link Records for the current primary record. This is a very versatile command which can help you fetch Linked Records of any Link Type.

It allows you to filter Trace Link Records by their ID Prefixes, i.e. you can choose to fetch traces for the current Master record for a specific type. You can also filter by Link Types and States.

\Fetch_Traced_Records_of_Type('<<ID Prefix>>' ,' <<Trace Types comma separated>>', '<<State filter>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Optional	Specify the Record Type Prefix such as 'SCR', 'SCR' that appears in front of the identifier. For example, in the identifier SCR-2348, SCR is the ID Prefix and represents Screen Prototype. If ID Prefix is specified, then Record Type specific columns can be generated i.e. fields displayed in the editor for that Record Type. If ID Prefix is not specified, then all generic Objects Type fields can be generated i.e. fields displayed in the Objects list for all Repository Objects Record Types. If multiple ID Prefixes are specified, then all generic Objects Type fields can be generated i.e. fields displayed in the Objects list for all Repository Objects Record Types.
<<Trace Types comma separated >>	Optional	Specify comma-separated Link Type names.
<<State filter>>	Optional	This is the filter on Pseudo States such as <<All Closed>>, <<All Open>>, <<Any>>. To get All Closed state records you can supply the following combinations: '<<All Closed>>' '<All Closed>' 'All Closed' '<AllClosed' 'All Closed'

<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator. If this parameter is not specified, it will sort in the same order as the Traceability sub-tab, i.e. Trace Type, Record Type and Record ID.

Example

\Fetch_Traced_Records_of_Type('TC','Traces into','all open')\

Examples

\Fetch_Use_Cases_By_Condition()\
\ scan(a) \
\ a : Name \ [\ a : Id \ ]
\Fetch_Traced_Records_of_Type()\
\if (! eof(b))\
Traced Records For [\ a : Id \ ]
\scan(b)\
\ b : trace type \ \ b : type \ \ b : name \ \ b : Id \
\endscan\\endif\
\endscan\

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	If ID Prefix is specified, then Record Type specific columns can be generated i.e. fields displayed in the editor for that Record Type. If multiple/no ID Prefix is specified, then all generic Objects Type fields can be generated i.e. fields displayed in the Objects list for all Repository Objects Record Types.

Fetch_Traced_Records_of_Type_by_condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Link records for the current primary record. It allows you to filter Trace Link records by Record Type using ID Prefixes, Link Types and additional custom filters.

\Fetch_Traced_Records_of_Type_by_condition ('<<ID Prefix>>', '<<Link types comma separated >>','<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Optional	Specify the Record Type Prefix such as 'SMK', 'SMK' will appear in front of the identifier. For example, in the identifier SMK-2348, SMK is the ID Prefix and represents Screen Mockups. If ID Prefix is specified, that Record Type specific columns can be generated i.e. fields displayed in the editor for that Record Type. If ID Prefix is not specified, linked records of all Record Types will be fetched. Also, all generic Object Types fields can be generated i.e. fields displayed in Objects list for all Repository Object Record Types. If multiple ID Prefixes are specified, all generic Object Type fields can be generated i.e. fields displayed in Objects list for all Repository Object Record Types.
<< Link Types comma separated >>	Optional	Specify comma separated Link Type names. If Link Types are specified, then it will fetch linked records that has specified Link Types. E.g. If 'Traces Into' Link Type is specified, then records having 'Traces Into' link will be

		fetched. If Link Types are not specified, then it will fetch linked records having links of any type.
<< Filter Condition>>	Optional	If Filter Condition is specified, then it will fetch linked records which satisfy the specified filter condition. If Filter Condition is not specified, then all linked records will be fetched.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with ascending and descending indicators. If this parameter is not specified, it will sort in the same order as in the Traceability tab, i.e. Trace Type, Record Type, Record ID.

Example

\Fetch_Traced_Records_of_Type_by_condition('TC', 'Traces into',' "Priority" = "High" ')\

Examples

\Fetch_Use_Cases_By_Condition()\
\ scan(a) \
\ a : Name \ [\ a : Id \ ]
\Fetch_Traced_Records_of_Type_by_condition('FEAT','', ' "Priority" = "High" ')\
\if (! eof(b))\
Fetching record For [\ a : Id \ ]
\scan(b)\
\ b : trace type \ \ b : type \ \ b : Title \ \ b : Id \
\endscan\ \endif\
\endscan\

Fields Available

Field	Description

Trace Type	
Note	
Is Suspect	
Record Type specific fields	If ID Prefix is specified, then Record Type specific columns can be generated i.e. fields displayed in the editor for that Record Type. If multiple/no ID Prefix is specified, then all generic Objects Type fields can be generated i.e. fields displayed in the Objects list for all Repository Objects Record Types.

Fetch_Traced_Test_Cases

NOTE: This command "Fetch_Traced_Test_Cases" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Links of the Test Case type for the current primary record. It allows you to filter Trace Test Case Records by Link Types and Pseudo States. You can generate any Test Case fields once you fetch the Records using this command.

\Fetch_Traced_Test_Cases('<<Link Types comma separated>>', '<<State filter>>', '<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify the comma separated Link Type names for the desired Link Types.
<<State filter>>	Optional	This is the filter on Pseudo States such as, <<All Closed>>, <<All Open>>, <<Any>>. To get All Closed state records, you can use the following combinations:

		• '<<All Closed>>' • '<All Closed>' • 'All Closed' • '<AllClosed' • 'All Closed'
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of Test Case Record Type including custom fields, may be inserted into the document.

Fetch_Traced_Test_Cases_By_Condition

NOTE: This command "Fetch_Traced_Test_Cases_By_Condition" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Links of Test Case Type for the current primary record. This command allows you to filter Trace Test Case Records by Link Types and custom filters. You can generate any Test Case fields once you fetch the Records using this command.

\Fetch_Traced_Test_Cases_By_Condition ('<<Link Types comma separated>>', '<< Filter Condition >>', '<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify comma separated Link Type names for the desired Link Types.
<< Filter Condition>>	Optional	Specify the filter condition where you can use any Test Case fields.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Examples

\Fetch_Use_Cases()\
\scan(a) \
\ a : Name \ [\ a : Id \ ]
\Fetch_Traced_Test_cases_by_condition('Traces Into',' "State" = "Open" ')\
Traced Records
\scan(b)\
\ b : trace type \ \ b : type \ \ b : Title \ \ b : Id \
\endscan\
\endscan\

Fields Available

Field		Description
Trace Type		
Note		
Is Suspect		
Record Type specific fields		All fields of Test Case Record Type including custom fields, may be inserted into the document.

Fetch_Traced_Use_Cases

NOTE: This command "Fetch_Traced_Use_Cases" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Link of Use Case Type for the current primary record. This command allows you to filter Trace Use Case Records by Link Types and Pseudo States. You can generate any Test Case field once you fetch the Records using this command..

\Fetch_Traced_Use_Cases('<<Link Types comma separated>>', '<<State filter>>', '<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify comma separated Link Type names for desired Link Types.
<<State filter>>	Optional	This is the filter on Pseudo States such as, <<All Closed>>, <<All Open>>, <<Any>>. To get All Closed State records, you can use the following combinations: '<<All Closed>>' '<All Closed>' 'All Closed' '<AllClosed' 'All Closed'
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with ascending and descending indicator.

Fields Available

Field	Description
Trace Type	

Note	
Is Suspect	
Record Type specific fields	All fields of Use Case Record Type including custom fields, may be inserted into the document.

Fetch_Traced_Use_Cases_By_Condition

NOTE: This command "Fetch_Traced_Use_Cases_By_Condition" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Links of Use Case type for the current primary record. It allows you to filter Trace Use Case Records by Link Types and custom filters. You can generate any Test Case field once you fetch the Records using this command.

\Fetch_Traced_Use_Cases_By_Condition ('<<Link Types comma separated>>', '<< Filter Condition>>', '<<Sort Order>>')

Parameters

<<Link Types comma separated>>	Optional	Specify the comma separated Link Type names for the desired Link Types.
<< Filter Condition>>	Optional	Specify the filter conditions where you can use any Use Case field.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Examples

\Fetch_Use_Cases()\
\scan(a) \
\ a : Name \ [\ a : Id \ ]
\Fetch_Traced_Use_cases_by_condition('Traced From',' "Priority" = "Low" ')\

Traced Records

\scan(b)\

\ b : trace type \ \ b : type \ \ b : Name \ \ b : Id \

\endscan\

\endscan\

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	All fields of Use Case Record Type including custom fields, may be inserted into the document.

Fetch_Traceability_Associations

NOTE: This command "Fetch_Traceability_Associations" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Links for a current primary record. You can filter the links by Trace Direction (Forward or Reverse), Types of Links (Trace Into, Used In, etc.) and Record Type of Records using ID Prefixes.
If you specify a single ID Prefix, you can generate or sort any field of that Record Type.

\Fetch_Traceability_Associations('<<Trace Direction>>', '<< Link Type Names comma separated>>', '<< ID Prefix, comma separated>>',<<Sort Order>>')

Parameters

<<Trace Direction>>	Optional	Trace Direction values are 'Forward' or 'Reverse'. If you specify 'Forward' and Link Type as '<<include>>', it will fetch only "<<include>>" Links and NOT "<<included in>>" Links.

<< Link Types comma separated>>	Optional	Specify comma separated Link Type names.
<< ID Prefix, comma separated>>	Optional	Specify the Record Type Prefixes such as 'UC', 'FREQ' that appears in front of the identifier. For example, in the identifier UC-2348, UC is the ID Prefix and represents Use Case.
<<Sort Order>>	Optional	Specify one or more Record Type Field Names separated by commas along with an ascending and descending indicator. If you specify a single ID Prefix, you can use the field for that Record Type, i.e. if you specify ID Prefix as "UC", you can use any field of a Use Case for sorting.

Fields Available

Field	Description
Trace Type	
Note	
Is Suspect	
Record Type specific fields	If ID Prefix is specified, then all fields of that Record Type including custom fields, may be inserted into the document, i.e. the fields that are displayed in the editor of that Record Type. If ID Prefix is not specified or multiple ID Prefixes specified, then all common Repository Objects Types fields are available i.e. fields displayed in the Objects list.

Examples

Fetch_Traceability_Associations('','Traces into')

Fetch_Traceability_Associations('','Traced from')

NOTE: Both of the above commands will fetch the same records unless we write them in any of the following formats:

Fetch_Traceability_Associations('Forward','Traces into')
Fetch_Traceability_Associations('Reverse','Traced into')

Examples

\Fetch_Use_Cases ()\
\ scan(a) \
\a:Name\ \ a: Id\
\Fetch_Traceability_Associations()\
\if (! eof(b))\
Traceability Links
\scan(b)\
\ b : trace type \ \ b : type \ \ b : name \ \ b : Id \
\InsertRtf(b : Description)\
\endscan\
\endif\
\endscan\

Fetch_Traceability

NOTE: This command "Fetch_Traceability" has been deprecated. As an alternative, use command Fetch_Traced_Records_of_Type_by_condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Trace Links for the current primary record. This command doesn't give much flexibility to generate the report using various options; it is recommended to use the ones described above.

\Fetch_Traceability('<<Sort Order>>')\

Parameters

<<Sort Order>>	Optional	Specify one or more Repository Objects Field Names separated by commas along with an ascending and

descending indicator.

You can use the following fields for specifying the sort order:

Field Name	Description
Trace Type	Type of Trace Link
ID	Unique identifier
Suspect	Flag for Suspect **TRUE** or **FALSE** (Values = **Y/N**)
Object Name	Name of the Repository Object
Object Type	Record Type of the Repository Object
Project	Containing Project

By default, Records are sorted by Trace Type and then by Record Type of Traced Objects.

Example

\Fetch_Traceability('Object Type')\

Fields Available

Field	Description
Trace Type	This is the Type of Trace Link.
ID	This is the ID of the Traced Record.
Suspect	This Flag indicates if the Record is Suspect - **TRUE** or **FALSE** (Values = **Y/N**)
Object Name	This is the name of the Repository Object.
Object Type	This is the Record Type of the Repository Object.
Project	This is the containing Project.

Fetch Sub-report Commands for Repository Objects

Linked Tasks Sub-report Commands

Fetch_Linked_Tasks

NOTE: This command "Fetch_Linked_Tasks" has been deprecated. As an alternative, use the command Fetch_Linked_Issues_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Linked Task Records which are linked to the current primary record.

\Fetch_Linked_Tasks('<<Sort Order>>')

Parameter

<< Sort Order>>	Optional	

Example

\Fetch_Linked_Tasks('State, Asgnd To') \

Fetch_Linked_Tasks_By_Condition

NOTE: This command "Fetch_Linked_Tasks_By_Condition" has been deprecated. As an alternative, use the command Fetch_Linked_Issues_By_Condition.

This secondary command fetches Linked Task Records which are linked to the current primary record based on the specified filter conditions (optional).

\Fetch_Linked_Tasks_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Example

\Fetch_Linked_Tasks_By_Condition (‘ “priority” = “high” ’,’State, Asgnd To’)\

Fields Available

All fields of the Tracking Items Record Type including custom fields, may be inserted into the document.

Linked Issues Sub-report Commands

Fetch_Linked_Issues

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Linked Tracking Items (Issues, CRs, Tasks, etc.) Records which are linked to the current primary record.

\Fetch_Linked_Issues(‘<<Sort Order>>’)

Parameter

<< Sort Order>>	Optional	

Example

\Fetch_Linked_Issues()\

Fetch_Linked_Issues_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Linked Tracking Items (Issues, CRs, Tasks, etc.) Records which are linked to the current primary record, based on the specified filter conditions (optional).

\Fetch_Linked_Issues_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Example

\Fetch_Linked_Issues_By_Condition(' "State" = "Open" ')\

Fields Available

All fields of Tracking Items Record Type including custom fields, may be inserted into the document.

Tracking Items Commands

Fetch Tracking Items Commands

Fetch_Tracking_Item_By_ID

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches a single Tracking Item Record for the specified ID. If the Version Number is also specified, it fetches that specific version of the Tracking Item.

\Fetch_Tracking_Item_By_ID('<<ID>>', '<< Version Num >>')

Parameters

<< ID>>	Mandatory	Specify the Tracking Items identifier.
<< Version Num>>	Optional	Specify the Version Number you wish to fetch.

Example

\ Fetch_Tracking_Item_By_ID('ISS-2123')\

Fetch_Tracking_Items_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Tracking Items Records based on specified filter conditions (optional).

\ Fetch_Tracking_Items_By_Condition('<<ID Prefix>>', '<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<ID Prefix>>	Optional	Specify the Record Type Prefix such as 'TSK', 'ISS' that appears in front of the identifier. For example, in the identifier TSK-2348, CR is the ID Prefix and represents "Change Request".
<< Filter Condition>>	Optional	Specify the filter condition.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Examples

\ Fetch_Tracking_Items_By_Condition('ISS', ' "rslvd by" <> "User1" ', 'Severity')\

This fetches all Issues for the specified filters, sorted by Severity.

\ Fetch_Tracking_Items_By_Condition(‘’, ‘ “Due Dt” Next n weeks “1” ‘, ‘Due dt desc’) \

This fetches all Tracking Items Records that have a Due date in the next week, sorted by Due Date.

\ Fetch_Tracking_Items_By_Condition(‘’, ‘ ”Asgnd to” = “Me” ‘) \

This fetches all Tracking Items Records for the specified filters.

Fetch_Tracking_Items

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Tracking Items Records based on the specified filters (optional). This is a generic function and can be used to fetch any types of Tracking Items (E.g. Tasks, Change Requests, Issues, Problem Reports, etc.)

\Fetch_Tracking_Items(‘<<ID Prefix>>’, ‘<<Filter Name>>’, ‘<<Sort Order>>’)

Parameters

<<ID Prefix>>	Optional	The ID Prefix is the Record Type Prefix such as ‘ISS’, ‘CR’ that appears in front of the identifier. For example, in the identifier ISS-2342, ISS is the ID Prefix and represents Issues.
<< Filter Name>>	Optional	Specify the filters. The Filter Name specified here should be defined on the Tracking Items Grid/List interface.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Examples

\Fetch_Tracking_Items(‘ISS’, ‘All Unresolved Issues’, ‘Severity’) \

This command fetches all Issues Records for the specified filter All Unresolved Issues, and sorted by Severity.

\Fetch_Tracking_Items(‘’, ‘Items due next week’, ‘Due dt desc’) \

This command fetches all Tracking Items Records for the specified filters, sorted by the Due Date in a descending order.

\Fetch_Tracking_Items('', 'Assigned to Me') \

This command fetches all Tracking Items Records for the specified filter Assigned to Me.

Fetch_Item_Tree_Starting_With

This primary command fetches Tracking Items by hierarchy for the specified Start ID. This command should be used when you want to output Tracking Items and their corresponding WBS Codes.

\Fetch_Item_Tree_Starting_With('<<Starting Item ID>>', '<<Filter Name>>')

NOTE: This command will be available in a future release of DocProcessor.

Parameters

<<Starting Item ID>>	Mandatory	This is the identifier (ID) of the Starting Tracking Items Record from which the Items hierarchy has to be fetched.
<<Filter Name>>	Optional	Specify Filter Name from the filters defined on the corresponding tree interface, Tracking Items Tree or Tasks Tree.

Example

\Fetch_Item_Tree_Starting_With('TSK-1209','') \

Tracking Items Sub-Reports

Fetch_Linked_Impacts

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches child Tracking Items for current primary Tracking Items Records.

\Fetch_Linked_Impacts('<<Sort Order>>')

Parameter

<<Sort Order>>	Optional	Specify one or more Tracking Items Field Names separated by commas along with an ascending and descending indicator.

Example

\Fetch_Linked_Impacts()\

Fields Available

All fields of Tracking Items Record Type including custom fields, may be inserted into the document.

Fetch_Linked_Impacts_By_Condition

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches child Tracking Items for current primary Tracking Items Records. Specify the filter conditions if you wish to fetch specific Tracking Items.

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

Parameters

<< Filter Condition>>	Optional	Specify the filter conditions where you can use any Tracking Items field.
<< Sort Order>>	Optional	Specify one or more Field Names separated by commas along with an ascending and descending indicator.

Example

\Fetch_Linked_Impacts_By_Condition (‘ “State” = “Open” ’)\

Fields Available

All fields of the Tracking Items Record Type including custom fields, may be inserted into the document.

Test Case Commands

Fetch_Test_Cases_By_Condition

NOTE: This command "Fetch_Test_Cases_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 3.35 and above.

This primary command fetches Test Case Records based on the specified filter conditions (optional).

\Fetch_Test_Cases_By_Condition ('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Example

\Fetch_Test_Cases_By_Condition (' "Last Result" = "Passed" ')\

Fetch_Bugs_Reported_For_Test_Run

Compatibility: Desktop App Version 10 and above.

This secondary command fetches bugs reported for a specific Test Run.

\Fetch_Bugs_Reported_For_Test_Run ('<<Open Only>>', '<<Sort Order>>')

Parameters

<<Open Only>>	Optional	True / False – Specify whether to fetch defects which are in Open State or all reported ones.
<< Sort Order>>	Optional	Specify the Field Names by which you want the results to be sorted.

Example

\Fetch_Bugs_Reported_For_Test_Run ('True', 'ID')\

Fields Available

All Defect (PR) record type fields are available for this command.

Sample Template 1

\Fetch_Test_Runs_By_Condition('')\
\scan(a)\

**\a : ID\ - \ a : Title \ - State: \a: State\ Result: \a: Result **

[\Insert_Permalink(a: ID)\]

Crt Dt : \ a: Crt dt\ **Target Release:** \ a: Target Release \
Priority: \a: Priority\ **Assigned To** : \a: Assigned To\
Start Dt: \a: Planned Start Date\ **End Date:** \a: Planned End Date\
Test Configuration: \a: Test Configuration\
Run By: \a: Run by\
Reported Defects

Defect	Reported for Test Case	Status	Detected by	Assigned to	Submit Date

\scan(b)\
\if(!eof(a))\

\b : Title\ [\Insert_Permalink(b: ID)\]	\Insert_Permalink(a : Linked Object ID, b : Linked Object)\	\b : State\	\b : Crt by\	\b : Asgnd to\	\b : Submit dt \

\endif\

\endscan\

\endscan\

Sample Template 2

\Fetch_Test_Runs_By_Condition('Result = Failed','Priority','TS-1234')\

Test Run

\scan(a)\

\ a : Title \

\ a : Id \

Owner: \ a : owner \	**State:** \ a : State\

Bugs Reported

\ Fetch_Bugs_Reported_For_Test_Run('true', 'Id')\

ID	Title	Target Release	State	Assigned To
\scan(b)\				
\ b : Id\	\ b : Title \	\ b: Target Release \	\ b: state\	\ b: Asgnd To \

\endscan\

\endscan\

Fetch_Test_Cases_For_Test_Set

Compatibility: Desktop App Version 10 and above.

This secondary command fetches Test Cases for a Test Set.

\Fetch_Test_Cases_For_Test_Set('<<Sort Order>>')

Parameters

<< Sort Order>>	Optional	Specify the Field Names by which you want the results to be sorted.

Example

\Fetch_Test_Cases_For_Test_Set ('Title')\

Fields Available

Field	Description
Title	
Description	
Level	
Type	
Crtd By	
Crt Dt	

Sample Template 1

\Fetch_Records_By_Folder_By_Condition('TS')\

Test Set

\scan(a)\

\ a : Title \

\ a : Id \

Owner: \ a : owner \	**State:** \ a : State\

Test Cases

\ Fetch_Test_Cases_For_Test_Set ()\

ID	Title	Target Release	State	Owner

\scan(b)\
\if (! eof(b))\

\ b : ID \	\ b : Title\	\ b : Target Release\	\ b : State \	\ b :Owner\

\endif\

\endscan\
\endscan\

Sample Template 2

\Fetch_Test_Sets_By_Conditions(' "id" = " TS-893" ','Title')\
\scan(a) \

\a:Title\ - [\Insert_Permalink(a: Id)\]

Test Cases -

\Fetch_Test_Cases_For_Test_Set ()\
\scan(b)\
\if(!eof(b))\
\if(b : Level = 1)\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\elsif(b : Level = 2)\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\elsif(b: Level = 3)\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\elsif(b: Level = 4)\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\elsif(b: Level = 5)\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\else\

\b : Title\ - [\Insert_Permalink(a: Id)\]

\endif\

\endif\

\endscan\

\endscan\

Fetch_Test_Results_By_Condition

Compatibility: Desktop App Version 10 and above.

This secondary command fetches Test Results as per the specified condition.

\Fetch_Test_Results_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<Filter Condition>>	Optional	Filter condition as per the required values of a record. If the condition is not specified, all records will be fetched.
<< Sort Order>>	Optional	Specify the Field Names by which you want the results to be sorted.

Example

\Fetch_Test_Results_By_Condition ('Result = Passed')\

Fields Available

Field	Description
Executed	
Title Result	

Run By	
Execution Date	
Defects	
Test Run	
Test Run Result	
Run Date	
Run End Date	
Test Set	
Created By	
Enabled	
Updt By	
Updt Dt	

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Test_Runs_By_Condition('')\
\scan(a)\

**\a : ID\ - \ a : Title \ - State: \a: State\ Result: \a: Result **

[\Insert_Permalink(a: ID)\]

Crt Dt : \ a: Crt dt\ **Target Release:** \ a: Target Release \
Priority: \a: Priority\ **Assigned To** : \a: Assigned To\
Start Dt: \a: Planned Start Date\ **End Date:** \a: Planned End Date\
Test Configuration: \a: Test Configuration\
Run By: \a: Run by\

\Fetch_Test_Results_by_Condition('Result = Passed')\

Executed	Title	Result	Executed by	Execution dt
\scan(b)\ \if (! eof(b))\				
\ b : Executed\	\ b : Title \	\b: Result \	\ b: Executed by\	\ b: Type\

\endif\ \endscan\\endscan\

Sample Template 2

Test Run

\scan(a) [,page] \

\ a : Title \

\ a : Id \

Owner: \ a : owner \	**State:** \ a : State\

Test Results

\ Fetch_Test_Results_By_Condition ('Result = Passed')\

Executed	Title	Result	Run By	Defects
\scan(b)\ \if (! eof(b))\				
\ b : Executed \	\ b : Title\	\ b : Result\	\ b : Run By \	\ b :Defects\

\endif\
\endscan\
\endscan\

Fetch_Test_Case_Steps

Compatibility: Desktop App Version 10 and above.

This secondary command fetches Test Case's steps.

\Fetch_Test_Case_Steps('<<Steps Field>>')

Parameters

<< Steps Field>>	Mandatory	This Field contains Steps data.

Example

\Fetch_Test_Case_Steps(b : Steps)\

Fields Available

Field	Description
Step No.	
Action	
Expected Result	
Custom Column	Any Custom Column that you have added in the Test Case Steps table.
PassFail & Actual Result	If you are generating Test Result, then "PassFail & Actual Result" columns can also be generated

Sample Template 1

\Fetch_Test_cases_By_Condition()\

Test Cases

\scan(a) \

\a:Title - [\Insert_Permalink(a: Id)\]

\if(Is_Field_Non_Empty(a:Steps))\

Test Case Steps -

\Fetch_Test_Case_Steps(a : Steps)\

Step No.	Action	Test Data	Expected Result

\scan(b)\

\ b : Step No\	\ InsertRtf (b : Action)\	\InsertRtf (b : Test Data)\	\InsertRtf (b : Expected Result)\

\endscan\
\endif\
\endscan\

Sample Template 2

\Fetch_Test_Results_In_Test_Run('TR-1234')\

Test Results

\scan(a) \

\a:Title - [\Insert_Permalink(a: Id)\]
**\a: Result **

\if(Is_Field_Non_Empty(a:Steps))\

Test Case Steps -

\Fetch_Test_Case_Steps(a : Steps)\

Step No.	Action	Test Data	Expected Result	Result	Actual Result

\scan(b)\

\ b : Step No\	\ InsertRtf (b : Action)\	\InsertRtf (b : Test Data)\	\InsertRtf (b : Expected Result)\	\b : Result\	\InsertRtf(b : Actual Result)\

\endscan\
\endif\
\endscan\

Fetch_Test_Runs_By_Condition

NOTE: This command "Fetch_Test_Runs_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 10 and above.

This primary command fetches Test Runs in system as per the specified condition.

\Fetch_Test_Runs_By_Condition('<<Filter Condition>>','<<Sort Order>>','<<Test Set ID>>')

Parameters

<<Filter Condition>>	Optional	Filer condition as per the required values of a record. If the condition is not specified, all records will be fetched.
<<Sort Order>>	Optional	Specify Field Names by which you want the results to be sorted.
<<Test Set ID>>	Optional	Specify Test Set ID which will fetch Test Runs of only particular Test Set ID.

Example

\Fetch_Test_Runs_By_Condition(' "Result" ="Passed" ', 'Title')\

Fields Available

Field	Description
Id	
Title	
Assigned To	
Result	
Planned Start Date	
Planned End Date	
Run By	
Remarks	
Run Date	
Project	
Priority	
Stability	
State	
Target Release	
Updt By	
UPdt Dt	
Crt By	
Crt Dt	

Sample Template 1

```
\Fetch_Test_Runs_By_Condition(‘ "State" = "Completed" ')\
\scan(a)\
```

\a : ID\ - \ a : Title \ - State: \a: State\ Result: \a: Result \

[\Insert_Permalink(a: ID)\]

Crt Dt : \ a: Crt dt\ **Target Release:** \ a: Target Release \
Priority: \a: Priority\ **Assigned To** : \a: Assigned To\
Start Dt: \a: Planned Start Date\ **End Date:** \a: Planned End Date\
Test Configuration: \a: Test Configuration\
Run By: \a: Run by\

\endscan\

Sample Template 2

Test Runs

\Fetch_Test_Runs_By_Condition(' "State" = "Completed" ','Priority','TS-3401')\

Contents

Fetch_Test_Sets_By_Condition

NOTE: This command "Fetch_Test_Sets_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 10 and above.

This primary command fetches Test Sets in system as per the specified condition.

\Fetch_Test_Sets_By_Condition('<<Filter Condition>>','<<Sort Order>>')

Parameters

<<Filter Condition>>	Optional	Filer condition as per the required values of a record. If the condition is not specified, all records will be fetched.
<<Sort Order>>	Optional	Specify Field Names by which you want the results to be sorted.

Example

\Fetch_Test_Sets_By_Condition(' Project = Megaflix', 'Title')\

Fields Available

Field	Description
Id	
Title	
Description	
Project	
State	
Updt By	
UPdt Dt	
Crt By	
Crt Dt	

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Test_Sets_By_Condition(' "id" = "TS-893" ','Title')\

Test Sets

ID	Title	Project	Crt by	Upd dt

\scan(a)\

\ a : ID\	\ a : Title \	\a: Project \	\ a: Crt by\	\ a: Upd dt \

\endscan\

Sample Template 2

\Set_Project('$CURRENT_PROJECT$')\

\Insert_Company_Logo('',4,4, 'INCHES')\

\ PROJECT_NAME \

Test Sets

\Fetch_Test_Sets_By_Condition('Project = Megaflix','Title')\

Contents

Test Sets

ID	Title	Project	Crt by	Upd dt

\scan(a)\

\if (! eof(a))\

\ a : ID\	\ a : Title \	\a: Project \	\ a: Crt by\	\ a: Upd dt \

\endif\

\endscan\

Fetch_Bugs_Reported_For_Test_Result

Compatibility: Desktop App Version 10 and above.

This secondary command fetches bugs reported for a specific Test Result.

\Fetch_Bugs_Reported_For_Test_Result('<<Open Only>>','<<Sort Order>>')

Parameters

<<Open Only>>	Optional	True / False – Specify whether to fetch defects which are in Open State or all reported ones.
<<Sort Order>>	Optional	Specify Field Names by which you want the results to be sorted.

Example

\Fetch_Bugs_Reported_For_Test_Result( )\

Fields Available

All Defect (PR) record type Fields are available for this command.

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Test_Runs_By_Condition('')\
\scan(a)\

**\a : ID\ - \ a : Title \ - State: \a: State\ Result: \a: Result **

[\Insert_Permalink(a: ID)\]

Crt Dt : \ a: Crt dt\ **Target Release:** \ a: Target Release \
Priority: \a: Priority\ **Assigned To** : \a: Assigned To\
Start Dt: \a: Planned Start Date\ **End Date:** \a: Planned End Date\
Test Configuration: \a: Test Configuration\

Run By: \a: Run by\

\Fetch_Test_Results_In_Test_Run(a:id)\

Executed	Title	Result	Executed by	Execution dt

\scan(b)\
\if (! eof(b))\

\ b : Executed\	\ b : Title \	\b: Result \	\ b: Executed by\	\ b: Type\

\Fetch_Bugs_Reported_For_Test_Result('true', 'id')\

\scan(c)\

**Bugs Reported For Test Result : \ b : Title **

[\Insert_Permalink(c: ID)\] - \ c : Title \
Assigned To : \ c: Asgnd To \

State : \c : State\

\endscan\
\endif\ \endscan\\endscan\

Sample Template 2

\Fetch_Test_Results_In_Test_Run ('TEXE-1234')\

Test Result

\scan(a)\

\ a : Title \

\ a : Id \

Owner: \ a : owner \	**State:** \ a : State\

Bugs Reported

\ Fetch_Bugs_Reported_For_Test_Result('true', 'ID')\

Id	Title	Assigned To	Target Release	State
\scan(b)\				
\ b : Id\	\ b : Title \	\ b: Asgnd To \	\ b: Target Release\	\ b: State\

\endscan\
\endscan\

Fetch_Test_Results_In_Test_Run

Compatibility: Desktop App Version 10 and above.

This primary command fetches Test Results in Test Run.

\Fetch_Test_Results_In_Test_Run('<<Test Run ID>>')

Parameters

<<Test Run ID>>	Mandatory	Fetches Test Results in Test Run by Test Run's ID

Example

\Fetch_Test_Results_In_Test_Run('TR-4356')\

Fields Available

Field	Description
Executed	
Title	
Description	
Result	
Steps	

Remarks	
Pre Condition	
Post Condition	
Run By	
Crt by	
Upd by	
Crt dt	
Upd dt	
Start Date	
Run Date	
Run End Date	
Defects	
Test Run	
Test Set	
State	
Type	
ID	
Test Run Result	
Execution Date	

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Test_Runs_By_Condition('')\
\scan(a)\

\a : ID\ - \ a : Title \ - State: \a: State\ Result: \a: Result \

[\Insert_Permalink(a: ID)\]

Crt Dt : \ a: Crt dt\ **Target Release:** \ a: Target Release \

Priority: \a: Priority\ **Assigned To** : \a: Assigned To\

Start Dt: \a: Planned Start Date\ **End Date:** \a: Planned End Date\

Test Configuration: \a: Test Configuration\

Run By: \a: Run by\

\Fetch_Test_Results_In_Test_Run(a:id)\

Executed	Title	Result	Executed by	Execution dt
\scan(b)\ \if (! eof(b))\				
\ b : Executed\	\ b : Title \	\b: Result \	\ b: Executed by\	\ b: Type\

\endif\ \endscan\\endscan\

Sample Template 2

Test Results

\Fetch_Test_Results_In_Test_Run('TR-4356')\

Contents

Test Results

Executed	Title	Result	Executed by	Execution dt	Defects

\scan(a)\

\if (! eof(a))\

\ a : Executed \	\ a : Title \	\a: Result \	\ a: Executed by\	\ a: Run End Date\	\a: Defects\

\endif\

\endscan\

Project Based Commands

This Project-based command will generate Project details for the current or supplied Project.

Fetch_Projects

Compatibility: Desktop App Version 6.20 and above.

This command is a Master level command used to fetch the Project details for all the Projects available in a database.

\Fetch_Projects()

Parameters

There are no parameters for this command.

Fields Available

Field	Description
ID	
Project	
Project Path	Compatibility: 8.11 and above
Project Manager	

Description	
Plnd Strt dt	
Plnd End dt	
Act Strt dt	
Act End dt	
Est Cost	
Act Cost	
Crt by	
Upd by	
Crt dt	
Upd dt	
Frozen	
Deleted	
Enforce State Transition Rules	
Enforce Security Rules on the Project	

Example

\Fetch_Projects()\

Examples

\Fetch_Projects()\
\scan(a)\

ID: \ a: ID\

Project Name: \a:project\

Project Manager: \ a: Project Manager\

Description: \a: description\

\endscan\

Fetch_Project_By_ID

Compatibility: Desktop App Version 4.20 and above.

This Master level command fetches Project details based on the specified Display ID.

\Fetch_Project_By_ID('<<Project display ID>>')

Parameter

<<Project Display ID>>	Mandatory	This is the Project identifier or Display ID of a particular Project.

Example

\Fetch_Project_By_Id('$CURRENT_PROJECT$')\
\ Fetch_Project_By_Id('PRJ-141')\

Fields Available

Field	Description
ID	
Project	
Project Manager	
Description	
Plnd. Strt. dt	
Plnd. End dt	
Act. Strt dt	
Act. End dt	
Est Cost	

Act Cost	
Crt by	
Upd by	
Crt dt	
Upd dt	
Frozen	
Deleted	
Enforce State Transition Rules	
Enforce Security Rules on the Project	

Examples

\Fetch_Project_By_ID('PRJ-1000')\
\Fetch_Project_By_ID('$CURRENT_PROJECT$')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Project_By_ID('$CURRENT_PROJECT$')\
\scan(a)\

Id: \ a: ID\
Project Name: \a:project\
Project Manager: \ a: Project Manager\
Description: \a: description\

\endscan\

Sample Template

\Fetch_Project_By_ID('PRJ-1000')\
\scan(a)\

ID: \ a: ID\
Project Name: \a:project\
Project Manager: \ a: Project Manager\
Description: \a: description\

\endscan\

Fetch_Project_Inclusion_Types

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Record Types for a Project which were fetched using the Fetch_Project_By_ID command.

\Fetch_Project_Inclusion_Types ('<<Project ID Field>>')

Parameter

<<Project IDField>>	Mandatory	Specify the ID Field of the Project for which you want to fetch Record Types.

Example

\Fetch_Project_Inclusion_Types (a: ID)\

Examples

\ Fetch_Project_By_Id('PRJ-141')\
\scan(a) \
\a: Project\
\Fetch_Project_Inclusion_Types (a: ID)\

Record Type	Display Seq.	Security Enforced	Versioned	WBS Start No.	Hidden
\scan(b) \					
\ b : Record Type\	\b:	\if (b :	\if(b :	\ b : WBS	\if(b :

Display Seq\	Security Enforced = 'Y')\Y \endif\	Versioned = 'Y')\Y\endif\	Start No\	Hidden= 'Y')\Y\endif\

\endscan\
\endscan\

Fields Available

Field	Description
Record Type	
Security Enforced	
Versioned	
Hidden	
Display Seq.	
WBS Start No.	
Crt by	
Upd by	
Upd dt	

Fetch_Project_Team_Members

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Team Members for a Project, which were fetched using the Fetch_Project_By_ID command. It is a sub report level command which is used to fetch Team Member details of a specified project.

\Fetch_Project_Team_Members ('<<Project ID Field>>')\', << Sort_Order >>')

Compatibility: Desktop App Version 9.5 and above.

\Fetch_Project_Team_Members ('<<Project ID Field>>')\', << Sort_Order >>', '<< Is_Show_User >>')

Parameters

<<Project ID Field>>	Mandatory	Specify the ID Field of the Project for which you want to fetch Team Members.
<<Sort_Order>>	Optional	Specify the field name by which you want to sort the Team Members.
<< IS_Show_User >>	Optional	Team Members can be Users or User Groups. User Groups contain one or more users known as Group Members. Default value of this parameter is **FALSE** and will display User Groups as it is. If its value is set to **TRUE**, then it will display User Group Members instead of User Groups.

Fields Available

Field	Description
User Name	
Inactive	
User Type	

Examples

\ Fetch_Project_Team_Members (a: ID)\

\ Fetch_Project_By_Id('PRJ-141')\
\scan(a) \

\a:Project\

\Fetch_Project_Team_Members (a: ID)\
\if (! eof(b))\

Team Members

\scan(b)\

\b: User Name\

\endscan\
\endif\
\endscan\

\Fetch_Project_Team_Members('PRJ-1000')\
\Fetch_Project_Team_Members(a: ID, 'User Name', True)\

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\
\PROJECT_NAME\
\Fetch_Project_By_Id('$CURRENT_PROJECT$')\
\scan(a)\

Project : \a:Project\
Id: \a: Id\

\Fetch_Project_Team_Members(a: ID, 'User Name')\

Team Members

\scan(b)\
\b : User Name\
\endscan\
\endscan\

Sample Template 2

\Set_Project('$CURRENT_PROJECT$')\
\PROJECT_NAME\
\Fetch_Project_By_Id('$CURRENT_PROJECT$')\
\scan(a)\

Project : \a:Project\
Id: \a: Id\

\Fetch_Project_Team_Members(a: ID, 'User Name', True)\

Team Members

\scan(b)\

\b : User Name\

\endscan\
\endscan\

Fetch_Roles

Compatibility: Desktop App Version 8.15 and above.

This command is used to fetch all Roles. This is the Master as well as a Sub-report command.

\Fetch_Roles_('<<Sort_Order>>')

Parameter

<<Sort_Order>>	Optional	Specify the Field Name by which you want to sort the Records.

Fields Available

Field	Description
Role	
Description	
System	
Rol Crt dt	
Rol_Crt_by_Usr_Name	
Rol_Deleted	
Rol_Rec_Type	
Rol_System	

Examples

\ Fetch_Roles()\
\ Fetch_Roles('Role')\

\Fetch_Roles('')\

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

Roles

\ Fetch_Roles('Role')\

Role Name	Description
\scan(a)\	
\a: Role\	\a: Description\

Privileges

\ Fetch_Privileges_Granted_To_Roles(a:Role)\

Record Type Name	Action Type	Privileges
\scan(b)\		
\ b : Record Type\	\b: Action Type\	\b: action\

\endscan\

\endscan\

Fetch_Roles_Granted_To_Team_Members

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Roles Grants to Team Members for a project which were fetched using the Fetch_Project_By_ID command.

\Fetch_Roles_Granted_To_Team_Members ('<<Project display Id>>', '<< Sort_Order >>')

Compatibility: Desktop App Version 9.5 and above.

\Fetch_Roles_Granted_To_Team_Members ('<<Project display Id>>', '<< Sort_Order >>', '<< Is_Show_User >>')

Parameters

<<Project ID Field>>	Mandatory	Specify the ID Field of a Project for which you want to fetch Team Member's Roles Grants.
<<Sort_Order>>	Optional	Specify the field name by which you want to sort the Role Grants.
<< IS_Show_User >>	Optional	Team Members can be Users or User Groups. User Groups contain one or more users known as Group Members. Default value of this parameter is **FALSE** and will display User Groups as it is. If this value is set to **TRUE**, then it will display User Group Members instead of User Groups.

Fields Available

Field	Description
User	
Role	
Role ID	
User Type	

Example

\ Fetch_Roles_Granted_To_Team_Members (a: ID)\
\if (! eof(b))\

User Name Role Name

\scan(b)\
\if (! eof(b))\

\b: User\ - \b:Role\

\endIf\\endscan\\endIf\
\Fetch_Roles_Granted_To_Team_Members('PRJ-1000')\

Sample Template

\Fetch_Project_By_ID('PRJ-1000')\
\scan(a)\

Project Name: \a:project\
Project Manager: \ a: Project Manager\
Id: \ a: ID\
Description: \a: description\

\Fetch_Roles_Granted_To_Team_Members(a: ID)\
\if (! eof(b))\

User Name	Role Name
\scan(b)\ \if (! eof(b))\	
\b: User\	\b:Role\
\endIf\ \endscan\ \endIf\ \endscan\	

Fetch_Privileges_Granted_To_Roles

Compatibility: Desktop App Version 6.50 and above.

This command fetches all Privileges granted to particular Roles. This is a master, as well as sub report level command.

\Fetch_Team_Members_Privileges('<<Project Id >>','<<User Name>>', '<< Is_Show_User >>', '<<Sort_Order>>')

Parameters

<<Role Name>>	Mandatory	This fetches all the Privileges granted to a specified Role.
<<Sort Order>>	Optional	Specify the field names by which you want to sort the records. If not specified, the data is not sorted.

Fields Available

Field	Description
Record Type	
User Name	
Action	
Action Type	

Examples

\ Fetch_Privileges_Granted_To_Roles('QA Manager')\
\ Fetch_Privileges_Granted_To_Roles(b:Role, 'Record Type')\

Fetch_Team_Members_Privileges

Compatibility: Desktop App Version 4.20 and above.

This secondary command fetches Privileges granted to Team Members of a project which were fetched using the Fetch_Project_By_ID command.

\Fetch_Team_Members_Privileges ('<<Project Id >>', '<<User Name>>', '<<Sort_Order>>')

Compatibility: Desktop App Version 9.5 and above.

\Fetch_Team_Members_Privileges ('<<Project Id >>', '<<User Name>>', '<<Sort_Order>>')

Parameters

<<Project ID Field>>	Mandatory	Specify the ID of the Project for which you want to fetch Team Member Privileges.
<<User Name Field>>	Optional	This is the User Name Field that is fetched with the Fetch_Project_Team_Members command. If this parameter is not supplied, then it will fetch Privileges for all Team Members in the Project.
<<Sort_Order>>	Optional	Specify the field name by which you want to sort the Team Member Privileges.
<< IS_Show_User >>	Optional	Team Members can be Users or User Groups. User Groups contain one or more users known as Group Members. Default value of this parameter is **FALSE** and will display User Groups as it is. If this value is set to **TRUE**, then it will display User Group Members instead of User Groups.

Fields Available

Field	Description
Record Type	
User Name	
Action	
Action Type	
User Type	

Example

\ Fetch_Team_Members_Privileges(a: ID)\

\ Fetch_Team_Members_Privileges(a: ID, b: Name)\

\ Fetch_Project_By_Id('735')\
\scan(a) \

\a:project\
Id: \ a: PRJ_DISP_ID\
\Fetch_Team_Members_Privileges(a: ID)\

\if (! eof(b))\

Team Member's Privileges
User Name Record Type Name Privilege

\scan(b)\

\b:User Name\ \ b : Record Type\ \b: action\

\endscan\\endif\
\endscan\

\Fetch_Team_Members_Privileges('PRJ-1000', '')\
\Fetch_Team_Members_Privileges('PRJ-1000', b:user name)\
\Fetch_Team_Members_Privileges(a: ID, b: User Name, '', 'Record Type')\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Project_By_Id('$CURRENT_PROJECT$')\
\scan(a)\

Project Name: \a:project\
Project Manager: \ a: Project Manager\
Id: \ a: ID\
Description: \a: description\
\Fetch_Team_Members_Privileges(a: ID, '')\

\if (! eof(b))\

Record Type Name	Action Type	Privileges

\scan(b)\

\if (! eof(b))\

\ b : Record Type\	\b: Action Type \	\b: Action\

\endIf\

\endscan\

\endIf\

\endscan\

Fetch_Project_Baselines

Compatibility: Desktop App Version V8 and above.

This command fetches Baselines created for a specified Project.

\Fetch_Project_Baselines()\

Parameters

There are no parameters for this command.

Fields Available

Field	Description
Baseline	Baseline name
Crt dt	Baseline Created date
Crt by	Baseline Created by
Upd dt	Baseline Updated date
Upd by	Baseline Updated by
Baseline Type	Baseline Type – Complete or Selective
Baseline for	Baseline for – Project or Requirements Document
Baseline Date	Baseline Created for Date
Project Name	Project Name

Example

\ Fetch_Project_Baselines()\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Generates Project Baselines for the specified Project\

Project Baselines

\Declare_Variable_Project('variable_Project1', 'Select Project')\

\Prompt_For_Variable_Values('ALL')\

\Set_Project(variable_Project1)\

\Fetch_Project_Baselines()\
\if (! eof(a))\

Baseline	Crt dt	Crt by

\endif\
\scan(a)\

\a:Baseline\	\a:Crt dt\	\a:Crt by\

\endscan\

User Based Commands

Fetch_Users

Compatibility: Desktop App Version 7.0 and above.

This command fetches all TopTeam users. Users will be sorted by the Display Name field.

\Fetch_ Users()

Parameters

There are no parameters for this command.

Fields Available

Field	Description
Details	
Username	This is the Username that is used for login. This name is not used anywhere else.
Display Name	This is the name that appears in the system elsewhere. E.g. Created by, Updated by, Owner, etc.
Is System Administrator	If a user that has been granted System Administrator privileges, this field will have the values "**Y**", else "**N**".
Authentication Type	The possible values for this field are "**Native Authentication**" and "**LDAP Authentication**".
License Type	This field has a value if the user has concurrent or name licenses type assigned. The value will be "**Y**" if the user has been assigned a concurrent license type. The value will be "**N**" if the user has been assigned a name license type.
Email Id 1	
Use Email1 for notifications	This field value indicates that Email ID1 will be used for notifications.
Email Id 2	
Use Email2 for notifications	This field value indicates that Email ID2 will be used for notifications.
Is External	If a user is an External User, this field will display the value "**E**". If it is an internal TopTeam user, its value will be "**I**".
Inactive	
Disable Login	
Personal Info	
First name	

Mid Name	
Last name	
Job Title	
Is Consultant	
Address	
Address	
City	
State	
Country	
Zip	
Phone	
Office	
Extn	
Home	
Mobile	
Pager	
Fax	
Description	
Description	
Create date	
Created by	
Update date	
Updated by	

Examples

\Fetch_ Users()\
\scan(a)\

User: \ a : Display Name \ [\ a : Username\]
Is System Administrator: \if(a : System Administrator = 'Y')\Yes\elseif(a : System Administrator = 'N')\No\endif\
Type: **\if(a : Is External = 'E')**External**\elseif(a : Is External = 'I')**Internal**\endif**

\endscan\

Fetch_System_Privileges

Compatibility: Desktop App Version 7.0 and above.

This command fetches System Privileges granted to supplied users.

\Fetch_System_Privileges('<<Username>>')

Parameter

<<Username>>	Mandatory	This is the user login name or Username field from Fetch Users results.

Fields Available

Field	Description
Privilege Name	
Description	Privilege Description

Example

\Fetch_ Users()\
\scan(a)\

User: \ a : Display Name \ [\ a : Username\]

\if(a : System Administrator = 'N')\ \ Fetch_System_Privileges(a: Username)\

Privileges:

\scan (b)\

\ b : Privilege Name \

\endscan\
\endscan\

Fetch_Projects_For_User

Compatibility: Desktop App Version 6.50 and above.

This Master command fetches all those Projects where the specified user is a Team Member. It can also be used as a Sub-report level command.

\Fetch_Projects_For_User('<<User Name>>')

Parameter

<<User Name>>	Mandatory	This fetches all Projects where the specified user is a Team Member.

Fields Available

Field	Description
ID	
Project	
Project Manager	
Description	
Plnd Strt dt	
Plnd End dt	
Act Strt dt	
Act End dt	
Est Cost	
Act Cost	

Crt by	
Upd by	
Crt dt	
Upd dt	
Frozen	
Deleted	
Enforce State Transition Rules	
Enforce Security Rules on the Project	

Example

\Fetch_Projects_For_User('John DBA')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\ PROJECT_NAME \
John DBA
\Fetch_Projects_For_User('John DBA')\
\scan(a)\
Project: \a:Project\ [\ a: ID\]
\endscan\

Fetch_User_Groups

Compatibility: Desktop App Version 9.5 and above.

This command shows User Groups created in the system and is used at master level with scan loop.

\Fetch_User_Groups()

Parameters

There are no parameters for this command.

Fields Available

All fields from User Group Editor are available in this command.

Example

\Fetch_User_Groups()\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\
\ PROJECT_NAME \

User Groups

\Fetch_User_Groups()\

Name	Member Count	Inactive

\scan(a) \

\a: Name\	\ a : Users \	\ a : Inactive \

\endscan\

Fetch_User_Group_Members

Compatibility: Desktop App Version 9.5 and above.

This command is used to fetch User Groups where specified user is a member and can be used within scan loop at master or sub report levels.

\Fetch_Groups_Where_User_Is_Member(<<User Group Name>>)

Parameter

<User Group Name>>	Mandatory	It will fetch group members for specified user group name.

Fields Available

All fields from User Editor are available in this command.

Examples

\Fetch_User_Group_Members('Database Team')\
\Fetch_User_Group_Members(a: GroupName)\

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\
\ PROJECT_NAME \

User Group members

\Fetch_User_Group_Members('DBTeam')\

Name	License Type	Inactive

\scan(a) \

\a: UserName\	\ a : License Type \	\ a : Inactive \

\endscan\

Sample Template 2

User Group members

\Fetch_User_Groups()\
\scan(a)\

**Group :: \ a : Name **

\Fetch_User_Group_Members(a:Name)\

Name	License Type	Inactive

\scan(b) \

\b: UserName\	\ b : License Type \	\ b : Inactive \

\endscan\

\endscan\

\Declare_Variable_User('variable_User1','False', 'True','Select Users from All Projects')\

\Prompt_For_Variable_Values('ALL')\

Groups for User

\ Fetch_Groups_Where_User_Is_Member(variable_User1)\

Name	Member Count	Inactive

\scan(a) \

\a: Name\	\ a : Users \	\ a : Inactive \

\endscan\

Sample Template 3

Groups for User

\ Fetch_Users()\

\scan(a) \

**User : \ a : UserName **

\ Fetch_Groups_Where_User_Is_Member(a : UserName)\

Name	Member Count	Inactive

\scan(b) \

\b: Name\	\ b : Users \	\ b : Inactive \

\endscan\
\endscan\

Fetch_Groups_Where_User_Is_Member

Compatibility: Desktop App Version 9.5 and above.

This command is used to fetch User Groups where specified user is a member and can be used within scan loop at master or sub report levels.

\Fetch_Groups_Where_User_Is_Member()

Parameter

<<User Name>>	Mandatory	It will fetch User Groups where specified user is a member.

Fields Available

All fields from User Group Editor are available in this command.

Examples

\Fetch_Groups_Where_User_Is_Member()\

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\
\ PROJECT_NAME \

\Declare_Variable_User('variable_User1','False', 'True','Select Users from All Projects')\
\Prompt_For_Variable_Values('ALL')\

Groups for User

\ Fetch_Groups_Where_User_Is_Member(variable_User1)\

Name	Member Count	Inactive

\scan(a) \

\a: Name\	\ a : Users \	\ a : Inactive \

\endscan\

Sample Template 2

Groups for User

\ Fetch_Users()\
\scan(a) \

**User : \ a : UserName **

\ Fetch_Groups_Where_User_Is_Member(a : UserName)\

Name	Member Count	Inactive

\scan(b) \

\b: Name\	\ b : Users \	\ b : Inactive \

\endscan\
\endscan\

Glossary (Terms and Acronyms)

Fetch_Glossary

NOTE: This command "Fetch_Glossary" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 4.20 and above.

This primary command fetches Glossary Terms and Acronyms for the current Project sorted in alphabetical order.

\Fetch_Glossary()

Parameters

There are no parameters for this command.

Example

\Fetch_Glossary()\

Fetch_Glossary_By_Condition

NOTE: This command "Fetch_Glossary_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 6.50 and above.

This primary command generates Terms based on filter conditions.

\Fetch_Glossary_By_Condition('<<Filter Condition>>','<<Sort by Field>>')

Parameters

<<Filter Condition>>	Optional	If not specified, all Records will be fetched for the selected Project.
<<Sort by Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is not sorted.

Examples

\ Fetch_Glossary_By_Condition() \
\ Fetch_Glossary_By_Condition(' "Category" = "Car" ') \
\ Fetch_Glossary_By_Condition(' "Crt by" = "<< Me >>" and "Category" = "IMP" ' ,'Term') \
\ Fetch_Glossary_By_Condition('' , 'Term') \
\ Fetch_Glossary_By_Condition('' , 'Category,Term') \
\ Fetch_Glossary_By_Condition('{Having incoming traces ("REQ")}') \

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Glossary_By_Condition(' "Category" = "Business" ', 'Term')\
\scan(a)\

- **\a:Term\ - \a: Aliases**

\endscan\

Fetch_Glossary_With_Aliases_By_Condition

NOTE: This command "Fetch_Glossary_With_Aliases_By_Condition" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 6.50 and above.

This primary command generates Terms as well their Aliases based on a filter conditions.

\Fetch_Glossary_With_Aliases_By_Condition('<<Filter Condition>>', '<<Sort by Field>>')

Parameters

<<Filter Condition>>	Optional	If not specified, all Records will be fetched for the selected Project.
<<Sort by Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is not sorted.

Examples

\ Fetch_Glossary_With_Aliases_By_Condition()\
\ Fetch_Glossary_With_Aliases_By_Condition(' "Category" = "Car" ')\
\ Fetch_Glossary_With_Aliases_By_Condition(' "Crt by" = "<< Me >>" and "Name" = "IMP" ' ,'Term')\
\ Fetch_Glossary_With_Aliases_By_Condition('' , 'Term')\
\ Fetch_Glossary_With_Aliases_By_Condition('' , 'Category,Term')\
\ Fetch_Glossary_With_Aliases_By_Condition('{Having incoming traces ("REQ")}')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Glossary_With_Aliases_By_Condition()\
\scan(a)\
\a:Term\ - \a: Aliases
\endscan\

Fields Available

Field	Description
ID	
Term	Term name
Description	Rich text formatted Term Description
Category	Term Category
Aliases	Aliases for the Term
Version	
Project	
Crt by	
Crt dt	
Upd by	
Upd dt	

Conditional Output using Sections

You can use Sections commands to divide your document template into various Sections and at runtime, you can prompt the user to select which Sections should be outputted into the document.

Create_Sections

Compatibility: Desktop App Version 4.20 and above.

This command specifies Section names in a document template. You can specify as many Sections as required.

\Create_Sections('<<Section1>>', '<<Section2>>',.... , '<<SectionN>>')

Enter a Section name for each Section that you need to define within the document. These rules must be followed for naming Sections:

- Section name can only have letters, numbers and the underscore ('_') symbol.
- There must be no spaces in the name.
- The name of the Section must be unique in that document.

Example

\Create_Sections('Vision', 'Actors', 'Use_Cases', 'Use Case Diagram', 'Glossary')\

Pre-selecting Sections

This is the technique for setting the default value of a Section to **ON** or **OFF**. If you want certain document Sections to be turned **ON** by default, you can do so by pre-selecting the Sections by setting their values to **TRUE**.

\ <<Section_Name>> = 'True'

Example

\Use_Cases = 'True'\

Confirm_Sections

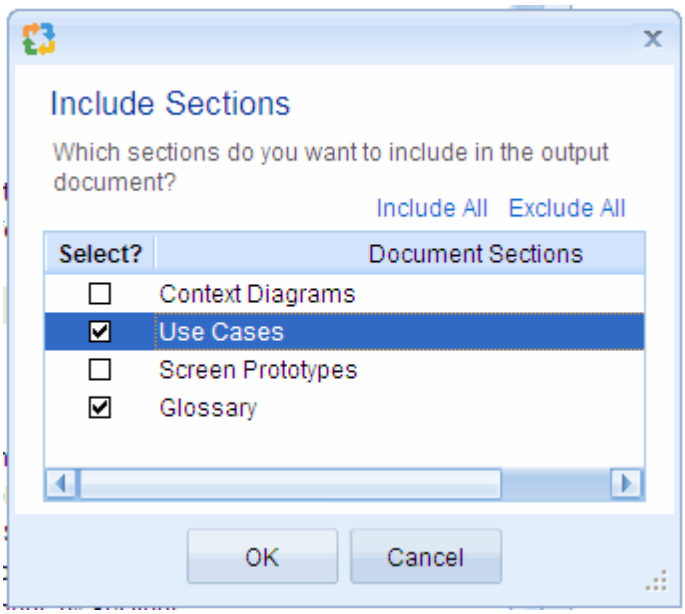
Compatibility: Desktop App Version 4.20 and above.

This command prompts the user into selecting Sections to generate in the document.

\Confirm_Sections('<<Section1>>', '<<Section2>>',.... , '<<SectionN>>')

Example

\Confirm_Sections('Contect_Diagrams', 'Use_Cases', 'Screen_Prototypes', 'Glossary')\



Dividing Document into Sections

This is the technique of dividing a document into Sections. Enclose a part of the document that you want to put into a Section, inside an IF-ENDIF command pair. You can use the IF command to test if the Section has been turned **ON**, when a document is generated.

**\if(<<Section_Name>>) **

The Fetch commands, Fields, etc. for this Section go here:

\endif

Examples

\Create_Sections('Use_Cases', 'Glossary')\
\Confirm_Sections('Use_Cases', 'Glossary')\

\if(Use_Cases) \

\Fetch_Use_Cases()\

Use Cases

\Scan(a) \

\a : Name\ [\ a : Id \ ]

\ a : Description Goal in context \

\endscan\

\endif\

\if(Glossary) \

\Fetch_Glossary()\

Glossary

\ scan(a) \

\a : Term\ [\ a : Id \ ]

\InsertRTF(a : Description)\
\endscan\
\endif\

Include_section

Compatibility: Desktop App Version 3.35 and above.

This command interactively prompts the user for every Section. The Include_section command allows you to interactively prompt the user to include or exclude a Section of the document at runtime.

This command is complementary to the Create_Sections/Confirm_Sections command. The primary difference is, with the Include_section command, you do not need to define the Sections in advance. However with Include_section, the user will be prompted multiple times during document generation.

\if(Include_section('<<A name for the optional section>>'))

The Fetch commands, Fields, etc. for this Section go here.

\endif

When generating a document, this command will prompt the user with the following message:

Do you want to include the following Section in the generated output?

"<<A name for the optional Section>>"

Examples

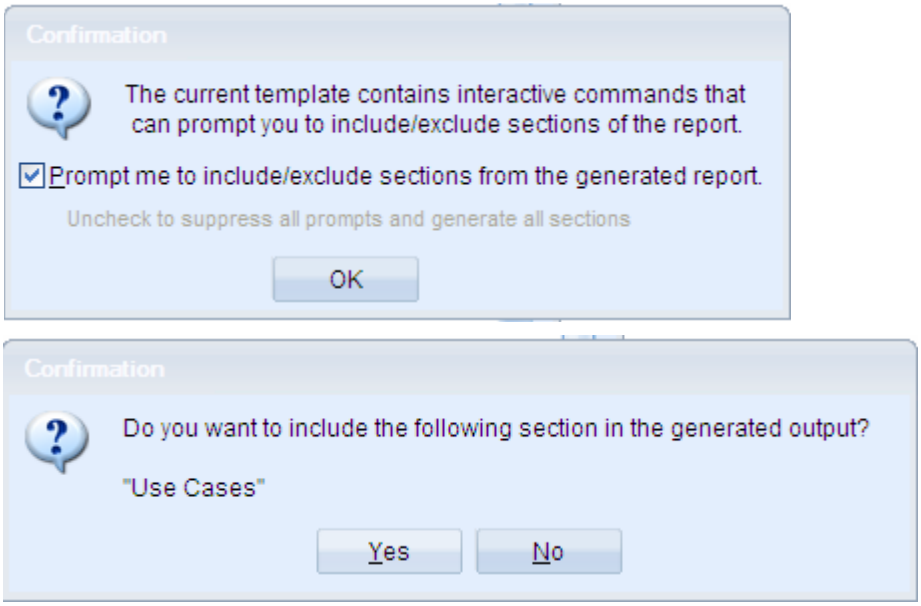
\if(Include_section('Use Cases'))\
\Fetch_Use_Cases()\

Use Cases

\scan(a) \

\ a : Name \ [\ a : Id \]
Description (Goal in context)
\ a : Description Goal in context \

\EndScan\
\endif\



NOTE:
The Confirm_Sections command is a better alternative to this command.
The Confirm_Sections is a newer version of the Include_sections command.
The primary difference is, with Include_sections you will be prompted multiple times during document generation.

Commonly used commands

InsertRTF

Compatibility: Desktop App Version 4.20 and above.

This command inserts the contents of a Rich Text field into the document. For example, the Description field for Actors is a Rich Text field.

\ InsertRTF(<<Dataset Identifier>> : <<Rich Text Field Caption>>)

Compatibility for additional Font Name and Font Size parameters: Desktop App Version 8.01 and above.

\ InsertRTF(<<Field>>, <<Font Name>>, <<Font Size>>)

Parameters

<<Font Name>>	Optional	Specify the Name of the Font as second parameter
<<Font Size>>	Optional	Specify the Size of the Font as third parameter

Fields Available

No fields are available for this command.

Examples

\ InsertRTF(a : Description)\
\ InsertRTF(b : Pre Condition)\

Example with Font Name and Font Size parameters

Insert_Rich_Text_Using_Word(a:Description, 'Verdana' , 9)

Insert_Rich_Text_Using_Word

Compatibility: Desktop App Version 6.50 and above.

This miscellaneous command is used to generate table in the specified Rich Text field. It inserts the contents of a Rich Text field into the document. For example, the Description field for Actors is a Rich Text field. The advantage of using this command over the InsertRTF command is that if you have a Rich Text table in the field, that table can be generated without any distortions. This command can be used anywhere within the template.

There is an issue with the InsertRTF command when it is used in the DocProcessor template inside a table cell and the Rich Text field has a table. This command may work slower as compared to InsertRTF when it is used extensively in a single template and there are multiple tables in the Rich Text fields. Even if Microsoft Word is not installed on the computer, this command can be used, but it may change some of the Styles and Fonts of some of the fields.

\ Insert_rich_Text_Using_Word(<<Rich Text Field Caption>>)

Parameter

<<Rich text Field>>	Mandatory	Specify Rich Text as the first parameter

Compatibility for additional Font Name and Font Size parameters: Desktop App Version 8.01 and above.

<<Font Name>>	Optional	Specify the Name of the Font as second parameter
<<Font Size>>	Optional	Specify the Size of the Font as third parameter

Fields Available

No fields are available for this command.

Examples

\ Insert_Rich_Text_Using_Word(a:Description) \
\ Insert_Rich_Text_Using_Word(a:Description, 'Verdana', 20) \

Examples

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

\Fetch_Requirements_By_Condition('' ,'Title')\

\scan(a)\
[\ a : Id \] - \a : Title

\ Insert_Rich_Text_Using_Word (a : Description) \

\endscan\

FRTF_Using_Word()

Compatibility: Desktop App Version 7.09 and above.

This miscellaneous command generates a table-in–a-table, in the specified Rich Text field. It can be used anywhere within the template.

FRTF_Using_Word(<<Rich text Field>>)

Parameter

<<Rich text Field>>	Mandatory	Specify Rich Text as the first parameter.

Fields Available

No fields are available for this command.

Examples

FRTF_Using_Word(a:Value1)

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Project Baseline Comparison

\Fetch_Compare_Baselines('$PROMPT_BASELINE$', '$PROMPT_BASELINE$')\
\scan(a)\\if (Bof(a))\

Baseline 1: **\a:Baseline1**

Type:\a:Baseline1Type\
Created:\a:Baseline1CrtDt\

Baseline 2: **\a:Baseline2**

Type:\a:Baseline2Type \
Created:\a:Baseline2CrtDt\

\endif\\endscan\

Legend	
Added	Record Added in new Baseline
Modified	Record Modified in new Baseline
~~Deleted~~	Record Deleted in new Baseline

\scan(a)\\if(a : Is Header = True)\

\ a:Name\

\elsif(a : Action code = 'Added in baseline1')\

\a:Name\

\a:Change\ Updated By \a: Upd By1\ On \a: Upd Dt1\ Version \a: Version1\
\elsif(a: Action code = 'Deleted from baseline1')\

~~\a:Name\~~

\a:Change\ **Created By** \a: Upd By2\ **On** \a: Upd Dt2\ **Version** \a: Version2\
\elsif (a : Action code = 'Added in baseline2')\

\a:Name\

\a:Change\ Updated By \a: Upd By2\ On \a: Upd Dt2\ Version \a: Version2\
\elsif (a : Action code = 'Deleted from baseline2')\

~~\a:Name\~~

\a:Change\ **Created By** \a: Upd By1\ **On** \a: Upd Dt1\ **Version** \a: Version1\
\elsif(a : Action code = 'Modified')\

\a:Name\

\a:Change\ **Updated By** \a: Upd By2\ **On** \a: Upd Dt2\ **Version** \a: Version2\
\else\\endif\\if(a : Action = 'Modified')\\Fetch_Compare_Versions(a: Id, a: VersionId1, a: VersionId2)\\scan(b)\\if(Bof(b))\

Field	**\a: Baseline1**	**\a: Baseline2**

\endif\
\if(b : Type = 'RTF')\

\b:Field\	\ FRTF_Using_Word(b: Value1)\	\ FRTF_Using_Word(b: Value2)\

\elsif(b : Type = 'Diagram')\

\b:Field\	\Insert_Field_As_Diagram_In_CMs (b:Value1, '9', '', 'EMF')\	\Insert_Field_As_Diagram_In_CMs (b:Value2, '9', '', 'EMF')\

\elsif(b : Type = 'Merged')\

\b:Field\	\ Insert_Rich_Text_Using_Word(b: Merged)\

\elsif(b : Type = 'Text')\

\b:Field\	\Insert_Differences (b: Value1, b: Value2)\

\else\

\b:Field\	\b: Value1\	\b: Value2\

\endif\\endscan\\endif\\endscan\

InsertRtfAsText

Compatibility: Desktop App Version 4.20 and above.

This is a miscellaneous command that can be used independently at any place in the template. This command is used to convert Rich Text into simple text.

\InsertRtfAsText(<<Name of the Field containing RTF Text>>)

Parameter

<<Name of the Field containing RTF Text>>	Mandatory	Specify the field names containing Rich Text that you wish to convert into simple text.

Examples

\InsertRtfAsText(a : Description)\
\InsertRtfAsText(a : Pre conditions)\
\InsertRtfAsText(a : Post conditions)\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

Id: \ a: Id\ \a:Name\
Pre conditions

\InsertRtfAsText(a : Pre conditions)\

Post conditions

\InsertRtfAsText(a : Post conditions)\
\endscan\

Insert_Indented_Rtf

Compatibility: Desktop App Version 4.20 and above.

This command displays the Rich Text indented by a specified level.

**\ Insert_Indented_Rtf (<<Field Tag of RTF Field>>, '<<Indentation Level>>') **

Parameters

<< Field Tag of RTF Field >>	Mandatory	Specify the Field Names containing Rich Text.
<< Indentation Level >>	Mandatory	Specify the levels by which you want to indent the Rich Text.

Examples

\ Insert_Indented_Rtf(a : Description, '1')\
\ Insert_Indented_Rtf(a : Pre conditions, '2')\
\ Insert_Indented_Rtf(a : Post conditions, '3')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Requirements_Tree_By_Document_Id_By_Condition('$DEFAULT_REQUIREMENTS_DOCUMENT$')\
\scan(a) \
\if (! eof(a))\
\if (a : Indentation Level = '1')\

\a: wbs \ [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '1')\

\elsif (a : Indentation Level = '2')\

\a: wbs \ [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '2')\

\elsif (a : Indentation Level = '3')\

\a: wbs \ [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '3')\

\else\

\a: wbs \ [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '4')\

\endif\

\endif\

\endscan\

Is_Field_Non_Empty

Compatibility: Desktop App Version 3.35 and above.

This command checks whether or not a field is assigned a value. You can use this command in conjunction with an IF command for conditional processing. It returns **TRUE** if the field has a value and **FALSE** if the field is empty.

**\ Is_Field_Non_Empty(<<Dataset Identifier>> : <<Field caption>>) **

Examples

\if (Is_Field_Non_Empty (a : Description))\

Description

\ InsertRtf (a : Description) \

\endif\

Is_Field_Empty

Compatibility: Desktop App Version 3.35 and above.

This command checks whether or not a field is empty. You can use this command in conjunction with an IF command to perform conditional processing. This command is **TRUE** if the field is empty and **FALSE** if the field is assigned a value.

**\ Is_Field_Empty(<<Dataset Identifier>> : <<Field caption>>) **

Examples

\if (Is_Field_Empty (a : Description))\

This indicates that the Description field is empty.

\endif\

Is_Value_Selected

Compatibility: Desktop App Version 4.20 and above.

This command checks whether or not a Checklist field contains a specified value.
If the value is checked/selected in the Checklist field, then the result is **TRUE**. And if the specified value is not then the result is **FALSE**.

This command is a miscellaneous command. It can be used within any primary Fetch command. It is applicable to Checklist fields only and returns Boolean (**TRUE** or **FALSE**) values.

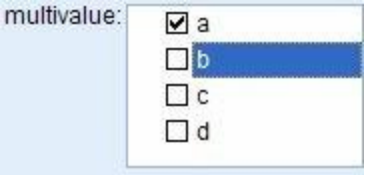
**\ Is_Value_Selected(<<CheckList Field>>, '<<One Value from the List>>') **

Parameters

<<Checklist Field>>	Mandatory	Specify the Checklist field as the first parameter. If a field other than the Checklist field is specified, then this command will return an error message.
<<One Value from the List>>	Mandatory	Enter only one value from the Checklist field.

Examples

Refer to the following screenshot:



The following example will return the result as **TRUE**:

Is_Value_Selected(a:multivalue , 'a')

The following example will return the result as **FALSE**:

Is_Value_Selected(a:multivalue , 'b')

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

\a : Name\ [\ a : Id \]

\if(Is_Value_Selected(a:multi value , 'A'))\

Here means A is selected.

\endIf\

\if(!(Is_Value_Selected(a:multi value , 'B')))\

Here means B is not selected.

\endIf\
\endscan\

Insert_URL_For_Record

Compatibility: Desktop App Version 4.20 and above.

This command inserts a live URL to a Record in the document output. When the user opens the Link from the generated document (using Ctrl+Click or any other means), Windows will launch the Record in TopTeam or Visual Use Case Desktop App (Windows Client).

\Insert_URL_For_Record(<<ID Field>>, <<Record Type Field>>, <<Any field to represent the text of the URL>> or <<Static Text>>)

Examples

\Insert_URL_For_Record(a : ID, a : Type, a : Name)\

This command inserts a Link to the Record in the output document. Assuming the Name field contains the string Reserve a Video, the output will look like the following, and the user of the document will be able to Ctrl+Click on the Link to open the Record in Windows Client.

Reserve a Video

\Insert_URL_For_Record(a : ID, a : Type, 'Ctrl + Click here to open this record')\

This command inserts a Link to the Record in the output document. The following output will display and the user of the document will be able to Ctrl+Click on the Link to open the Record in Windows Client.

Ctrl + Click here to open this record

Insert_URL_for_Record_ID

Compatibility: Desktop App Version 4.50 and above.

These commands insert a URL for a Record with the ID specified as the first parameter. They are used to insert URLs for TopTeam Desktop App as well as Web TopTeam Records. This is a miscellaneous command that can be used independently at any place in template.

\Insert_Win_URL_For_ID('<<ID>>' , '<< URL Text >>')
\Insert_Web_URL_For_ID('<<ID>>' , '<< URL Text >>')

Repository Objects URL Commands

\Insert_Win_URL_For_Object_ID('<<ID>>' , '<< URL Text>>')
\Insert_Web_URL_For_Object_ID('<<ID>>' , '<< URL Text>>')

Tracking Items URL Commands

\Insert_Win_URL_For_Item_ID('<<ID>>' , '<< URL Text>>')

\Insert_Web_URL_For_Item_ID('<<ID>>' , '<< URL Text>>')

Parameters

<<ID>>	Mandatory	The Record Identifier can have the ID_With or Without_Prefix. This parameter is used in both Repository Objects URL Commands and Tracking Items URL Commands. In this command, applying the Prefix to the ID is not necessary. Even if you apply a Tracking Items Prefix in the case of a Repository Objects command, this command will fetch the Repository Objects Records of the specified ID. The value contained by this parameter can be a field or a hard coded string. If a Record with specified ID does not exist, this command will generate an error. If the ID without a Prefix is specified, it will be considered as the Repository Objects Record ID and the Link for the Repository Objects Record will be created in the template. To create a URL for Tracking Items Records, an ID Prefix MUST be supplied. For example, in the identifier UCD-2342, UCD is the ID Prefix and represents Use Case Diagrams and 2342 is the record identifier.
<< URL Text >>	Optional	Specify any Field Name containing the URL text. If not specified, the text of URL will be similar to UC-1212 as in the example below.

Fields Available

There are no fields available for this command.

Examples

\Insert_WIN_URL_For_ID('UC-1212')\- The text of URL will be like UC-1212.
\Insert_WIN_URL_For_ID('1212')\-Create URL for Repository Object having ID '1212'
\Insert_WIN_URL_For_ID('CR-1212')\
\Insert_WIN_URL_For_ID('CR-1212', 'Link for My Change Request')\
\Insert_Win_URL_For_Object_ID('111')\
\Insert_Win_URL_For_Item_ID('222' , 'Link for My Change Request')\

Use within a scan loop

\Insert_WIN_URL_For_ID (a : ID)\
\Insert_WIN_URL_For_ID (a : ID, a : Title)\
\Insert_Win_URL_For_Object_ID(a : ID)\
\Insert_Win_URL_For_Item_ID(a : ID, a : Title)\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Use_Cases('' ,'Name')\
\scan(a)\

\a:Name\ [\ a: Id\]

\Insert_WIN_URL_For_ID (a : ID)\
\endscan\

Insert_URL(Caption, Address)

Compatibility: Desktop App Version 12.5 and above.

This command generates a link in the output document with the specified caption name and address. This command can be used independently or within a Scan Loop.

\Insert_URL('<<URL Name>>', '<<URL Address>>')

Parameters

<<URL Name>>	Mandatory	This parameter can be a Field or String that inserts a URL link for the specified Name.
<<URL Address >>	Optional	This parameter can be a Field or String that inserts a URL link for the specified URL address.

Fields Available

There are no fields available for this command.

Examples

Use Independently

\Insert_URL('TechnoSolutions','http://www.technosolutions.com')\

Use within a Scan Loop

\Insert_URL (a : URL Name, a : URL Address)\

Sample Templates

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

```
//Prerequisite – Field URL Name and Field URL Address with valid data
\Fetch_Repository_objects_by_Condition('Name')\
\scan(a) \
\ a: Id\-\ a : Name \
\Insert_URL(a : URL Name, a : URL Address)\
\endscan\
```

Insert_Permalink

Compatibility: Desktop App Version 8 and above.

This command inserts a URL Link for a specified Record ID. This UDF will give first preference to the Web URL. If the Web URL is not configured, it will create a WIN URL without raising an exception. This command can be used independently or within a Scan Loop.

\Insert_Permalink('<<ID>>', '<<URL Text>>', '<<Is_WIN_URL>>')
\Insert_Permalink('<<ID>>', '<<URL Text>>', '<<Is_WIN_URL>>',
'<<Is_LinkToSpecificVersion >>')

Parameters

<<ID>>	Mandatory	The record identifier can be ID_With or Without_Prefix. This parameter is used for both Repository Objects and Tracking Items. This parameter can be a Field or a String. If the specified record ID is not present, this command will generate an error.
<< URL Text >>	Optional	Specify any Field Name containing the URL Text or Text you want to show as the URL Text. If not specified, the Text of URL will be like UC-1212.
<< Is_WIN_URL >>	Optional	Allow to create a WIN URL even if WEB URL is configured. By default, it is **FALSE** so: If WEB URL is configured, it will create WEB URL. If WEB URL is not configured, it will create WIN URL. Set **TRUE** to create WIN URL , even if WEB URL is configured in settings.
<< Is Link to Specific Version >>	Optional	Default is **FALSE**. If **FALSE**, URL will always open the Current/Latest version of the record. If **TRUE**, URL will always open old version of the record when this URL is created.

Fields Available

There are no fields available for this command.

Examples

Use Independently

\Insert_Permalink('UC-1212')\- The text of URL will be like UC-1212.

\Insert_Permalink('CR-1212')\

\Insert_Permalink('1212')\

Use within a Scan Loop

\Insert_Permalink (a : ID)\

\Insert_Permalink (a : ID, a : Title)\

\Insert_Permalink(a : ID, a : Title, True)\

Sample Templates

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

Independent Objects ID with Prefix

Insert_Permalink ('UC-762')

\Insert_Permalink ('UC-762')\

Independent Items ID with Prefix

Insert_Permalink ('ISS-123')

\Insert_Permalink ('ISS-123')\

Independent Objects ID without Prefix

Insert_Permalink ('762')

\Insert_PermalinkD ('762')\

Independent Items ID without Prefix

Insert_Permalink ('123')

\Insert_Permalink ('123')\

Independent Objects ID with Prefix WIN URL even if WEB is configured

\Insert_Permalink('CR-1212', 'Link for My Change Request', True)\

Independent Objects ID with Prefix will open latest Version

\Insert_Permalink('CR-1212', '')\

Independent Objects ID with Prefix will open old Version

\Insert_Permalink('CR-1212', '', '', True)\

Objects ID within a Scan Loop

Insert_Permalink (a : ID)
\Fetch_Use_Cases_By_Condition('' ,'Name')\
\scan(a)\
\a:id\ \a:Name\
\Insert_Permalink (a : ID)\
\endscan\

Items ID within a Scan Loop

Insert_Permalink (a : ID)
\Fetch_Tracking_Items_By_Condition('', '', 'Id')\
\scan(a)\
\a:id\ \a:Title\
\Insert_Permalink (a : ID)\
\endscan\

Insert_Bookmark_For_Record

Compatibility: Desktop App Version 8.2 and above.

This command inserts a Bookmark for a specified record ID. This UDF will point to the bookmark of a record ID/link in the output document. If there is no bookmark specified for a record

ID/link, then the record will open in TopTeam Web or TopTeam Desktop App as per the configuration. This command can be used independently or within a Scan Loop.

\Insert_Bookmark_For_Record('<<ID>>')

Parameters

<<ID>>	Mandatory	The record identifier can be ID_With or Without_Prefix. This parameter is used for both Repository Objects and Tracking Items. This parameter can be a Field or a String. If the specified record ID is not present, this command will generate an error.

Fields Available

There are no fields available for this command.

Examples

Use Independently

\Insert_Bookmark_For_Record('UC-1212')\

Use within a Scan Loop

\Insert_Bookmark_For_Record (a : ID)\

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\
\ PROJECT_NAME \

Insert_Permalink (a : ID)
\Fetch_Use_Cases_By_Condition('' ,'Name')\
\scan(a)\
\Insert_Bookmark_For_Record(a : ID)\\a:Name\
\Insert_Permalink (a : ID)\
\endscan\

Sample Template 2

\ PROJECT_NAME \

Use Cases

\scan(a)\\if (! EOF (a))\
\a:Name\
\Insert_Permalink(a: Id)\

\InsertFlows(a : Flows)\
\Fetch_Linked_Requirements(' Type, Id')\
\if(! eof(b))\
Linked Requirements

Id	Title
\scan(b)\ \if (! eof(b))\	
\Insert_Bookmark_For_Record(b : ID)\\Insert_Permalink(b: Id)\	\ b : Title \\if (Is_Field_Non_Empty (b : Description))\ \ Insert_Rich_Text_Using_Word (b : Description,'Segoe UI', '10')\ \endif\
\endif\\endscan\\endif\ \endif\\endscan\	

Insert_Formatted_Title

Compatibility: Desktop App Version 12.5 and above.

This command inserts formatted record title in the output document as per the format specified in *Global Settings*. This is a miscellaneous command and can be used inside Scan-Endscan.

\Insert_Formatted_Title('<<Title Field Name>>')

Parameters

<<Title Field Name>>	Mandatory	Specify the *Title* field to format the title value as per the *Global Settings* format. If the parameter is not specified, this command will generate an error.

Fields Available

There are no fields available for this command.

Examples

\ Insert_Formatted_Title(a : Title)\
\ Insert_Formatted_Title(a : Name)\
\ Insert_Formatted_Title(a : ID)\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \
[\ a: Id\]-\ a : Name \
\Insert_Formatted_Title(a : Name)\
\endscan\

Insert_Record_Location

Compatibility: Desktop App Version 12.5 and above.

This command inserts record path or location in the output document to display the complete path of the record such as *Project path/Folder/OneView Section/Requirements Document*, etc.This is a miscellaneous command and can be used inside Scan-Endscan.

\Insert_Record _Location('<<ID Field Name>>')\

Parameters

<<ID Field Name>>	Mandatory	Specify the *ID* field to insert record location. If the parameter is not specified, this command will generate an error.

Fields Available

There are no fields available for this command.

Examples

\Insert_Record_Location(a : ID)\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \
[\ a: Id\]-\ a : Name \
\Insert_Record_Location(a : ID)\
\endscan\

Insert_User_Image

Compatibility: Desktop App Version 8.1 and above

This is a miscellaneous command that is it can be used independently in a template. This command is used to output user images (configured in *Administration>Manage User Accounts*) in the desired width and height, and in the required image format type.

Insert_User_Image('<<User Name Or Image Path>>','<<Width>>','<<Height>>','<<Image Size Unit>>','<<Is_Force_Size>>')

Parameter

<< User Name Or Image	Optional	If the user name/image path is blank, by default, logged in user's image will display.

Path >>		
<<Width>>	Optional	Set the desired width of the image to display in the output.
<<Height>>	Optional	Set the desired height of the image to display in the output.
<<Image Size Unit>>	Optional	This parameter contains one of the following values: • **CMs** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in cms. • **Inches** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in inches. • **Pixels** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in pixels By default, the value of this parameter is **Pixels**.
<<Is Force Size>>	Optional	By default, this parameter is FALSE. The image will be resized only if the actual image size is greater than the specified size. If the actual size of the image is small and the required specified size is greater, set this value to TRUE to enforce sizing.

Fields Available:

There are no fields available for this command.

Examples:

\Insert_User_Image()\

Sample Template:

\Insert_User_Image()\
\Insert_User_Image(‘’,2,1, ‘cms’, ‘EMF’)\
\Insert_User_Image(‘Steve Project Manager’,2,1, ‘cms’, ‘EMF’)\
\Insert_User_Image(‘D:\Images\Steve.JPG’,2,1, ‘cms’, ‘EMF’)\

Insert_Multi_Value_Field

Compatibility: Desktop App Version 13.32 and above.

This miscellaneous command can be used to replace the default “Enter” separator with the specified delimiter text. This command can be used inside Scan-Endscan.

\Insert_Multi_Value_Field(‘<<MultiValue Field Name>>’,'<<Delimiter_Text>>')

Parameters

<<MultiValue Field Name>>	Mandatory	Specify MultiValue field names.
<<Delimiter_Text>>	Optional	Specify the Delimiter_Text to replace default the "Enter" separator with the specified delimiter text.

Fields Available

There are no fields available for this command.

Examples

\ Insert_Multi_Value_Field(a : multivalue, ', ')\
\ Insert_Multi_Value_Field(a : multivalue, '; ')\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \

**[\ a: Id\]-\ a : Name **
\ a : multivalue\
\ Insert_Multi_Value_Field(a : multivalue, ', ')\
\endscan\

Comments

Compatibility: Desktop App Version 3.35 and above.

This command adds comments to the document template. The text written inside this command is not generated into the document. You can also use the short form of this command, which is simply the letter C.

\Comments('<<Comment Text>>')
\C('<<Comment Text>> ')

Examples

\Comments ('This section was added here by John Doe on 12/10/2005.')\
\C('This section was added here by John Doe on 12/10/2005.')\

Execute_Template_For_Id

Compatibility: Desktop App Version 6.20 and above.

This miscellaneous command can be used independently at any place in the template. It is used to insert another template into the Report.

\Execute_Template_For_Id('<<ID Prefix>>', '<<Version_Num>>', '<<Template Name>>')

Parameters

<< ID Prefix>>	Mandatory	This is the record identifier.

<<Version_Num>>	Optional	Specify the Version Number if you want to fetch a specific version of a Record. e.g. '1.3'.
<<Template Name>>	Optional	Specify the Template Name. This template must be available in the Document Generate Dialog for current Record Type Editor/List/Tree. If template is not available for current Record Type, then the Base Repository Objects or Tracking Items template will be used. If not specified, the command will default to the RTF Preview template for the current Record Type.

Fields Available

There are no fields available for this command.

Examples

\Execute_Template_For_Id(UC-2838)\
\Execute_Template_For_Id(UC-2838, '1.08')\
\Execute_Template_For_Id(b:ID, b:Version)\
\Execute_Template_For_Id(b:ID, b:Version, 'Insert Report Detail')\

Examples

\Declare_Variable('Record_ID', 'Text', 'Enter Record Display Id', '')\
\Declare_Variable('Record_Version', 'Float', 'Enter record version', '')\
\Prompt_For_Variable_Values('Record_ID', 'Record_Version')\
\Execute_Template_For_Id(Record_ID, Record_Version, '')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Repository_Objects_By_Condition('RPG')\
\scan(a) \

Package: \a : Name\ [\ a : ID\]

\Fetch_Package_Contents(a : ID)\

\scan(b) \

Details for: \ b : Name\

\Execute_Template_For_Id(b:ID, b:Version, 'Insert Report Detail')\
\endscan\
\endscan\

Replace_Text

Compatibility: Desktop App Version 13.32 and above.

This miscellaneous command can be used to find and replace text from the specified field/string. This command can be used inside Scan-Endscan.

\Replace_Text('<<Field Name/Source Text>>','<<Find_Text>>','<<Replace_Text>>')

Parameters

<<Field Name/Source Text>>	Mandatory	Specify the field names/Source text to find and replace text.
<<Find_Text>>	Optional	Specify Find_Text to be replaced from value of the Field or Source Text.
<<Replace_Text>>	Optional	Specify Replace_Text to search and replace Find_Text from Source Text.

Fields Available

There are no fields available for this command.

Examples

\Replace_Text(a : Description, 'Sarch', 'Search')\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \

**[\ a: Id\]-\ a : Name **

\ Replace_Text(a : Description, 'Single', 'Multi')\
\endscan\

Record Type Commands

Fetch_Record_Type_Details

Compatibility:
<<Display Prefix of Record Type>> - Mandatory - Desktop App Version 3.35 and above.
<<Display Prefix of Record Type>> - Optional - Desktop App Version 6.40 and above.
<<Name of Record Type>> - Optional – Desktop App Version 8.2 and above.

This is a Master level command used to display details of the Record Type for which the ID Prefix is specified.

\Fetch_Record_Type_Details('<<Record Type>>')

Parameter

<<Record Type>>	Optional	"Record Identifier" of the record. User can specify, Record type Singular Name OR Record type Plural Name OR Record type Display Prefix If Record Type is specified - It will display the details of the SINGLE record type. If Record Type is NOT specified - It will display the details of the ALL records.

Fields Available

Field	Description
Created on	
Modified on	
Name (Plural)	
Name (Singular)	
Record Type Unique Name	
Versioned	
WBS Start Number	
Description	
Prefix for ID	
Default Display Order	
Foldered	
Image	
Initial State	

Examples

\Fetch_Record_Type_Details('DIA')\
\Fetch_Record_Type_Details('SCR')\

\FETCH_RECORD_TYPE_DETAILS('UC')\
\FETCH_RECORD_TYPE_DETAILS('UCS')\
\FETCH_RECORD_TYPE_DETAILS('Use Case')\
\FETCH_RECORD_TYPE_DETAILS('Use Cases')\

Examples

\Fetch_Record_Type_Details('UC')\
\scan(a) \

Name : \a: Name Singular \

ID : \a: Prefix For ID\

Description:

\InsertRtf (a : Description)\

\endscan\

Examples

\Fetch_Record_Type_Details()\

\scan(a) \

Name : \a: Name Singular \

ID : \a: Prefix For ID\

Description:

\InsertRtf (a : Description)\

\endscan\

Fetch_Record_Type_Associations

Compatibility: Desktop App Version 3.35 and above.

This command is a Master level command used to fetch Record Types Links, according to the Record Type ID Prefix specified.

\Fetch_Record_Type_Associations('<<ID Prefix>>')

Parameter

<<ID Prefix>>	Mandatory	This is the record identifier.

Fields Available

Field	Description
Record Type	
Incoming Record Type	

Forward Record Type	
Association Type	Link Type
Disp Seq	
Inactive	

Examples

\Fetch_Record_Type_Associations('BREQ')\
\Fetch_Record_Type_Associations('UC')\

Examples

\Fetch_Record_Type_Associations('UC')\
\scan(a) \
\if (bof(a))\

Record Type: \a : Record Type \

\endif\

Incoming Record Type: \a: Incoming Record Type\
Link Type: \a: Association Type\
Forward Record Type: \a: Forward Record Type\

\endscan\

Fetch_Record_Type_Project_Inclusions

Compatibility: Desktop App Version 3.35 and above.
This is a Master level command used to display Project names according to the specified ID Prefix which is included. This command list generates the names of the Projects in which it is PIT(Project Included Type).

\Fetch_Record_Type_Project_Inclusions('<<ID Prefix>>')

Parameter

<<ID Prefix>>	Mandatory	This is the record identifier.

Fields Available

Field	Description
Record Type	
Project Name	
Display Seq.	
Security Enforced	
Hidden	
Versioned	
WBS Start No.	
Crt by	
Upd by	
Upd dt	
Child Usage Counter	

Examples

\Fetch_Record_Type_Project_Inclusions('DIA')\
\Fetch_Record_Type_Project_Inclusions('SCR')\

Examples

\Fetch_Record_Type_Project_Inclusions('BREQ')\
\scan(a)\
\if (bof(a))\

Record Type: \a: Record Type \

\endif\

Project Name: \a: Project Name\
Display Seq: \a: Display Seq\
Security Enforced: \if (a : Security Enforced = 'Y')\□\endif\

\endscan\

Fetch_Record_Type_States

Compatibility: Desktop App Version 3.35 and above.

This is a Master level command used to fetch States available for a particular Record Type, according to the ID Prefix specified.

\Fetch_Record_Type_States('<<ID Prefix>>')

Parameter

<<ID Prefix>>	Mandatory	Record identifier of the record.

Fields Available

Field	Description
State	
Record Type	
Description	
Activity Description	
Disabled	
In Progress	
Automatically Assign Record to Role	

Use as Source	
Consider as Open	
Use as Destination	
Upd dt	
Crt dt	
Note Behavior	

Examples

\Fetch_Record_Type_States('DIA')\
\Fetch_Record_Type_States('SCR')\

Examples

\Fetch_Record_Type_States('Req')\
\scan(a)\
\if(bof(a))\

Record Type: \a: Record Type \

\endif\

State:\ a : State\
Description: \a: Description\

\endscan\

Fetch_Record_Type_State_Allowed_Action

Compatibility: Desktop App Version 3.35 and above.

This is a Master level command used to fetch Record Types States Allowed Actions according to the ID Prefix specified. It fetches all the Actions available in the system for that particular Record Type in the current State.

\Fetch_Record_Type_State_Allowed_Action('<<ID Prefix>>')

Parameter

<<ID Prefix>>	Mandatory	Record identifier of the record.

Fields Available

Field	Description
State Allowed Action	
Record Type	
Allow	
State	

Examples

\Fetch_Record_Type_State_Allowed_Action('DIA')\
\Fetch_Record_Type_State_Allowed_Action('SCR')\

Examples

\Fetch_Record_Type_State_Allowed_Action('BREQ')\
\scan(a)\
\if (bof(a))\

Record Type: \a: Record Type \
\endif\
State:\ a : State\ \ a : State Allowed Action\
\endscan\

Fetch_Record_Type_State_Transition

Compatibility: Desktop App Version 3.35 and above.

This is a Master level command used to fetch State Transitions of Record Types, according to the ID Prefix specified. State Transitions is the process of updating the State field of a Record from the current State to another State. Only those Transitions that you choose to allow here will appear in the State change menu of various editors.

\Fetch_Record_Type_State_Transition('<<ID Prefix>>')\

Parameter

<<ID Prefix>>	Mandatory	Record identifier of the record.

Fields Available

Field	Description
State	
Transition	
Record Type	
Allow	
Lst upd by	
Crt dt	

Examples

\Fetch_Record_Type_State_Transition('BREQ')\
\Fetch_Record_Type_State_Transition('UC')\

Examples

\Fetch_Record_Type_State_Transition('UC')\
\VAR(LastState) \
\ LastState:=''\
\scan(a)\
\if (bof(a))\

Record Type: \a : Record Type \

\endif\
\if (LastState <> a : State)\

State: \a : State \
\endif\

Transition: \ a : Transition\

\SET(LastState, a : State)\
\endscan\

Fetch_Record_Type_Versioning_Setting

Compatibility: Desktop App Version 3.35 and above.

This Master level command is used to fetch the Record Types Versioning Settings, according to the Record Type Prefix specified. Record Versioning is the capability to maintain the older copies of the Records whenever the data is updated. Older copies of the Records are also called as Versions or Revisions. In TopTeam, we can enable Versioning for Repository Objects and Tracking Items Record Types.

\Fetch_Record_Type_Versioning_Setting('<<ID Prefix>>')

Parameter

<<ID Prefix>>	Mandatory	This is the record identifier.

Fields Available

Field	Description
Record Type	
Field Name	
Increment Version on Change	

Examples

\Fetch_Record_Type_Versioning_Setting('BREQ')\
\Fetch_Record_Type_Versioning_Setting('UC')\

Examples

\Fetch_Record_Type_Versioning_Setting('UC')\

\scan(a)\
\if (bof(a))\

Record Type: \a : Record Type\

\endif\
Field Name: \ a : Field Name\
Increment Version On Change: \if (a : Increment Version On Change = 'Y')\✓\endif\
\endscan\

Fetch_Record_Type_Columns

Compatibility: Desktop App Version 3.35 and above.

This Master level command is used to fetch the Record Types Columns, according to the Record Type Prefix specified. The fields can be customized for each Record Type. You can create fields of the desired types (strings, integers etc.) and specify its Display Captions.

\Fetch_Record_Type_Columns('<<Display Prefix of Record Type>>')

Parameter

<<Display Prefix of Record Type>>	Mandatory	This is the Record identifier.

Examples

\Fetch_Record_Type_Columns('BREQ')\
\Fetch_Record_Type_Columns('UC')\

Example

\Fetch_Record_Type_Columns('UC')\
\scan(a)\
\if (bof(a))\

Record Type: \ a : Record Type\

Column Names

\endIf\

\ a : Field Name\ \ a : ECL_DB_COL_NAME\

\endscan\

Fields Available

Field	Description
Record Type	
Field Name	
DB Column Name	
ID	
BASE ID	
Record Type ID	
Display Type	
Visible	
List ID	
Dep on ID	
Display Seq	
Inc Version	
User Field Kind	
Is Custom Field	
Field in Use	

Insert_Record_Type_Image

Compatibility: Desktop App Version 8.15 and above.

This command is used to generate a record type image of a desired width, height and image type format. If the image is forced to resize, it will be resized based on the height and width specified by you. Else, it will be printed in its default size.
Image format cannot be set, that is you cannot specify the image format. Default image format is JPG. This is a miscellaneous command and can be used independently in the template.

Insert_Record_Type_Image ('<<Record Type name Or ID Prefix>>', '<<Width>>', '<<Height>>', '<<Image Size Unit>>', '<<Is_Force_Size>>')

Parameters

<<Record Type Name or its ID Prefix>>	Mandatory	You can specify, Record type Singular Name OR Record type Plural Name OR Record type ID Prefix OR Record Id with prefix
<<Width>>	Optional	Set desired Width of the diagram to display in the output.
<<Height>>	Optional	Set desired height of the diagram to display in the output.
<<Image size Unit>>	Optional	This parameter contains one of the value from the following: **CMs** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in cms. **Inches** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in Inches. **Pixels** - If this is specified, the sizing dimensions i.e. height and width of the diagram will be measured in Pixels.

		By default the value of this parameter is "Pixels".
<<Is_Force_Size>>	Optional	**By default, this parameter is False.** So, the Image will be resized only if the actual Image size is greater than the specified size. If the actual size of the image is small and required specified size is greater, set this value to True to enforce sizing.

Fields Available

No fields are available for this command.

Examples

\Insert_Record_Type_Image('UC')\
\Insert_Record_Type_Image('UC-551')\

Example

\Insert_Record_Type_Image('OLE')\
\Insert_Record_Type_Image('UCS',2,1, 'cms')\
\Insert_Record_Type_Image ('ACTR',2,1, 'cms', true)\
\Insert_Record_Type_Image('Context Diagrams',4,4, 'INCHES')\

Insert Trace Matrix

Insert_Trace_Matrix_by_Condition

Compatibility: Desktop App Version 4.50 and above.

This is a miscellaneous command used independently at any place in the template. It is used to output a Multi-Level Trace hierarchy tabular report for a specified Starting Record Type Prefix. The maximum level generated in this report is up to five. If you specify more parameters, this command will generate an error.

\Insert_Trace_Matrix_by_Condition('<<Starting Record Type ID Prefix>>', '<<Single_Link_Type>>', '<<Container Path>>', '<<Filter Condition>>', '<< Sort By Hierarchy>>', '<<Level2 Linked Record Type Names>>', '<<Level3 Linked Record Type Names>>', '<<Level4 Linked Record Type Names>>', '<<Level5 Linked Record Type Names>>',)

Compatibility: Desktop App Version 12.3 and above.

\Insert_Trace_Matrix_by_Condition('<<Starting Record Type ID Prefix>>', '<<Multiple_Link_ Types>>', '<<Container Path>>', '<<Filter Condition>>', '<< Sort By Hierarchy>>', '<<Level2 Linked Record Type Names>>', '<<Level3 Linked Record Type Names>>', '<<Level4 Linked Record Type Names>>', '<<Level5 Linked Record Type Names>>',)

Parameters

<<Starting Record Type ID Prefix>>	Mandatory	This is the Record Type ID Prefix. There must be only one ID Prefix per Record Type. Otherwise an error will occur.
<<Link Types>>	Mandatory	This is the Trace Type for which Trace Records are to be fetched. If the value specified for this parameter is blank then the command will forward Trace Type default, "Traces Into". • If you are using *TopTeam Desktop App* version 12.3 and above, you can fetch multiple Trace Types for either forward or reverse direction. • If you are using *TopTeam Desktop App* version less than 12.3, you can fetch only one Trace Type. Otherwise, an error will occur.
<<Container Path>>	Optional	The Container Path can be a Project, Folder or a Requirements Document. When this parameter is blank, it will default to the Project Path, which is currently selected in the DocProcessor Generate dialog.
<<Filter	Optional	Specify the filter condition.

Condition>>		
<< Sort By Hierarchy>>	Optional	This Boolean parameter is applicable for generating Records at the first level i.e. Records of the Starting Record Type ID Prefix. The following scenarios illustrate how this parameter affects the output: • If the Requirements Type is specified, this parameter value is specified as **FALSE** and the Requirements Document is specified in the Container Path. This command will sort the Records by Parent Requirement, Requirements Titles and Order of Requirements in the Requirements Tree. • If any Requirements Type is specified as the first parameter and this value is **TRUE** and the Requirements Document is not specified in the Container Path parameter, then it will generate an error message. • If Project or Folder Path is specified, the parameter is not used and the Records are generated in the sequence in which they appear in the Repository Explorer.
<<Level2 Linked Record Type ID Prefixes>>	Optional	These are the comma separated second level Record Type ID Prefixes. This command will fetch all Link Records of the Master/Starting Record Type. If you want to fetch all Traces for the Starting Record Type, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for Starting Record Type. The Header Column heading will appear as <<ALL>>. If this parameter is not specified and only the first five

		parameters are specified in the command, then <All>> Traces will be fetched for the Master Level Record Type. This is true only for this parameter and not for parameters Level3 and higher.
<<Level3 Linked Record Type ID Prefixes>>	Optional	These are the comma separated third level Record Type ID Prefixes. This command will fetch all Linked Records of the second level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for second level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for second level Trace Records. The Header Column heading will appear as <<ALL>>.
<<Level4 Linked Record Type ID Prefixes>>	Optional	These are the comma separated fourth level Record Type ID Prefixes. This command will fetch all Linked Records of the third level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for third level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for third level Trace Records. The Header Column heading will appear as <<ALL>>.
<<Level5 Linked Record Type ID Prefixes>>	Optional	These are the comma separated fifth level Record Type ID Prefixes. This command will fetch all Linked Records of the fourth level Records.

		Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for fourth level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for fourth level Trace Records. The heading will appear as <<ALL>>.

Sorting: For all second level onward Traces, sorting will occur in the same way as it is done under the Traceability tab, i.e. Link Type Names, Link Display Order, Record Types and Trace Object Names.

Examples

\Insert_Trace_Matrix_by_Condition('UC','Traced from', 'Rental Management',' "Priority" = "High" ', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

\Insert_Trace_Matrix_by_Condition('BREQ', 'Traces Into','Requirements Document','','True', 'ODOC,CTX,SCR', 'DIA,NMP','MOD', 'STT')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\
\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

Insert_Trace_Matrix_By_Filter

Compatibility: Desktop App Version 4.50 and above.

This miscellaneous command can be used independently at any place in the template. It is used to display the Multi-Level Trace hierarchy tabular report for the Starting Record Type Prefix. The

maximum level generated in this report is up to five. If you specify more parameters, this command will generate an error message.

\Insert_Trace_Matrix_by_Filter('<<Single Trace Type>>', '<<Starting Record Type ID Prefix>>', '<<Container Path>>', '<<Filter Name>>', '<<Sort By Hierarchy>>', '<<Level2 Record Type Names>>', '<<Level3 Record Type Names>>', '<<Level4 Record Type Names>>', '<<Level5 Record Type Names>>',)

Parameters

<<Single Trace Type>>	Optional	This is the Trace Type for which Trace Records are to be fetched. If this parameter is blank, the command will default to "Traces Into", which is the Forward Trace type. Only one Trace Type is acceptable. An error message will display if more than one Trace Type is specified.
<<Starting Record Type ID Prefix>>	Mandatory	This is the ID Prefix of the Record. There must be only one ID Prefix per Record Type. An error will display if multiple ID Prefixes are specified or no ID Prefix is specified.
<<Container Path>>	Optional	The Container Path can be a Folder or a Requirement Document. If the parameter is blank, the default Project which is currently selected in the Doc Engine dialog will display.
<<Filter Name>>	Optional	This is the specified Filter Name for the Starting Record Type Prefix.
<< Sort By Hierarchy>>	Optional	This Boolean parameter generates records at the first level i.e. records of the Starting Record Type ID Prefix. The following scenarios illustrate how this parameter affects the output: • If the Requirements Type is specified, this parameter value is specified as **FALSE** and the Requirements Document is specified in the Container Path, then this command will sort the records by Parent Requirement, Requirements Titles and Order of

		Requirements in the Requirements Tree. • If any Requirements Type is specified as the first parameter, this parameter value is specified as **TRUE** and Requirements Document is not specified in the Container Path parameter, then it generates an error message. • If Project or Folder Path is specified, then the parameter is not used and Records will be generated in the sequence in which they appear in the Repository Explorer.
<<Level2 Record Type ID Prefixes>>	Optional	These are the comma separated second level Record Type ID Prefixes. This command will fetch all Linked Records of the Master/Starting Record Type. If you want to fetch all Traces for Starting Record Type, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for Starting Record Type. The Header Column will display as <<ALL>> in above case. If this parameter is not specified and only the first five parameters are specified in the command, then <All>> Traces will be fetched for the Master level Record Type. This is only true for this parameter and not for Level3 and higher parameters.
<<Level3 Record Type ID Prefixes>>	Optional	These are the comma separated third level Record Type ID Prefixes. This command fetches all Linked second level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter.

		If you want to fetch all Traces for second level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the traces for second level Trace Records. The Header Column will display as <<ALL>>.
<<Level4 Record Type ID Prefixes>>	Optional	These are the comma separated fourth level Record Type ID Prefixes. This command will fetch all Linked third level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for third level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for third level Trace Records. The Header Column will display as <<ALL>>.
<<Level5 Record Type ID Prefixes>>	Optional	These are the comma separated fifth level Record Type ID Prefixes. This command will fetch all Linked forth level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for fourth level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for fourth level Trace Records. The Header Column will display as <<ALL>>.

Sorting: For all second level onward Traces, the sorting will occur in the same way as it is done in the Traceability tab i.e. Link Type Names, Link Display Order, Record Types and Trace Object Names.

Examples

\Insert_Trace_Matrix_by_Filter('UC','Traces Into','Folder 1',' priority high', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\
\Insert_Trace_Matrix_by_Filter('BREQ','Traces Into','Requirements Document','','True', 'ODOC,CTX,SCR', 'DIA,NMP','MOD', 'STT')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

\Insert_Trace_Matrix_by_Filter('UC','Traced from', 'Rental Management','My Use Cases ', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

Insert_Trace_Matrix_By_IDs

Compatibility: Desktop App Version 4.50 and above.

This miscellaneous command can be used independently at any place in the template. It is used to display the Multi-Level Trace hierarchy tabular report for multiple Record Type IDs, with or without Prefixes. The maximum level covered in this report is five. If user attempts to go higher than this level, an error message will display.

We can use this command in case of Baselines to view the Trace Records for a Baseline Record.

\Insert_Trace_Matrix_By_IDs('<<Multiple Record IDs With OR Without Prefix>>', '<<Single Trace Type>>', '<<Level2 Record Type ID Prefixes>>', '<<Level3 Record Type ID Prefixes>>', '<<Level4 Record Type ID Prefixes>>', '<<Level5 Record Type ID Prefixes>>')

Parameters

<<Multiple Record IDs	Mandatory	This parameter can contain Multiple Record IDs with or without Prefixes and can also contain a single ID

With OR Without Prefix>>		in the form of a field of the Master command as follows: E.g.: \Fetch_Use_Case_By_Id('UC-2838')\ \scan(a) \ \a:Name\ [\ a: Id\] \Insert_Trace_Matrix_By_Ids(a: Id, 'Uses', '','','','')\ \endscan\ The Record with the specified ID will be fetched only if a user is a Team Member of the Project. • If the user is System Admin, then all the Records will be fetched specified in this parameter. • If no value is specified, an error message will display. • If the ID of a deleted Record is specified, this command will not fetch the deleted Record field of the Master command.
<<Single Trace Type>>	Optional	This is the Trace Type for which Trace Records are to be fetched. If the parameter is blank, the command will default to Traces Into which is the Forward Trace type. Only one Trace Type is acceptable. Otherwise, this command will generate an error message if more than one Trace Type is specified.
<<Level2 Record Type ID Prefixes>>	Optional	These are the comma separated second level Record Type ID Prefixes. This command will fetch all Linked Records of the Master/Starting Record Type. If you want to fetch all Traces for the Starting Record Type, then specify the parameter value as '<<All>>'. Even if blank quotes ('') are specified, the command will fetch all the Traces for Starting Record Type.

		The Header Column will display as <<ALL>>.If this parameter is not specified and only first two parameters are specified in the command, then <All>> Traces will be fetched for the Master level Record Type. This is true only for this parameter and not for Level3 and higher parameters.
<<Level3 Record Type ID Prefixes>>	Optional	These are the comma separated third level Record Type ID Prefixes. This command fetches all Linked second level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for second level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for second level Trace Records. The Header Column will display as <<ALL>>.
<<Level4 Record Type ID Prefixes>>	Optional	These are the comma separated fourth level Record Type ID Prefixes. This command fetches all Linked second level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for third level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for third level Trace Records. The Header Column will display as <<ALL>>.
<<Level5 Record Type	Optional	These are the comma separated fifth level Record Type ID Prefixes. This command fetches all Linked

ID Prefixes>>		second level Records. Linked Records are fetched as per the Record Type ID Prefixes specified in this parameter. If you want to fetch all Traces for fourth level Trace Records, specify the parameter value as '<<All>>'. If blank quotes ('') are specified, the command will fetch all the Traces for fourth level Trace Records. The Header Column will display as <<ALL>>.

Examples

\Insert_Trace_Matrix_By_Ids('ACTR-382, ACTR-384, ACTR-383, uc-682, UC-680, UC-681, req-300', 'Traces Into', 'ODOC,CTX,SCR','DIA,NMP','MOD','STT')\

\Insert_Trace_Matrix_By_Ids('BREQ-140, FEAT-134, BREQ-135, BREQ-136, BREQ-137','Traces Into','', '', 'MOD')\
\Insert_Trace_Matrix_By_Ids('ACTR-382, ACTR-384, ACTR-383, uc-682, UC-680, UC-681, req-300','Traces Into', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\
\Insert_Trace_Matrix_By_Ids('BREQ-140, FEAT-134, BREQ-135, BREQ-136, BREQ-137', 'Traces Into', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD','STT')\

\Insert_Trace_Matrix_By_Ids('BREQ-140, FEAT-134, BREQ-135, BREQ-136, BREQ-137','Traces Into','', '', 'MOD')\

\Insert_Trace_Matrix_By_Ids('BREQ-140, FEAT-134, BREQ-135, BREQ-136, BREQ-137', 'Traces Into')\

Set_Trace_Matrix_Cell_Font

Compatibility: Desktop App Version 4.50 and above.

This miscellaneous command is used before Insert_Trace_Matrix commands are called. This command is used to set the cell's Font Properties of the Trace Matrix that will be outputted using Insert_Trace_Matrix.

\Set_Trace_Matrix_Cell_Font('<<Font Name>>', '<<Font Size>>', '<<Font Style>>')

Parameters

<<Font Name>>	Optional	This is the Name of the Font supported by the Windows Operating System. If the Font name specified is invalid, this command will result into an error. By default the Font name supported by this command is Verdana.
<<Font Size>>	Optional	This is the Font size parameter. By default the value of this parameter is 8.
<<Font Style>>	Optional	Specify the Font styles for this command. The possible values for this parameter are as follows: • 'B' - for displaying the text in Bold. • 'I' - for displaying the text in Italics. • 'U' - for displaying the text as Underlined. If the style specified is other than the **'B','I'** or **'U',** an error message will display. By default, no style is applied to the Font.

Examples

\Set_Trace_Matrix_Cell_Font()\
\Set_Trace_Matrix_Cell_Font('Tahoma','10', 'B')\
\Set_Trace_Matrix_Cell_Font('Times New Roman','15', 'B,I')\
\Set_Trace_Matrix_Cell_Font('Verdana','10', 'B,I,U')\
\Set_Trace_Matrix_Cell_Font('','10')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Set_Trace_Matrix_Header_Font('Arial','10', 'B')\

\Set_Trace_Matrix_Cell_Font('Arial', '9', '')\

\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'SCR')\

\Set_Trace_Matrix_Header_Font('Book Antiqua', '10', 'B')\
\Set_Trace_Matrix_Cell_Font('Times New Roman', '9', '')\

\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

\Set_Trace_Matrix_Header_Font('Verdana', '10', 'U,I')\
\Set_Trace_Matrix_Cell_Font('Times New Roman', '8', 'B')\

\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'SCR')\

\Set_Trace_Matrix_Header_Font('Verdana', '10', 'U,I')\
\Set_Trace_Matrix_Cell_Font('Times New Book Roman', '8', 'B')\
\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'True', 'SCR')\

Set_Trace_Matrix_Header_Font

Compatibility: Desktop App Version 4.50 and above.

This miscellaneous command is used before Insert_Trace_Matrix commands are called. It is used to set the Font Properties of the Header Row of the Trace Matrix output table.

\Set_Trace_Matrix_Header_Font('<<Font Name>>', '<<Font Size>>', '<<Font Style>>')

Parameters

<<Font Name>>	Optional	This is the name of the Font supported by the Windows Operating System. If the Font name specified is invalid, an error message will display. By default, the Font name supported by this command is Arial.

<<Font Size>>	Optional	This is the Font size parameter. By default, the value of this parameter is 10.
<<Font Style>>	Optional	Specify the Font style for this command. The possible values for this parameter are as follows: • 'B' - for displaying the text in Bold. • 'I' - for displaying the text in Italics. • 'U' - for displaying the text as Underlined. If the style specified is other than the **'B','I'** or **'U',** an error message will display. By default, the 'Bold' style will be applied to the Font.

Examples

\Set_Trace_Matrix_Header_Font()\
\Set_Trace_Matrix_Header_Font('Arial','10', 'B')\
\Set_Trace_Matrix_Header_Font('Book Antiqua', '10', 'B')\
\Set_Trace_Matrix_Header_Font('Verdana', '10', 'U,I')\
\Set_Trace_Matrix_Header_Font('','10')\

Examples

\Set_Trace_Matrix_Header_Font('Verdana', '10', 'U,I')\
\Insert_Trace_Matrix_by_Condition('Actr','Traced From','','', 'True', 'SCR')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1', '', 'True', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')

\Set_Trace_Matrix_Cell_Font('Times New Roman', '12', '')\
\Set_Trace_Matrix_Header_Font('Calibri', '12', 'B,U')\

\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1', '', 'True', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

Set_Trace_Matrix_Table_Properties

Compatibility: Desktop App Version 4.50 and above.

This miscellaneous command is used before Insert_Trace_Matrix commands are called. This command is used to set the Table Properties in the Trace Matrix output table.

\Set_Trace_Matrix_Table_Properties('<<Cell Width>>', '<<Show Record Id in the report or not(True or False)>>', '<<Record ID Position(Left or Right side of the name)>>', '<<Show the record type image>>')

Parameters

<<Cell Width>>	Optional	Specify the cell Width of the Trace Matrix table.
<<Show Id in the report or not(True or False)>>	Optional	This Boolean parameter confirms whether or not to display the ID in the Trace Matrix report. By default, the value of this parameter is **FALSE**.
<<ID Display Position(Left or Right side of the name)>>	Optional	If the second parameter <<Show ID in the report or not(TRUE or FALSE)>> is **TRUE**, then confirm which side of the name the user wants to display the ID. The following values are available for this parameter: Left Right You need to specify the above strings in the template. If the second parameter is **FALSE**, the value written in this parameter is ignored by the DocProcessor. By default, the value of this parameter is **Left.**

<<Display the record type image>>	Optional	This parameter displays Record Type image in front of the Record name. By default, the value of this parameter is **FALSE**.

Examples

\Set_Trace_Matrix_Table_Properties('200', 'TRUE', 'Right', 'True')\
\Insert_Trace_Matrix_by_Condition('UC','Traces Into', 'Rental Management','', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Set_Trace_Matrix_Table_Properties('200', 'TRUE', 'Right', 'True')\
\Insert_Trace_Matrix_by_Condition('UC', 'Traces Into', 'Folder 1','', 'False', 'ODOC,CTX,SCR', 'DIA,NMP', 'MOD', 'STT')\

\Set_Trace_Matrix_Table_Properties('300', 'True', 'Left', 'True')\
\Insert_Trace_Matrix_by_Condition('UC','Traces Into', 'Folder 1','', 'False', 'ODOC,CTX,SCR', 'DIA,NMP')\

\Set_Trace_Matrix_Table_Properties('300','False','','True')\
\Insert_Trace_Matrix_by_Condition('UC','Traces Into', 'Folder 1','', 'False', 'ODOC,CTX,SCR', 'DIA,NMP')\

Insert Chart Portlets

This command is used to generate a Dashboard Chart in the output template based on the Portlet's name.

Insert_Chart_Using_Portlet

Compatibility: Desktop App Version 4.50 and above.

This command generates Chart Portlets from the Dashboard into a Word document. It can be used independently at any place in the template. This command is not Project dependent.

Insert_Chart_Using_Portlet('<<Chart Portlet Name>>,'<<Width>>','<<Height>>','<<Image Size Unit>>','<<Image Type>>')

Parameters

<<Chart Portlet Name>>	Mandatory	Specify the name of the Dashboard Portlet.
<<Width>>	Optional	Set desired Width of the Portlet to be displayed in the output.
<<Height>>	Optional	Set desired Height of the Portlet to be displayed in the output.
<<Image Size Unit>>	Optional	This parameter contains one of the values from the following: • CMs - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in Centimeters. • Inches - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in Inches. • Pixels - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in Pixels. By default, the value of this parameter is Pixels.
<<Image Formats>>	Optional	The image formats supported by this command are:

		• EMF • WMF • JPEG You need to specify one of the above values in this parameter. This command will generate the diagram in the corresponding format specified by the user. By default, the diagram will be generated in EMF format.

NOTE:

- If a parameter for Width and Height is not supplied, this command automatically calculates the other parameter in proportion to the supplied value.
- When both Height and Width parameters are specified, this command generates the image with the specified Height and Width.
- If Width and Height are set to blank, the original size of the image will be generated.

Fields Available

There are no fields available for this command.

Examples

\Insert_Chart_Using_Portlet('New Items Assigned to you') \

\Insert_Chart_Using_Portlet('Requirements Assigned to Me (by State)', 20,8, 'cms', 'jpeg')\

\Insert_Chart_Using_Portlet('Requirements Assigned to Me (by State)', 10,20, 'inches', 'wmf')\

\Insert_Chart_Using_Portlet('Use Cases Assigned to Me (by State)', 350,280, 'pixels','jpeg')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\ Comments (Here "Use Cases Assigned to Me (by State)" is name of a Portlet defined in Dashboard.) \
\Insert_Chart_Using_Portlet('Use Cases Assigned to Me (by State)') \

Insert_Pie_Chart_Custom

Compatibility: Added in TopTeam Version 8.203.

This command is used to output a Pie Chart of desired Width and Height and in the required image type format.
This is a miscellaneous command and can be used independently in the template.

Insert_Pie_Chart_Custom('<<Chart Parameters>>,'<<Width>>','<<Height>>','<<Image Size Unit>>','<<Image Type>>')

Parameters

<<Chart_Parameters>>	Mandatory	The Chart will be displayed on the basis of these parameters: <<Chart Parameter>> is the only variable we use in the command but this variable is made of many Name, Value pairs separated by commas needed to create a Chart. Those Name, Value pairs are as below: **Note** 1. All the parameters must be in Name, Value pair format. **Syntax** Parameter Name = Parameter Value **Example** ChartName = Bar Chart for Use Cases 2. All Name, Value pairs must be comma separated. 3. Order of Name, Value pairs is not important.

		4. Spaces can be used but ENTER characters cannot be used. 5. Record Type - Use ID Prefix or Record Type Singular or Record Type Plural name. 6. Filter Name - Use any filters created from Dashboard, List or Tree or a Filter Condition - Specify the conditions to filter the Records. At a time only one filter facility will work and we cannot use Filter Condition and Filter Name both simultaneously to filter Records. 7. Space is Mandatory between Name, Value and '=' sign in Filter Condition. 8. For Filter Condition multiple conditions should be separated by 'and'. **Mandatory** 1. RecordType (Disp Prefix or Record Name Singular or Record Name Plural) 2. Distributionby 'RecordType=UC, Distributionby=State' **Optional** **Default values** ChartName= FilterName= FilterCondition= Show3D = True ShowMarks = False ShowBorder = False ShowXAxisText = True ShowDataPointsForZeroValue = False

		ShowTitle = True ShowFooter = False ShowLegend = True ShowSubTitle = False ShowPageNo = False ScaleLastPage = False Weightage = '' LegendPosition = Left SelectTheme = Microsoft@Excel ColorPalette = **Sample values** ChartName = Pie Chart for Use Cases FilterName = My Use Cases Filtero OR FilterCondition = "State" = "Brief" Show3D = True ShowMarks = False ShowBorder = False ShowXAxisText = True ShowDataPointsForZeroValue = False ShowTitle = True ShowFooter = False ShowLegend = True ShowSubTitle = False//?? ShowPageNo = False ScaleLastPage = False Weightage = 'Est. Cost/Est. Effort (Hrs)' LegendPosition = 'Left/Right/Top/Bottom' SelectTheme = 'Microsoft@Excel/Classic' ColorPalette = 'Classic/Excel/Modern'
<<Width>>	Optional	Set desired Width of the Diagram to display in the

		output. **Width Constraints** • If width is set blank, this command adjusts the Width as per the given Height. • If Height is also set blank then the original size of the image will be returned. • And when both Height and Width are given by the user, then this command sets the image with the specified Height and Width.
<<Height>>	Optional	Set desired Height of the diagram to display in the output. **Height Constraints** • If Height is set blank then image will adjust the Height as per the given Width. • If Width is also set blank then the original size of the image will be returned. • And when both Height and Width are given by the user, then this command will insert the image with the specified Height and Width.
<<Image Size Unit>>	Optional	This parameter contains one of the values from the following: • CMs - If this is specified the sizing dimensions i.e. Height and Width of the diagram will be measured in CMs. • Inches - If this is specified the sizing dimensions i.e. Height and Width of the diagram will be measured in Inches. • Pixels - If this is specified the sizing dimensions i.e. Height and width of the Diagram will be measured in Pixels.

		By default the value of this parameter is Pixels.
<<Image Type>>	Optional	Image formats supported by this command are: • EMF • WMF • JPEG You need to pass one of the values from above in this parameter. And this command will output the diagram in the corresponding format given by the user. By default, the diagram will be outputted in **EMF** format.

Fields Available:

No fields available for this command.

Examples

\ LChartParams:= 'RecordType=SREQ,Distributionby=State'\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=My Requirements'\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,FilterName=My Req Filter'\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=New Items,FilterName=Req Filter'\

\LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=New Items,FilterCondition="State" = "Brief"'\

Sample Template 1

\var(LChartParams)\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=Pie Chart for Requirements Assigned to Me,FilterName=MY Req Filter'\

\Insert_Pie_Chart_Custom(LChartParams, 10,10, 'cms', 'emf')\

Sample Template 2

\var(LChartParams)\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=Pie Chart for Requirements Assigned to Me,FilterCondition="Priority" = "High" '\

\Insert_Pie_Chart_Custom(LChartParams, 10,10, 'cms', 'emf')\

Sample Template 3

\Set_Project('$CURRENT_PROJECT$')\

**\ PROJECT_NAME **

\var(LChartParams)\

\ LChartParams:= 'RecordType=SREQ,Distributionby=State,ChartName=Pie Chart for Requirements Assigned to Me,FilterName=MY Req Filter'\

\Insert_Pie_Chart_Custom(LChartParams)\
\var(LChartParams)\

Record Type Disp Prefix
\ LChartParams:= 'RecordType=REQ,Distributionby=State, ChartName=Requirements Assigned to Me,FilterName='\
\Insert_Pie_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

Record Type Name Singular or Plural
\ LChartParams:= 'RecordType=Use Case,Distributionby=State, ChartName=Use Cases Assigned to Me ,FilterName='\
\Insert_Pie_Chart_Custom(LChartParams, 10,10, 'cms', 'wmf')\

ShowTitle
\ LchartParams:= 'RecordType=TC, Distributionby=State, ShowTitle = False,
ChartName=Pie Chart for Test Cases with State'\
\Insert_Pie_Chart_Custom(LchartParams,20,15, 'cms', 'jpeg')\

Filter
\ LchartParams:= 'RecordType=TC, Distributionby=State,ChartName=Pie Chart for Test Cases with State, FilterName= Priority'\
\Insert_Pie_Chart_Custom(LchartParams,20,15, 'cms', 'jpeg')\

Record Type Name Singular or Plural
\ LChartParams:= 'RecordType=Use Case,Distributionby=State, ChartName=Use Cases Assigned to Me'\
\Insert_Pie_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

Weightage
\ LChartParams:= 'RecordType=Use Case,Distributionby=State, ChartName=Use Cases Assigned to Me ,

Weightage = Est. Effort (Hrs)'\
\Insert_Pie_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

Insert_Bar_Chart_Custom

Compatibility: Added in TopTeam Version 8.203.

This command is used to output a Bar Chart of desired Width and Height and in the required image type format.
This is a miscellaneous command and can be used independently in the template.

Insert_Bar_Chart_Custom('<<Chart_Parameters>>','<<Width>>','<<Height>>','<<Image Size Unit>>','<<Image_Type>>')

Parameters

<<Chart_Parameters>>	Mandatory	The Chart will be displayed on the basis of these parameters: <<Charts Parameter>> is the only variable we use in the command but this variable is made of many Name, Value pairs separated by commas needed to create a Chart. Those Name, Value pairs are as below: **Note** 1. All parameters must be in Name, Value pair format. **Syntax** Parameter Name = Parameter Value **Example** ChartName = Bar Chart for Use Cases 2. All Name, Value pairs must be comma separated. 3. Order of Name, Value pairs is not important.

		4. Spaces can be used but ENTER characters cannot be used. 5. Record Type - Use ID Prefix or Record Type Singular or Record Type Plural name. 6. LimitStackToValues must be separated by a semicolon (;). 7. Filter Name - Use any filters created from Dashboard, List or Tree or Filter Condition - Specify the conditions to filter Records. At a time only one Filter parameter will work and we cannot use **Filter** Condition and Filter Name both simultaneously to filter Records. 8. Space is Mandatory between Name, Value and '=' sign in Filter Condition. 9. For Filter Condition multiple conditions should be separated by 'and'. **Mandatory** 1. RecordType 2. ShowBarsFor 3. DivideBarsBy RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority **Optional**

		Default values ChartName= FilterName= FilterCondition= Show3D = True ShowMarks = False ShowBorder = False ShowGrid = False ShowXAxisText = True ShowYAxisText = True ShowDataPointsForZeroValue = False ShowTitle = True ShowFooter = False ShowLegend = True ShowSubTitle = False ShowPageNo = False ScaleLastPage = False LimitStackTo = '' LegendPosition = Left SelectTheme = Microsoft@Excel ColorPalette = **Sample values** ChartName = Bar Chart for Use Cases FilterName = My Use Cases Filter

		OR FilterCondition = "State" = "Brief" Show3D = True ShowMarks = False ShowBorder = False ShowGrid = False ShowXAxisText = True ShowYAxisText = True ShowDataPointsForZeroValue = False ShowTitle = True ShowFooter = False ShowLegend = True ShowSubTitle = False//?? ShowPageNo = False ScaleLastPage = False LimitStackTo = '' LegendPosition = 'Left/Right/Top/Bottom' SelectTheme = 'Microsoft@Excel/Classic' ColorPalette = 'Classic/Excel/Modern'
<<Width>>	Optional	Set desired Width of the diagram to display in the output. **Width Constraints** If Width is set blank,then this command adjusts the Width as per the given Height. If Height is also set blank then the original size of the image will be returned.

		And when both Height and Width are given by the user, then this command sets the image with the specified Height and Width.
<<Height>>	Optional	Set desired Height of the diagram to display in the output. **Height Constraints** If Height is set blank then image adjusts the Height as per the given Width. If Width is also set blank then the original size of the image will be returned. And when both Height and Width are given by the user, then this command will insert the image with the specified Height and Width.
<<Image size Unit>>	Optional	This parameter contains one of the values from the following: CMs - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in CMs. Inches - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in Inches. Pixels - If this is specified, the sizing dimensions i.e. Height and Width of the diagram will be measured in Pixels. By default the value of this parameter is Pixels.
<<Image Type>>	Optional	Image format supported by this command are: EMF WMF JPEG

		You need to pass one of the values from above in this parameter. And this command will output the diagram in the corresponding format given by the user. By default, the diagram will be outputted in EMF format.

Fields Available:

No fields are available for this command.

Examples

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority'\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,ChartName=Use Cases'\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,FilterName=Use Cases Filter'\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,FilterCondition="State" = "Brief"'\

\LChartParams:='RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,ChartName=Bar,FilterCondition= "Priority" = "High"'\

Sample Template 1

\var(LChartParams)\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,ChartName=Bar Chart for Use Cases,FilterName=My Use Cases Filter'\

\Insert_Bar_Chart_Custom(LChartParams, 10,10, 'cms', 'emf')\

Sample Template 2

\var(LChartParams)\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,ChartName=Bar Chart for Use Cases,FilterCondition= "state" = "Brief" '\

\Insert_Bar_Chart_Custom(LChartParams, 10,10, 'cms', 'emf')\

Sample Template 3

\Set_Project('$CURRENT_PROJECT$')\
**\ PROJECT_NAME **

\var(LChartParams)\

\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Priority,ChartName=Bar Chart for Use Cases,FilterName=My Use Cases Filter'\

\Insert_Bar_Chart_Custom(LChartParams)\

Sample Template 4

\Set_Project('$CURRENT_PROJECT$')\
**\ PROJECT_NAME **
\var(LChartParams)\

ShowGrid
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True, FilterName=,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

ShowMarks
\ LChartParams:= 'RecordType=UC,ShowBarsFor=State,DivideBarsBy=Complexity, ShowMarks=True,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

LegendPosition
\ LChartParams:= 'RecordType=UC, ShowBarsFor=State,DivideBarsBy=Level, LegendPosition=Bottom, FilterName=,ChartName=Requirements Assigned to Me'\

Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

ShowLegend
\ LChartParams:= 'RecordType=UC, ShowBarsFor=State, DivideBarsBy=Level, ShowLegend=False, FilterName=, ChartName=Requirements Assigned to Me,'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

ShowDataPointsForZeroValue
\ LChartParams:= 'RecordType=Req, ShowBarsFor=State,DivideBarsBy=Priority,

FilterName=,ShowDataPointsForZeroValue=True,ShowGrid=True,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

LimitStackTo
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True,
LimitStackTo=Very High;High;Medium;Low,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

\var(LChartParams)\
LimitStackTo
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True,
LimitStackTo=Very High;High;Medium;Low,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

LimitStackTo
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True,
LimitStackTo=Very High;High;Medium,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

LimitStackTo
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True,
LimitStackTo=Very High;High,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')\

LimitStackTo
\LChartParams:='RecordType=UC, ShowBarsFor=State,DivideBarsBy=Complexity,ShowGrid=True,
LimitStackTo=Very High,ChartName=Requirements Assigned to Me'\

\Insert_Bar_Chart_Custom(LChartParams, 20,20, 'cms', 'wmf')

Commands to Insert Application Logo and Company Logo

These commands are used to output Logos of desired width, height and image type format.

\Insert_Application_Logo()\

\Insert_Company_Logo()\

Compatibility: Desktop App Version 7.0 and above.

They are miscellaneous commands and can be used independently in the template.

Insert_Application_Logo('<<Image Path>>','<<Width>>','<<Height>>','<<Image Size Unit>>','<<Is_Force_Size>>')

Insert_Company_Logo('<<Image Path>>','<<Width>>','<<Height>>','<<Image Size Unit>>','<<Is_Force_Size>>')

Parameters

<<Image Path>>	Optional	If the Image Path is blank then it will display default Application Logo. Default Application Logo Image Name = 'Application_Logo.JPG' Default Company Logo Image Name = 'Company_Logo.JPG' If Image Path is empty, then the above default images will be fetched from the folder where TopTeam application is running.
<<Width>>	Optional	Set desired Width of the Logo to be displayed in the output.
<<Height>>	Optional	Set desired Height of the Logo to be displayed in the output.

<<Image Size Unit>>	Optional	This parameter contains one of the values from the following • CMs - If this is specified, the sizing dimensions i.e. Height and Width of the Logo will be measured in Centimeters. • Inches - If this is specified, the sizing dimensions i.e Height and Width of the Logo will be measured in Inches. • Pixels - If this is specified, the sizing dimensions i.e. Height and Width of the Logo will be measured in Pixels. By default the value of this parameter is "Pixels".
<<Is_Force_Size>>	Optional	By default, this parameter is False. So image will be resized, only if actual image size is greater than the specified size. If actual size of image is small and required specified size is greater, then only set this value to True to enforce sizing.

Fields Available

There are no fields available for these commands.

Examples

\Insert_Application_Logo()\
\Insert_Company_Logo()\

Sample Templates

Insert_Application_Logo()
\Insert_Application_Logo()\
Insert_Company_Logo()
\Insert_Company_Logo()\

Insert_Application_Logo('',2,1, 'cms', 'EMF')

\Insert_Application_Logo(‘’,2,1, ‘cms’, ‘EMF’)\
Insert_Company_Logo(‘’,2,1, ‘cms’, ‘EMF’)
\Insert_Company_Logo(‘’,2,1, ‘cms’, ‘EMF’)\

Insert_Application_Logo(‘Application_Logo.JPG’,2,1, ‘cms’, ‘EMF’)
\Insert_Application_Logo(‘Application_Logo.JPG’,2,1, ‘cms’, ‘EMF’)\
Insert_Company_Logo(‘Company_Logo.JPG’,2,1, ‘cms’, ‘EMF’)
\Insert_Company_Logo(‘Company_Logo.JPG’,2,1, ‘cms’, ‘EMF’)\

Insert_Application_Logo(‘D:\Images\Application_Logo.JPG’,2,1, ‘cms’, ‘EMF’)
\Insert_Application_Logo(‘D:\Images\Application_Logo.JPG’,2,1, ‘cms’, ‘EMF’)\
Insert_Company_Logo(‘D:\Images\Company_Logo.JPG’,2,1, ‘cms’, ‘EMF’)
\Insert_Company_Logo(‘D:\Images\Company_Logo.JPG’,2,1, ‘cms’, ‘EMF’)\

\var(LImagePath)\
\ LImagePath:= ‘D:\Images\Company_Logo.JPEG’\
\Insert_Application_Logo(LImagePath)\

Baseline Commands

Fetch_Compare_Baselines

Compatibility: Desktop App Version 4.5 and above.

This command fetches the data of two Baselines and compares these two Baselines for modifications done in the Records since the Baselines were created.

\Fetch_Compare_Baselines(‘<<Name of first Baseline>>’, ‘<<Name of second Baseline>>’)

Parameters

<<Name of first Baseline>>	Mandatory	Specify the first Baseline name for which you want to compare data. In case of the current Baseline, the Baseline name should be Current.

		This command does not support As_Of_Date feature.
<<Name of second Baseline>>	Mandatory	Specify the second Baseline name for which you want to compare data. In case of the current Baseline, the Baseline name should be Current. This command does not support the As_Of_Date feature.

Examples

\Fetch_Compare_Baselines('Baseline1', 'Baseline 3')\
\Fetch_Compare_Baselines('BSL-Versioning is OFF for all record types', 'BSL1 - UC Records Inserted')\
\Fetch_Compare_Baselines('BSL1 - UC Records Inserted', 'BSL2 - UC Records Modified')\

Fields Available

Field	Description
Baseline1	
Baseline2	
Name	
Name1	
Version1	
Upd Dt1	
Upd By1	
Priority1	
State1	
Owner1	
Functional Area1	

Name2	
Version2	
Upd Dt2	
Upd By2	
Priority2	
State2	
Owner2	
Record Type	
Record Type1	
Record Type2	
Functional Area2	
Is Header	This is a Boolean field; this is **TRUE** if this is the Header Record.
Indentation Level	This indicates the level of the Record in a hierarchy.
Action	This field value checks which operation is performed on the Baseline Record i.e. Insertion, Deletion or Update. This field can have the following values: • Added in Baseline1 • Added in Baseline2 • Modified • Deleted from Baseline1 • Deleted from Baseline2 • No Change

Fetch_Compare_Records()

Compatibility: Desktop App Version 6.20 and above.

This secondary command (under the Master command for Baselines) is used to compare any two Records, i.e. shared source and destination Records.

\Fetch_Compare_Records(<<Field for Unique Id of the record>>, <<Field for Unique Id of the record>>, <<Show All Fields>>, <<Show Merged Output As Rich Text>>)

Parameters

<<Field for Unique ID of the Record>>	Mandatory	Pass field containing the unique ID of Record1.
<<Field for Unique ID of the Record>>	Mandatory	Pass field containing the unique ID of Record2.
<<Show All Fields>>	Optional	These are All or Difference fields. If you specify **TRUE** or **All**, then compare versions will display all fields. By default, its value is **FALSE** and therefore, the comparison of fields having different values will be displayed.
<<Show Merged Output As Rich Text>>	Optional	This is Rich or Simple Text. If you specify **TRUE**, then Merged fields output will display as Rich Text. By default, its value is **FALSE** and therefore, the comparison of Merged fields will display as Simple Text.

Fields Available

All fields of selected Record Type including custom fields, may be inserted into the document.

This command is dependent on other commands. It can compare four types of data including, rich text, simple text, diagrams and data other than these three types: date, name, state, etc.

Use the **\FRTF()** command for generating rich text data.

For generating diagrams, use the following commands:
\Insert_Field_As_Diagram_In_CMs() - generates the diagram in Centimeters.
\Insert_Field_As_Diagram_In_Inches() - generates the diagram in Inches.

For generating simple text, the **Insert_Differences()** command is used. You can generate the merged view of differences between the two versions. You can use the same command to generate rich text data in simple text format.

For generating data other than simple text such as Create Date, Name, Owner, State, Priority etc., simply write down the field names in the template without using the functions.

Example

\Fetch_Compare_Records(a: RecordId1, a:RecordId2)\

Examples

\Declare_Variable('Baseline_Name1', 'String')\
\Declare_Variable('Baseline_Name2', 'String')\
\Prompt_For_Variable_Values('Baseline_Name1', 'Baseline_Name2')
\scan(a) \

Package: [\ a : ID\] \a : Name\
Baselines Compared: \Baseline_Name1\ and \ Baseline_Name2\

\ Fetch_Compare_Package_Baselines(a:Id, Baseline_Name1, Baseline_Name2)\

\scan(b)\\if (Bof(b))\

Record	Version1	Version2	Person	Date	Comparison Status

\endif\
\if(b : Comparison Status = 'Added')\

\ b : Id \ \ b : Name \	\ b : Version \	\ b : Version2 \	\ b : Person \	\ b : Date \	Added in \b :Baseline2\

\endif\

\if(b : Comparison Status = 'Removed')\

\ b : Id \ \ b : Name \	\ b : Version\	\ b : Version2 \	\ b : Person \	\ b : Date \	Removed from \b :Baseline1\

\endif\

\if(b : Comparison Status = 'Modified')\

\ b : Id \ \ b : Name \	\ b : Version\	\ b : Version2 \	\ b : Person \	\ b : Date \	\ b : Comparison Status\

\Fetch_Compare_Records(b : RecordId1, b : RecordId2)\\scan(c)\\if(Bof(c))\

Field	\ b :Baseline1 \	\ b :Baseline2 \

\endif\

\if(c : Type = 'RTF')\

\c: Field\	\FRtf(c: Value1)\	\FRtf(c: Value2)\

\elsif(c : Type = 'Diagram')\

\c: Field\	\Insert_Field_As_Diagram_In_CMs (c:Value1, '9', '', 'EMF')\	\Insert_Field_As_Diagram_In_CMs (c:Value2, '9', '', 'EMF')\

\elsif(c : Type = 'Text')\

\c: Field\	\Insert_Differences (c: Value1, c: Value2)\

\else\

\c: Field\	\c: Value1\	\c:Value2\

\endif\\endscan\
\endif\

\endscan\
\endscan\

Fetch_Compare_Versions

Compatibility: Desktop App Version 4.5 and above.

This secondary command is under the Master command and is used for Baselines.

\Fetch_Compare_Versions(<<Field for ID of the record>>, <<Field for Unique ID of the record>>, <<Field for Unique ID of the record>>, <<Show All Fields>>,<<Show Merged Output As Rich Text>>)

Parameters

<<Field for ID of the record>>	Mandatory	Specify the ID of the Record for which Versions will be compared.
<<Field for Unique ID of the record>>	Mandatory	Pass Version field of Record Version1.
<<Field for Unique ID of the record>>	Mandatory	Pass Version field of Record Version2.
<<Show All Fields>>	Optional	These are All or Difference fields. If you specify **TRUE** or **All**, then Compare Versions will display all fields. By default, its value is **FALSE** and therefore, the comparison of fields having different values will

		be displayed.
<<Show Merged Output As Rich Text>>	Optional	This is Rich or Simple Text. If you specify **TRUE**, then merged fields output will display as Rich Text. By default, its value is **FALSE** and therefore, the comparison of merged fields will display as Simple Text.

Fields Available

Field	Description
Field	Name of the field.
Value1	Value of Field for Version1.
Value2	Value of Field for Version2.
Version1	First Version of the Record.
Version2	Second Version of the Record.
Is Modified	This is used to check whether or not the current field is modified in the second version.
Type	This is used to indicate which type of data is being compared 'RTF, 'Diagram', 'Text', or simple data.

Examples

\Fetch_Compare_Versions(a: Id, a: VersionId1, a:VersionId2)\
\Fetch_Compare_Versions(a: Id, a: VersionId1, a:VersionId2, 'All')\
\Fetch_Compare_Versions(a: Id, a: VersionId1, a:VersionId2, 'Difference')\

Details

The Fetch_Compare_Versions() command compares the two versions of a Record.
This is the secondary command and is always used under the Master command.

This command is specially designed for the Fetch_Compare_Baselines() command. The Compare Baselines editor allows you to view version comparisons, and consequently allows the same functionality in the document output.

It has a dependency on other commands as well. This command can compare four types of data: data of type rich text, simple text, diagrams and data other than these three types like date, name, state, etc.

For generating rich text data, you can use the **\FRTF()** command.

You can make use of the following commands for generating diagrams:
\Insert_Field_As_Diagram_In_CMs() - generates the diagram in Centimeters.
\Insert_Field_As_Diagram_In_Inches() - generates the diagram in Inches.
The above two commands are specially designed for 'Fetch_Compare_Versions()' command in order to show the images in the specified format.

For generating simple text data, the **Insert_Differences()** command is used. You can view the merged view of the differences present between the two versions. You can use the same command to display rich text data in simple text format.

For generating data other than simple text such as Create Date, Name, Owner, State, Priority, etc., simply enter the field names in the template without making use of any functions.

Insert_Differences

Compatibility: Desktop App Version 4.5 and above.

This secondary command is always used under the Master command. Insert_Differences() command is used to compare two text fields. It can be simple text fields or rich text fields however, this command will only display the differences in simple text format.

It is specially designed to use with Fetch_Compare_Versions() command in order to display the merged differences between fields. This command uses colors for showing the differences in text. The colors are set in Preferences under Preferences dialog used in TopTeam.

\Insert_Differences(<<First Text Field>>, <<Second Text Field>>)

Parameters

<<First Text Field>>	Mandatory	This is the Simple Text or Rich Text field. It cannot be a hard coded string. If the value is hard coded, an error message will display.
<<Second Text Field>>	Mandatory	This is a Simple Text or Rich Text field. It cannot be a hard coded string. If the value is hard coded, an error message will display.

Examples

\Fetch_Compare_Versions(a: Id, a: VersionId1, a: VersionId2)\
\scan(b)\
\if(b : Type = 'Text')\

\Insert_Differences(b: Value1, b: Value2)\

\endif\
\endscan\

Insert Diagram in Compare Versions

Use this command to Insert Diagrams while comparing versions.

Insert_Field_As_Diagram_In_CMs

Insert_Field_As_Diagram_In_Inches

Compatibility: Desktop App Version 4.5 and above.

This command inserts a diagram based on specified dimensions. These commands were specially designed to support the Fetch_Compare_Versions() command.

**\ Insert_Field_As_Diagram_In_CMs(<<Diagram Field>>,'<<Width>>', '<<Height>>', '<<Image Formats>>') **

\ Insert_Field_As_Diagram_In_Inches(<<Diagram Field>>, '<<Width>>', '<<Height>>', '<<Image Formats>>')

Parameters

<<Diagram Field>>	Mandatory	This is the field containing the in EMF format.
<<Width>>	Optional	Set the Width of the Diagram.
<<Height>>	Optional	Set the Height of the Diagram.
<<Image Formats>>	Optional	The image formats supported by this command are: • EMF • WMF • JPEG You need to specify one of the above values in this parameter. By default, the Diagram will be generated in EMF format.

Fields Available

There are no fields available for this command.

Examples

\Insert_Field_As_Diagram_In_CMs(b:Value1)\
\Insert_Field_As_Diagram_In_CMs(b:Value1,'4','2','EMF')\
\Insert_Field_As_Diagram_In_CMs(b:Value2,'2','6')\
\Insert_Field_As_Diagram_In_CMs(b:Value1,'10','15','JPEG')\

\Insert_Field_As_Diagram_In_Inches(b:Value2,'12','9','WMF')\
\Insert_Field_As_Diagram_In_Inches(b:Value1,'','','JPEG')\

Declare Variable

VAR

Compatibility: Desktop App Version 3.00 and above.

This command creates Variables which can be used in the template for any utility purpose. You can use these Variables in IF conditions or as parameters in Fetch commands.

This command is most often used along with the SET command. Refer to the following example to see how the commands are used.

**\ VAR(<<Variable Name>>) **

Parameter

Variable Name	Mandatory	Specify the Name of the Variable that you want to create.

Variable Naming Rules:

- The Variable Names can only have letters, numbers and the underscore character ('_').
- There must be no spaces in Variable Names.

Examples

```
\VAR(LastState) \
\ LastState:=''\
\scan(a)\
\if (LastState <> a : State)\
State:  \a : State \
\endif\
Transition: \ a : Transition\

\SET(LastState,  a : State)\
\endscan\
```

Declare_Variable

Compatibility: Desktop App Version 4.50 and above.

This command creates Variables for the Prompt_For_Variable_Values() command. You can use these to prompt the user to set Variable values at run-time. They can also be used as parameters in Fetch commands.

\Declare_Variable('<<Variable Name>>', '<<Data Type>>', '<<Usage Hint >>', '<<Default value>>')

Parameters

Variable Name	Mandatory	Specify the Name of the Variable that you want to create.
Data Type	Mandatory	Specify the Data Type that the Variable should hold.
Usage Hint	Optional	Specify a Hint for the Variable. This Hint will be displayed when a user is prompted for Variable values.
Default Value	Optional	Specify the default value of the Variable. This default value will be generated if the user does not set a value through Prompt_For_Variable_Values() command.

Variable Naming Rules:

- Variable Names can only have letters, numbers and the underscore character ('_').
- There must be no spaces in Variable Names.

Examples

\Declare_Variable('Record_ID', 'Text', 'Enter Requirement Document Id', 'RDOC-162')\
\Declare_Variable('Filter_Name', 'Text', 'Enter Filter Name to filter Requirement Document records', '')\
\Declare_Variable('WBS_Code', 'Float', 'Enter Wbs Code for Requirement Document records', '1')\
\Declare_Variable('Show_Parent', 'Flag', 'Check Show Parent option to display Header records OR Parent records in Requirement Document even if they do not satisfy filter condition and child record satisfies the filter condition', 'False')\
\Prompt_For_Variable_Values('Record_ID' , 'Filter_Name', 'WBS_Code', 'Show_Parent')\

Declare_Variable_Baseline()

Compatibility: Desktop App Version 5.x and above.

This command creates a Variable along with its value at runtime

The **Declare_Variable_Baseline** command is used to create Variables.
The **Prompt_For_Variable_Values** command is used to set values of Variables at runtime.

The value of the command parameter can be changed at runtime. Therefore, different outputs can be generated without editing the template.

\ Declare_Variable_Baseline('<<Variable Name>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

BASELINE selection

\Declare_Variable_Baseline('Baseline_Value1')\

BASELINE selection with Hint

\Declare_Variable_Baseline('Baseline_Value2', 'Select Baseline name ')\

BASELINE selection with Hint and default value

\Declare_Variable_Baseline('Baseline_Value2', 'Select Baseline name ', 'Beta Baseline')\

Example

```
\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_Baseline('Type_Value1', 'Select Baseline ' )\
\Prompt_For_Variable_Values('$ALL$')\
\VAR(LFilter_Variable) \
\ LFilter_Variable :=  Type_Value1 \
\Set_Baseline(LFilter_Variable)\
\Fetch_Use_Cases_By_Condition()\
\scan(a)\
\a: Name\ [\a:id\]
 \endscan\
```

Declare_Variable_Custom_Value()

Compatibility: Desktop App Version 5.x and above.

This command creates Variables along with its values at runtime.
Declare_Variable_Custom_Value: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

The value of the command parameter can be changed at runtime. Therefore, different outputs can be generated without editing the template.

\ Declare_Variable_Custom_Value('<<Variable Name>>', '<<Values List>>', '<<Is Multiple Selection>>', '<<Hint>>','<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable_Name>>	Mandatory	Specify the Name of the Variable that you want to create.
'<<Values List>>	Mandatory	Create a multiple List of Values separated by ';'.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single VALUE selection

\Declare_Variable_Custom_Value('Type_Value3','a;b;c;d')\

Multiple VALUE selection for Requirement Record types with Hint

\Declare_Variable_Custom_Value('Type_Value2','a;b;c;d', 'True', 'Select Multiple values ')

Multiple VALUE selection for Requirement Record types with Hint and default value

\Declare_Variable_Custom_Value('Type_Value2','a;b;c;d', 'True', 'Set Multiple values ', 'b;c')

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Declare_Variable_Custom_Value('Type_Value1', 'multi lst val1;multi lst val2;multi lst val3', False, 'Set Single value', ' "multi lst val1" ')\

\Prompt_For_Variable_Values('ALL')\
\VAR(LFilter_Variable) \
\ LFilter_Variable := ' "Multi value field " in list ' + Type_Value1 \
\Fetch_Use_Cases_By_Condition(LFilter_Variable)\
\scan(a)\

\a : name\ [\a:id\]
\a: Multi value field\

\endscan\

Declare_Variable_Filter()

Compatibility: Desktop App Version 5.x and above.

This command creates Variables along with its values at runtime.
Declare_Variable_Filter: Command is used to create Variables.
Prompt_For_Variable_Values: Command is used to set values of Variables at runtime.

It allows the user to change the values of a command parameter at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Filter('<<Variable Name>>', '<<ID Prefix>>', '<<Is Tree Filter>>', '<<Hint>>','<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<ID Prefix>>	Mandatory	Select the Filter for the specified Record Type.
<<Is Tree Filter>>	Optional	Grant permissions to select List or Tree Filter. Set **FALSE** for List Filter Selection. Set **TRUE** for Tree filter Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the variable for the Optional Value selection. Set **TRUE** to display '*' in front of the variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

FILTER selection for Use Case

\Declare_Variable_Filter('Filter_Value', 'UC')\

FILTER selection for Requirement Tree

\Declare_Variable_Filter('Filter_Value', 'REQ','True')\

FILTER selection for Use Case with Hint

\Declare_Variable_Filter('Filter_Value', 'UC','', ' Set Filter')\

FILTER selection for Use Case with Hint and default value

\Declare_Variable_Filter('Filter_Value', 'UC','', ' Set Filter ', 'Completed Use Cases')\

\Declare_Variable_Filter('Filter_Value','UC', '', 'Set Filter for Use Case')\
\Declare_Variable_Filter('Filter_Value1','MTG', '', 'Set Filter for Meeting')\
\Declare_Variable_Filter('Filter_Value2','MOD', '', 'Set Filter for Modules')\
\Declare_Variable_Filter('Filter_List_Req','REQ', '','Set Filter for Requirements List ')\
\Declare_Variable_Filter('Filter_Tree_Req','REQ','True','Set Filter for Requirements Tree ')\
\Declare_Variable_Filter('Filter_List_Task','TSK', '', 'Set Filter for Task list')\
\Declare_Variable_Filter('Filter_Tree_Task','TSK', 'True', 'Set Filter for Task Tree')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_Filter('Filter_Value', 'UC')\
\Declare_Variable_Filter('Filter_Tree_Req','REQ','True','Set Filter for Requirements Tree ')\
\Prompt_For_Variable_Values('ALL')\
\VAR(LVariable_Filter) \

Examples

\ LVariable_Filter := Filter_Value \
\Fetch_Use_Cases(LVariable_Filter)\
\scan(a) \

\a:Name\ [\a:id\]

\endscan\

Examples

\ LFilter_Variable := Filter_Tree_Req \

\Fetch_Requirements_Tree_By_Document_Id('$DEFAULT_REQUIREMENTS_DOCUMENT$', LFilter_Variable)\

\scan(a) \
\if (! eof(a))\
\if (a : Indentation Level = '1')\

**\a: wbs ** [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '1')\

\elsif (a : Indentation Level = '2')\

**\a: wbs ** [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '2')\

\elsif (a : Indentation Level = '3')\

**\a: wbs ** [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '3')\

\else\

**\a: wbs ** [\ a : Id \ ] - \ a : Title \

\ Insert_Indented_Rtf(a : Description, '4')\

\endif\
\endif\
\endscan\

Declare_Variable_Folder()

Compatibility: Desktop App Version 5.x and above.

This command creates Variables along with its values at runtime

Declare_Variable_Folder: This command is used to create Variables.

Prompt_For_Variable_Values: This command is used to set values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Folder('<<Variable Name>>', '<<ID Prefix>>','<<Hint>>', '<<Default_Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<< ID Prefix>>	Mandatory	Select the Filter for the specified Record Type.
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

PROJECT selection

\Declare_Variable_Folder('Folder_Path', 'UC')\

PROJECT selection with Hint

\Declare_Variable_Folder('Folder_Path', 'UC', 'Select Folder Path ')\

PROJECT selection with Hint and default value

\Declare_Variable_Folder('Folder_Path', 'UC', 'Select Folder Path', 'Video Rental system\Rental Management')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Declare_Variable_Folder('Folder_Path', 'UC', 'Select Folder Path', 'Video Rental system\Rental Management')\

\Prompt_For_Variable_Values('ALL')\
\Fetch_Records_By_Folder_By_Condition('UC', Folder_Path)\
\scan(a) \
\if (! eof(a))\
\if(a : Type= 'Folder')\

\a: Folder Path

\else\

ID-\a:ID\ \a:Name\

\endif\
\endif\
\endscan\

Declare_Variable_ID()

Compatibility: Desktop App Version 5.x and above.

This command creates variables along with its values at runtime.

Declare_Variable_ID: Command is used to create Variables.

Prompt_For_Variable_Values: This command is used to set values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_ID('<<Variable_Name>>', '<<ID Prefix>>', '<<Is Multiple Selection>>', '<<Hint>>','<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<ID Prefix>>	Optional	Select the Filter for the specified Record Type.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single ID selection for All Record types

\Declare_Variable_ID('ID_Value')\

Single ID selection from Use Case Record type

\Declare_Variable_ID('ID_Value', 'UC')\

Multiple ID selection from Use Case Record types

\Declare_Variable_ID('ID_Value', "UC ", True)\

Multiple ID selection from Use Case Record type with Hint

\Declare_Variable_ID('ID_Value', "UC " , True, 'Set ID', '')\

Multiple ID selection from Use Case Record types with Hint and default value

\Declare_Variable_ID('ID_Value', 'UC' , True, 'Set ID', '1532')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_ID('ID1_Value', 'UC', False, 'Set ID', '4638')\
\Declare_Variable_ID('ID2_Value', '' , True)\
\Prompt_For_Variable_Values('ALL')\

Examples

\VAR(LVariable_Filter) \
\ LVariable_Filter:= ' "ID" = ' + ID1_Value \
\Fetch_Use_Cases_By_Condition (LVariable_Filter)\
\scan(a) \

\a:ID \a:Name\

\endscan\

Examples

\ LVariable_Filter := ' "ID" in List ' + ID2_Value \
\Fetch_Use_Cases_By_Condition (LVariable_Filter)\
\scan(a) \

\a:ID \a:Name\

\endscan\

Declare_Variable_LOV()

Compatibility: Desktop App Version 5.x and above.

This command creates Variables along with its values at runtime.
Declare_Variable_LOV: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_LOV('<<Variable Name>>', '<<ID Prefix>>', '<<LOV Field Name>>', '<<Is Multiple Selection>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<ID Prefix>>	Mandatory	Select the Filter for the specified Record Type. The LOV Field Name parameter should be a field in this Record Type.
<<LOV Field Name>>	Mandatory	Specify any LOV Field Name included in the Record Type Record ID Prefix.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection.

		Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

This includes any List Type fields included in the Record Type ID Prefix such as Priority, Complexity, Functional Area, Size, etc.

Field	Description
Priority	
Complexity	
Functional	
Area	
Size	

Examples

Set Single PRIORITY value selection for Use Case

\Declare_Variable_LOV('Priority_Value', 'UC', 'Priority')\

Set Multiple PRIORITY value selection for Use Case

\Declare_Variable_LOV('Priority_Value', 'UC', 'Priority','True')\

Set Multiple PRIORITY value selection for Use Case with Hint

\Declare_Variable_LOV('Priority_Value', 'UC', 'Priority','True', 'Set Priority – Multiple Value selection ')\

Set Multiple PRIORITY value selection for Use Case with Hint and default value

\Declare_Variable_LOV('Priority_Value', 'UC', 'Priority','True', 'Set Priority – Multiple Value selection ', 'High')\

Set Single SIZE value selection for Use Case with Hint and default value

\Declare_Variable_LOV('Size_Value', 'UC', 'Size', 'False', 'Set Size – Single Value selection', 'Small')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Declare_Variable_LOV('Priority_Value', 'UC', 'Priority','True', 'Set Priority – Multiple Value selection ', 'High')\
\Declare_Variable_LOV('Size_Value', 'UC', 'Size','False','Set Size – Single Value selection','Small')\

\Prompt_For_Variable_Values('ALL')\

Examples

\VAR(LVariable_Filter) \
\ LVariable_Filter := ' Size = ' + Size_Value \
\Fetch_Use_Cases_By_Condition(LVariable_Filter)\
\scan(a) \

\a: Name\ [\a:id\]

\endscan\

Examples

\ LVariable_Filter := ' Priority in list ' + Priority_Value \
\Fetch_Use_Cases_By_Condition(LVariable_Filter)\
\scan(a) \

\a: Name\ [\a:id\]

\endscan\

Declare_Variable_Name()

Compatibility: Desktop App Version 5.x and above.

This command creates Variables along with its values at run-time.
Declare_Variable_Name: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Name('<<Variable Name>>', '<<ID Prefix>>', '<<Is Multiple Selection>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable_Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<ID Prefix>>	Optional	Select the Filter for the specified Record Type.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection.

		Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single Name selection for All Record types

\Declare_Variable_Name('Name_Value')\

Single Name selection from Use Case Record type

\Declare_Variable_Name('Name_Value', 'UC')\

Single Name selection from Use Case or Online Document Record types

\Declare_Variable_Name('Name_Value', "UC")\

Single ID selection from Use Case Record types with Hint and default value

\Declare_Variable_Name('Name_Value', 'UC' , False, 'Set Name', 'Record Sharing')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_Name('Name1_Value', 'UC' , False, 'Set Name', 'Record Sharing')\
\Declare_Variable_Name('Name2_Value', ' ', True, 'Set Name', '')\
\Prompt_For_Variable_Values('ALL')\

Examples

\VAR(LVariable_Filter) \
\ LVariable_Filter := ' Name = ' + Name1_Value \
\Fetch_Use_Cases_By_Condition (LVariable_Filter)\
\scan(a) \

\a:Name\ [\a:ID\]

\endscan\

Examples

\LVariable_Filter := ' Name **containing** ' + Name2_Value \
\Fetch_Use_Cases_By_Condition (LVariable_Filter)\
\scan(a) \

\a:Name\ [\a:ID\]

\endscan\

Declare_Variable_Package_Baseline()

Compatibility: Desktop App Version 5.x and above.
This command creates Variables along with its values at runtime.

Declare_Variable_Package_Baseline: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore generate different outputs without editing the template.

\ Declare_Variable_Package_Baseline('<<Variable Name>>', '<<Package Record Id>>','<<Hint option>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Package Record ID>>	Mandatory	Specify the Package ID for which you want to fetch the Baselines.
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.

<<Is Mandatory>>	Optional	Display '*' in front of the variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

BASELINE selection

\Declare_Variable_Package_Baseline('Baseline_Name')\

BASELINE selection For Package

\Declare_Variable_Package_Baseline('Baseline_Name', 'WPKG-3075')\

BASELINE selection with Hint

\Declare_Variable_Package_Baseline('Baseline_Name', 'WPKG-3075' , 'Select Baseline name')\

BASELINE selection with Hint and default value

\Declare_Variable_Package_Baseline('Baseline_Name', 'WPKG-3075','Select Baseline name', 'Beta Baseline')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_Package_Baseline('Baseline_Name', 'WPKG-3075')\
\Prompt_For_Variable_Values('Baseline_Name')\

Package Contents In Baseline: \Baseline_Name\

\scan(a) \

Package: [\ a : ID\] \a : Name\

\Fetch_Package_Contents(a:ID, Baseline_Name)\
\scan(b) \
\if (! eof(b))\
\if(b : Record Type= 'Folder')\

**\b: wbs\ \b : Name **

\elsif(b : Record Type= 'Collection Folder')\

**\b: wbs\ \b : Name **

\else\

**\b: wbs\ \b : Name **

\Execute_Template_For_Id(b:ID, b:Version, '')\
\endif\\endif\\endscan\
\endscan\

Declare_Variable_Project()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its value at runtime.
Declare_Variable_Project: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Project('<<Variable Name>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.

<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

PROJECT selection

\Declare_Variable_Project('Project_Value1')\

PROJECT selection with Hint

\Declare_Variable_Project('Project_Value2', 'Select Project name ')\

PROJECT selection with Hint and default value

\Declare_Variable_Project('Project_Value2', 'Select Project name ', 'Project1')\

Examples

```
\Declare_Variable_Project('Type_Value1', 'Select Project' )\
\Declare_Variable_Project('Type_Value2', 'Select Project' )\
\Prompt_For_Variable_Values('$ALL$')\
\VAR(LFilter_Variable) \
\ LFilter_Variable :=  Type_Value1 \
\Set_Project(LFilter_Variable)\
```

\Fetch_Use_Cases_By_Condition()\
\scan(a)\

\a: Name\ [\a:id\]

\endscan\

\ LFilter_Variable := Type_Value2 \
\Set_Project(LFilter_Variable)\
\Fetch_Use_Cases_By_Condition()\
\scan(a)\

\a: Name\ [\a:id\]

\endscan\

Declare_Variable_Record_Type()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its values at runtime.
Declare_Variable_Record_Type: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Record_Type('<<Variable Name>>', '<<Is Multiple Selection >>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<< Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection.

		Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single RECORD TYPE selection for current Project

\Declare_Variable_ Record_Type('Type_Value1')\

Multiple RECORD TYPE selection for current Project

\Declare_Variable_ Record_Type('Type_Value2', True)\

Multiple RECORD TYPE selection for current Project with Hint

\Declare_Variable_ Record_Type('Type_Value2', True, 'Select Multiple Record types')\

Multiple RECORD TYPE selection for current Project with Hint and default value

\Declare_Variable_ Record_Type('Type_Value2', True, 'Select Multiple Record types', 'Business Rules')\

Example

\Set_Project('$CURRENT_PROJECT$')\

Examples

\Declare_Variable_ Record_Type('Type_Value1', False, 'Select Single Record type', '')\
\Declare_Variable_ Record_Type('Type_Value2', True, 'Select Multiple Record types', '')\
\Prompt_For_Variable_Values('ALL')\
\VAR(LFilter_Variable) \
\ LFilter_Variable := ' "Type" = ' + Type_Value1 \
\Fetch_Requirements_By_Condition (LFilter_Variable,'id')\
\scan(a) \

\a:Title\ [\a:ID\]

\endscan\

Examples

\ LFilter_Variable := ' Type in List ' + Type_Value2 \
\Fetch_Requirements_By_Condition (LFilter_Variable,'id')\
\scan(a) \

\a:Title\ [\a:ID\]

\endscan\

Declare_Variable_Record_Type_Prefix()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its values at runtime.
Declare_Variable_Record_Type_Prefix: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Record_Type_Prefix('<<Variable name>>', '<<Is Multiple Selection >>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single RECORD TYPE selection for current Project

\Declare_Variable_Record_Type_Prefix('Type_Value1')\

Multiple RECORD TYPE selection for current Project

\Declare_Variable_Record_Type_Prefix('Type_Value2', True)\

Single RECORD TYPE selection for current Project with Hint

\Declare_Variable_Record_Type_Prefix('Type_Value1', '' , 'Select Single Record type Prefix')\

Multiple RECORD TYPE selection for current Project with Hint

\Declare_Variable_Record_Type_Prefix('Type_Value2', True, 'Select Multiple Record type Prefix')\

Multiple RECORD TYPE selection for current Project with Hint and default value

\Declare_Variable_Record_Type_Prefix('Type_Value2', True, 'Select Multiple Record type Prefix', 'Business Rules')\

Example

\Set_Project('$CURRENT_PROJECT$')\

Examples

\Declare_Variable_ Record_Type_Prefix('Type_Value1', False, 'Select Single Record type Prefix ', '')\
\Declare_Variable_ Record_Type_Prefix('Type_Value2', True, 'Select Multiple Record type Prefix', '')\
\Prompt_For_Variable_Values('ALL')\
\VAR(LFilter_Variable) \
\ LFilter_Variable := Type_Value1 \
\Fetch_Repository_Objects_By_Condition(LFilter_Variable)\
\scan(a)\

\a:Name\ [\ a : id\]

\endscan\

Examples

\ LFilter_Variable := Type_Value2 \
\Fetch_Requirements_By_Condition(' "Type" = "Features" ','Title')\
\scan(a) \

\a:Title\ [\ a : id\]

\Fetch_Traced_Records_Of_Type_By_Condition(LFilter_Variable, '' , '')\

\scan(b)\

\ b:Name\ [\ b : Id \]

\endscan\

\endscan\

Declare_Variable_Release()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its values at runtime.
Declare_Variable_Release: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_Release('<<Variable Name>>', '<<Is Multiple Selection>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.

<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single RELEASE selection

\Declare_Variable_Release('Release_Value1')\

MULTIPLE RELEASE selection

\Declare_Variable_Release('Release_Value1', True)\

Single RELEASE selection with Hint

\Declare_Variable_Release('Release_Value2','', 'Select Release name ')\

Single RELEASE selection with Hint and default value

\Declare_Variable_Release('Release_Value2','', 'Select Release name ', 'Beta Release')\

Examples

```
\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_Release('Type_Value1', '' , 'Select Release ' )\
\Prompt_For_Variable_Values('$ALL$')\
\VAR(LFilter_Variable) \
\ LFilter_Variable :=  Type_Value1 \
```

\Set_Release(LFilter_Variable)\
\Fetch_Use_Cases_By_Condition()\
\scan(a)\

\a: Name\ [\a:id\]

\endscan\

Declare_Variable_State()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its values at runtime.
Declare_Variable_State: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.
\ Declare_Variable_State('<<Variable Name>>', '<<Selected Record Disp Prefix>>', '<<Is Multiple Selection>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Selected Record Disp Prefix>>	Mandatory	Specify the Prefix for the Record Type you wish to display.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**

<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

There are no fields available for this command.

Examples

Single STATE selection for Requirement Record type

\Declare_Variable_State('State_Value1','REQ',)\
\Declare_Variable_State('State_Value1','REQ', '', 'Select single State for Requirements ')\

Single STATE selection for Use Case Record type

\Declare_Variable_State('State_Value3','UC')\
\Declare_Variable_State('State_Value3','UC', '',Select State for Use Case')\

Multiple STATE selection for Requirement Record types with Hint

\Declare_Variable_State('State_Value2','REQ', 'True', 'Select Multiple States for Requirements ')\

Multiple STATE selection for Requirement Record types with Hint and default value

\Declare_Variable_State('State_Value2','REQ', 'True', 'Set Multiple States for Requirements ', 'Completed')\

Examples

```
\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable_State('State_Value1','REQ',)\
\Prompt_For_Variable_Values('$ALL$')\
\VAR(LFilter_Variable) \
\ LFilter_Variable := ' "State" = ' + State_Value1 \
\Fetch_Requirements_By_Condition (LFilter_Variable,'id')\
\scan(a) \

\a:Title\ [\a:ID\]

\endscan\
```

Declare_Variable_User()

Compatibility: Desktop App Version 5.x and above.

This command creates the Variables along with its values at runtime.
Declare_Variable_User: This command is used to create Variables.
Prompt_For_Variable_Values: This command is used to set the values of Variables at runtime.

It allows the user to change the values of the command parameters at runtime and therefore, generate different outputs without editing the template.

\ Declare_Variable_User('<<Variable Name>>', '<<Is Multiple Selection>>','<<Is All Project Users>>', '<<Hint>>', '<<Default Value>>', '<<Is Mandatory>>')

Parameters

<<Variable Name>>	Mandatory	Specify the Name of the Variable that you want to create.
<<Is Multiple Selection>>	Optional	Grant permissions to select Single or Multiple Values. Set **FALSE** for Single Value Selection. Set **TRUE** for Multiple Values Selection. By default, it is **FALSE.**

<<Is All Project Users>>	Optional	Grant permissions to select Current or All Project Users. Set **FALSE** for Current Project User. Set **TRUE** for ALL Project Users. By default, it is **FALSE.**
<<Hint>>	Optional	Specify the Hint for the Variable.
<<Default Value>>	Optional	Specify the default value of the Variable.
<<Is Mandatory>>	Optional	Display '*' in front of the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' in front of the Variable for the Optional Value selection. Set **TRUE** to display '*' in front of the Variable for the Mandatory Value selection. By default, it is **TRUE.**

Fields Available

Field	Description
Owner	
Assigned To	
Crt By	
Upd By	

Examples

Set Single User from Project Team Members

\Declare_Variable_User('User1_Value')\

Set Single User from Project Team Members with Hint

\Declare_Variable_User('User1_Value', '', '', 'Set Single User from Project Team Members', '')\

Set Single User from Project Team Members with Hint and default value

\Declare_Variable_User('User1_Value', '', '', 'Set Single User from Project Team Members', 'Steve')\

Set Multiple Users from Project Team Members

\Declare_Variable_User('User3_Value', 'True')\

Set Single User from ALL Project Team Members

\Declare_Variable_User('User1_Value', 'False', 'True')\

Set Multiple Users from ALL Project Team Members

\Declare_Variable_User('User3_Value', 'True', 'True')\

Set Multiple Users from Project Team Members with Hint

\Declare_Variable_User('User2_Value','True',False,'Set Multiple Users from Project Team Members')\

Set Multiple Users from ALL Project Team Members with Hint and default value

\Declare_Variable_User('User3_Value', 'True', True, 'Set Multiple Users from all Project Users', 'User1_SA')\

Set Single User from ALL Project Team Members with Hint and default value

\Declare_Variable_User('User4_Value', '', True, 'Set Single User from all Project Users', 'User1_SA')\

Examples

```
\Set_Project('$CURRENT_PROJECT$')\
\VAR(LVariable_Filter) \
\Declare_Variable_ User('User1_Value', '', False, 'Set Single User from Project Team Members', '')\
\Prompt_For_Variable_Values('$ALL$')\
\ LVariable_Filter :=  ' "Owner"  =  '  + User1_Value \
\Fetch_Use_Cases_By_Condition( LVariable_Filter)\
\scan(a) \

\a:Name\ [\a:Id\]

\endscan\
```

Insert_Custom_Variable

Compatibility: Desktop App Version 8.15 and above.

This command inserts a Custom Variable.
This UDF will give the first preference to Project Variables if Project details are specified.
If Project details are not mentioned, it will display Global Variables without raising any exceptions.

This command can be used independently at any place in the template.

\Insert_Custom_Variable('<<Custom Variable>>', '<<Project Details>>')

Parameters

<<Custom Variable>>	Mandatory	This is a mandatory parameter. User can specify Custom Variables defined in Global as well as Project context. This command is not Project dependent. Specify Custom Variable Names by which you want the results to be displayed. If not specified, the command will not work.
<<Project Details>>	Optional	Specify Project details here. Project details can be ID With or Without Prefix, Project Name, Project Path, $CURRENT_PROJECT$. If Project details are not specified, Global values of Variables will be printed.

Fields Available

No fields are available in this command.

Examples

(Without Project Details)

- \Insert_Custom_Variable('ID')\
- \Insert_Custom_Variable('Company Name')\
- \Insert_Custom_Variable('Company Logo')\

(With Project Details)

- \Insert_Custom_Variable('ID','$CURRENT_PROJECT$')\
- \Insert_Custom_Variable('Company Name','New Project')\
- \Insert_Custom_Variable('ID','Online Video Rental System\ApplicationServer')\
- \Insert_Custom_Variable('ID','PRJ-101')\

Examples

```
\Set_Project('$CURRENT_PROJECT$')\
                                        \ PROJECT_NAME \

  \Insert_Custom_Variable('Company Name')\
   \Insert_Custom_Variable('Company Logo', '$CURRENT_PROJECT$')\
```

Prompt_For_Variable_Values()

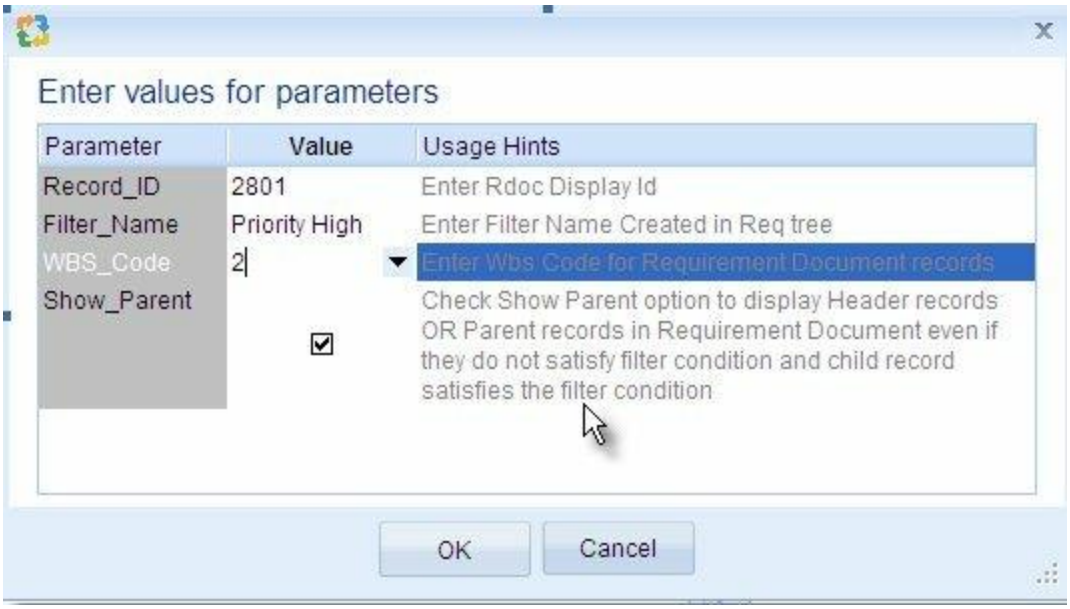
Compatibility: Desktop App Version 5.x and above.

This miscellaneous command prompts the user to enter values for Variables at runtime. It will be used before executing a Fetch command in the template. Fetch commands can be either a Master or Sub-report command.

The Variables are created/declared using the Declare Variable() command. With the help of this command users can do the following:

1. Set valuse of particular Variables at runtime.
2. Pass Variables to Fetch commands as parameters at runtime.

Prompt_For_Variable_Values(): This command displays the following dialog:



In the above dialog, the Record_ID is the declared parameter passed to this command. The Record_ID is an Integer. You need to set the values for the other parameters within this command, depending upon its data type.

You can specify on of the following types of parameters:

- Integer *or* Number
- Boolean *or* Flag
- Float *or* Decimal
- Text (By Default)
- LOV
- User
- ID
- Name
- State
- Type
- Filter

Any other parameters will be considered as text parameters, by default.

Once specified, these Variables along with their values are passed to the Fetch command and used in the template.

Prompt Dialog for ALL Variables

\Prompt_For_Variable_Values('ALL')

Parameter

<<ALL>>	Optional	This prompts the Dialog for setting the values of the all Variables created using the Declare Variable() command at runtime.

This is most widely used command for prompting all parameters.

OR

Prompt dialog for SELECTED variables

```
\Prompt_For_Variable_Values (   <<First Declared Variable>>,
                                <<Second Declared Variable>>,
                                <<Third Declared Variable>>
                                -
                                -
                                -
                                -
                                <<Nth Declared Variable>>) \
```

This command is rarely used.

Parameters

<<First Declared Variable>>	Mandatory	This is the first Declared Variable passed to this command. This command must contain at least one parameter.
<<Second Declared Variable>>	Optional	If two parameters are required, you may use this second Declared Variable.
.<<Nth Declared	Optional	If more than two parameters are required, you may use this nth Declared Variable.

Variable> >		

Fields Available

There are no fields available for this command

Examples

\Prompt_For_Variable_Values('ALL')\
\Prompt_For_Variable_Values('Record_ID' , 'Filter_Name', 'WBS_Code', 'Show_Parent')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable('Record_ID', 'Number', 'Enter Rdoc Display Id', '1382')\
\Prompt_For_Variable_Values('Record_ID')\
\Fetch_Requirements_Tree_By_Document_Id(Record_ID)\
\scan(a) \

**\a: wbs ** \ a : Id \ \ a : Title \

\endscan\

Examples

\Declare_Variable('Record_ID', 'Number', 'Enter Rdoc Display Id', '1382')\
\Declare_Variable('Filter_Name', 'Text', 'Enter Filter Name Created in Req tree', 'priority high')\
\Declare_Variable('WBS_Code', 'decimal', 'Enter Wbs Code for Requirement Document records', '1')\
\Declare_Variable('Show_Parent', 'flag', 'Check Show Parent option to display Header records OR Parent records in Requirement Document even if they do not satisfy filter condition and child record satisfies the filter condition', 'true')\
\Prompt_For_Variable_Values('ALL')\

\Fetch_Requirements_Tree_By_Document_Id(Record_ID, Filter_Name, WBS_Code, Show_Parent)\
\scan(a) \

**\a: wbs ** \ a : Id \ \ a : Title \

\endscan\

Review Packages Commands

Fetch_Package_Contents()

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Package Contents in a Baseline for the specified Package ID.

\Fetch_Package_Contents(<<Package Record ID>>,<<Package Baseline Name>>)

Parameters

<<Package Record ID>>	Mandatory	Specify the Display ID of the Package.
<<Package Baseline Name>>	Optional	Specify the Package Baseline Name to fetch the Package Contents for a specific Baseline. By default, the current Package Contents will be utilized if Baseline is not specified.

Fields Available

Field	Description
ID	
Name	
Is Pinned To Version	
Version	
Status	
Project	
Comments Count	

Record Type	
Indentation Level	

Examples

\Fetch_Package_Contents('PKG-6819')\

\Fetch_Package_Contents('6819')\

\Fetch_Package_Contents('6819', 'Alpha Package Baseline')\

\Fetch_Package_Contents('PKG-6819', 'Baseline Created after My Approval')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Declare_Variable('Package_ID', 'String')\

\Declare_Variable('Baseline_Name', 'String')\

\Prompt_For_Variable_Values('Package_ID', 'Baseline_Name')\

\Fetch_Package_Contents(Package_ID, Baseline_Name)\

\scan(a) \

\if (! eof(a))\

\if (a : Indentation Level = 0)\

[\ a : Id \ ] - \ a : Name \

\ a : Is Pinned To Version \-\ a : Version \-\ a : Comments Count\

\elsif (a : Indentation Level = 1)\

[\ a : Id \ ] - \ a : Name \

\ a : Is Pinned To Version \-\ a : Version \-\ a : Comments Count\

\elsif (a : Indentation Level = 2)\

[\ a : Id \ ] - \ a : Name \

\ a: Is Pinned To Version \-\ a : Version \-\ a : Comments Count\

\else\

[\ a : Id \ ] - \ a : Name \

\ a : Is Pinned To Version \-\ a : Version \-\ a : Comments Count\

\endif\

\endif\
\endscan\

Fetch_Approvals_For_Repository_Object_By_Condition()

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Approvals created for the current Master record.

\Fetch_Approvals_For_Repository_Object_By_Condition('<<Filter Condition>>', '<<Sort Order>>')

Parameters

<<Filter Condition>>	Optional	Specify the filter condition as per the required values of the Record. If this is not specified, all Records will be fetched.
<<Sort by Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is sorted by default on Create Date.

Fields Available

All Fields of Record Type Approval.

Examples

\Fetch_Approvals_For_Repository_Object_By_Condition()\
\Fetch_Approvals_For_Repository_Object_By_Condition(' "Priority" = "High" ')\
\Fetch_Approvals_For_Repository_Object_By_Condition('','Type, Id')\.
\Fetch_Approvals_For_Repository_Object_By_Condition(' "Priority" = "High" ','Type, Id')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \

\a: Name\ \ a: Id\

\Fetch_Approvals_for_Repository_Object_By_Condition (‘’, ‘’)\
\if (! eof(b))\

Approvals

Title	Approver	Upd dt

\scan(b)\
\if (! eof(b))\

\ b : Name \	\ b : Approver \	\ b : Upd dt \

\endif\
\endscan\
\endif\
\endscan\

Fetch_Compare_Package_Baselines()

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Package Contents modified after a specified Baseline for the specified Package ID.

\Fetch_Compare_Package_Baselines(<<Package Record ID>>,<<Baseline Name1>>,<<Baseline Name2>>)

Parameters

<<Package Record ID>>	Mandatory	Specify the Display ID of the Package.
<<Baseline Name1>>	Mandatory	Specify the Package Baseline_Name to fetch Package Contents Modified After specified Date.
<<Baseline Name2>>	Optional	Specify the Package Baseline_Name2 to compare Package Contents between Baseline_Name1 and Baseline_Name2.

Fields Available

Field	Description
ID	
Name	
Is Pinned To Version	
Version 1	
Version 2	
Project	
Record Type	
Comparison Status	
Comments Count	
Indentation Level	

Examples

\Fetch_Compare_Package_Baselines('WPKG-6819','Beta Baseline')\
\Fetch_Compare_Package_Baselines('6819', 'Baseline Created after My Approval')\
\Fetch_Compare_Package_Baselines('6819', 'Alpha Package Baseline','Beta Baseline')\

Example

\Set_Project('$CURRENT_PROJECT$')\

Examples

\Declare_Variable('Package_ID', 'String')\
\Declare_Variable('Baseline Name', 'String')\
\Prompt_For_Variable_Values('Package_ID', 'Baseline_Name')\
\Fetch_Compare_Package_Baselines(Package_ID, Baseline_Name)\
\scan(a)\\if (Bof(a))\

Record	Comments	Version

\endif\

\if(a : status = ‘Added’)\

**\ a : Id ** \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a: status = ‘Removed’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a : status = ‘Modified’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\endscan\

Examples

\Declare_Variable(‘Record_ID’, ‘String’)\

\Declare_Variable(‘Baseline_Name1’, ‘String’)\

\Declare_Variable(‘Baseline_Name2’, ‘String’)\

\Prompt_For_Variable_Values(‘Record_ID’, ‘Baseline_Name1’, ‘Baseline_Name2’)\

\Fetch_Compare_Package_Baselines (Record_ID, Baseline_Name1, Baseline_Name2)\

\scan(a)\\if (Bof(a))\

Record	Comments	Version

\endif\

\if(a : status = 'Added')\

**\ a : Id ** \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a: status = 'Removed')\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a : status = 'Modified')\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\endscan\

Fetch_Package_Baselines

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Baselines created for the specified Package ID.

\Fetch_Package_Baselines(<<Package Record ID>>)

Parameter

<<Package Record ID>>	Mandatory	Specify the Package ID.

Fields Available

Field	Description

Baseline	
Crt dt	
Crt by	

Examples

\Fetch_Package_Baselines('WPKG-6819')\
\Fetch_Package_Baselines('6819')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable('Package_ID', 'String')\
\Prompt_For_Variable_Values('Package_ID')\
\Fetch_Package_Baselines(Package_ID)\
\if (! eof(a))\

Baseline	Crt Dt	Crt by

\endif\
\scan(a)\

\a:Baseline \	\a:Crt dt\	\a:Crt by\

\endscan\

Fetch_Review_Comments_By_Condition()

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Review Comments By Condition for the specified Package.

\Fetch_Review_Comments_By_Condition('<<Filter_Condition>>', '<<Sort_Order>>')

Parameters

<<Filter Condition>>	Optional	Specify a filter condition based on the required values of the Record. If nothing is specified, all Records will be fetched.
<<Sort by Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If nothing is specified, the data is sorted by Create Date.

Fields Available

Field	Description
Comment	
Date	
Person	
Type	
Old State	
New State	
Old Asgnd to	
New Asgnd to	
Upd by	
Upd dt	

Examples

\Fetch_Review_Comments_By_Condition()\

\Fetch_Review_Comments_By_Condition(' "Person" = "Me" ')\

\Fetch_Review_Comments_By_Condition('','Type, Id')\

\Fetch_Review_Comments_By_Condition(' "Person" = "Me" ','Type, Id')\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Declare_Variable('Package_ID', 'String')\

\Prompt_For_Variable_Values('Package_ID')\
\Fetch_Package_Contents(Package_ID)\
\scan(a) \
\if (! eof(a))\

[\ a : Id \ ] \ a : Name \

\Fetch_Review_Comments_By_Condition('','Date')\
\if (! eof(b))\

Review Comments

Date	Person	Comment

\scan(b)\
\if (! eof(b))\

\ b: Date \	\ b : Person \	\ b : Comment \

\endif\
\endscan\
\endif\
\endif\
\endscan\

Fetch_Package_Contents_With_Comments_Modified_After_Date()

Compatibility: Desktop App Version 6.20 and above.

This command fetches Package Comments Modified After the specified Date for the specified Package ID.

\Fetch_Package_Contents_With_Comments_Modified_After_Date(<<Package Record ID>>,<<Modified After Date>>)

Parameters

<<Package Record ID>>	Mandatory	Specify the Display ID of the Package.

<<Modified After Date>>	Mandatory	Specify the Date in the format mm-dd-yyyy e.g. 12-31-2005 to fetch Package Contents Modified After specified Date.

Fields Available

Field	Description
ID	
Name	
Is Pinned To Version	
Version	
Project	
Comments Count	
Record Type	
Indentation Level	

Examples

\Fetch_Package_Contents_With_Comments_Modified_After_Date('WPKG-6819','12-31-2005')\
\Fetch_Package_Contents_With_Comments_Modified_After_Date('6819','1-15-2005')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable('Package_ID', 'String')\
\Declare_Variable('Modified_After_Date', 'String')\
\Prompt_For_Variable_Values('Package_ID', 'Modified_After_Date')\

\ Hyphen "-" delimited date format "mm-dd-yyyy" e.g.. 12-31-2005

\Fetch_Package_Contents_With_Comments_Modified_After_Date(Package_ID, Modified_After_Date)\
\scan(a)\\if (Bof(a))\

Record	Comments	Version

\endif\

\if(a : status = 'Added')\

**\ a : Id ** \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a: status = 'Removed')\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a : status = 'Modified')\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition('','Date')\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\endscan\

Fetch_Package_Contents_Modified_After_Approval()

Compatibility: Desktop App Version 6.20 and above.

This command fetches Package Contents Modified After Approval for the specified Package ID.

\Fetch_Package_Contents_Modified_After_Approval(<<Approval Record ID>>)

Parameter

<<Approval Record ID>>	Mandatory	Specify the Approval Display ID.

Fields Available

Field	Description
ID	
Name	
Is Pinned To Version	
Version	
Project	
Comparison Status	
Record Type	
Comments Count	
Indentation Level	

Examples

\Fetch_Package_Contents_Modified_After_Approval('APVR-6819')\
\Fetch_Package_Contents_Modified_After_Approval('6819')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable('Approval_Record_ID', 'String')\
\Prompt_For_Variable_Values('Approval_Record_ID')\
\Fetch_Package_Contents_Modified_After_Approval('3247')\
\scan(a)\\if (Bof(a))\

Record	Comments	Version

\endif\
\if(a : status = 'Added')\

**\ a : Id ** \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\if(a: status = ‘Removed’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\if(a : status = ‘Modified’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\
\endscan\

Fetch_Package_Contents_Modified_After_Date()

Compatibility: Desktop App Version 6.20 and above.

This command is used to fetch Package Contents Modified After specified Date for the specified Package ID.

\Fetch_Package_Contents_Modified_After_Date(<<Package ID>>,<<Modified After Date>>)

Parameters

<<Package ID>>	Mandatory	Specify the Package Display ID.
<<Modified After	Mandatory	Specify the Date in the format mm-dd-yyyy e.g. 12-31-2005 to Fetch Package Contents Modified

Date>>		After specified Date.

Fields Available

Field	Description
ID	
Name	
Is Pinned To Version	
Version	
Project	
Comparison Status	
Record Type	
Comments Count	
Indentation Level	

Examples

\Fetch_Package_Contents_Modified_After_Date('WPKG-6819','12-31-2005')\
\Fetch_Package_Contents_Modified_After_Date('6819','1-15-2005')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Declare_Variable('Package_ID', 'String')\
\Declare_Variable('Modified_After_Date', 'String')\
\Prompt_For_Variable_Values('Package_ID', 'Modified_After_Date')\

\ Hyphen "-" delimited date format "mm-dd-yyyy" e.g.. 12-31-2005

\Fetch_Package_Contents_Modified_After_Date(Package_ID, Modified_After_Date)\

\scan(a)\\if (Bof(a))\

Record	Comments	Version

\endif\

\if(a : status = ‘Added’)\

**\ a : Id ** \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a: status = ‘Removed’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\elsif(a : status = ‘Modified’)\

\ a : Id \ \ a : Name \	\Fetch_Review_Comments_By_Condition(‘’,’Date’)\ \scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	\ a : Version \

\endif\\endscan\

Screen Mockups

Fetch_Widgets

Compatibility: Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command that can be executed as a Master Report is also used to fetch Widgets on the Screen Mockups Records list.

\Fetch_Widgets(<<Record ID>>)\

Parameter

<<Record ID>>	Mandatory	Specify the Record ID or ID Field Name. If not specified, the command will not work.

Fields Available

Field	Description
Widget ID	
Widget Name	
Widget Type	
Owner Name	
Field Name	
Field Type	
Entity Name	

Examples

\Fetch_Widgets(a:Id)\
\Fetch_Widgets('WFR-1111')\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Diagrams_By_Condition('SMK' ,'', 'Name')\
\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\Fetch_Widgets(a:Id, 'Entity Name, Field Name')\\VAR(LastState) \\ LastState:='' \

Field Name	Field Type	Widget Name	Widget Type

\scan(b)\\if (b : Entity Name <> '')\\if (LastState <> b : Entity Name)\

Entity: \b :Entity Name\

\endif\

\ b : Field Name \	\b:Field Type \	\ b : Widget Name\	\b : Widget Type\

\SET(LastState, b : Entity Name)\ \endif\\endscan\
\endscan\

Fetch_Linked_Records_For_Screen_Mockup

Compatibility: Desktop App Version 7.10 and above.

This primary command fetches embedded Links info for Screen Mockups.

\Fetch_Linked_Records_For_Screen_Mockup(<<Diagram>>,'<<ID Prefix>>', '<<Sort_by_Field>>', <<FetchFamilyType>>)

Parameters

<<Diagram>>	Mandatory	This is the Screen (Diagram) field for the Screen Mockups from which the embedded Link details will be fetched.
<<ID Prefix>>	Optional	Specify the comma separated ID Prefix to find Links to the selected Record Type.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID
ID Prefix	ID Prefix for the Linked Record
Project	Project name of the Linked Record
Project ID	Project ID of the Linked Record (This is an internal property, not visible to the user).
Folder	Folder name of the Linked Record
Control ID	This is the unique ID of the Flow Element to which the Record is linked. **NOTE**: A record can be linked to multiple Flow Elements. Consequently, Multiple Rows will be fetched for one Linked Record (one Row for each Flow Element). The Control ID cannot be generated in the report. It can be used for any other secondary command where the Unique ID of the Flow Element is given as an input. E.g. **Fetch_Widget_Properties**

Examples

\Fetch_Linked_Records_For_Screen_Mockup(a:ID)\
\Fetch_ Linked_Records_For_Screen_Mockup(a:ID,'SMK,BPD')\
\Fetch_ Linked_Records_For_Screen_Mockup(a:ID,'SMK,BPD','Project')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\Fetch_Linked_Records_For_Screen_Mockup(a:Diagram,'BPD,SMK','Type')\
\VAR(LastState) \\ LastState:='' \

Linked Record Name	Project

\scan(b)\
\if (LastState <> b : Type)\

Linked Record Type : \b : Type\

\endif\

\ b : Name \	\ b : Project \

\SET(LastState, b : Type)\
\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \
Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Fetch_Linked_Records_For_Screen_Mockups_By_ID

Compatibility: Desktop App Version 7.10 and above.
This primary command fetches embedded Links info for Screen Mockups.

\Fetch_Linked_Records_For_Screen_Mockup_By_ID('<<ID>>','<<ID Prefix>>', '<<Sort_by_Field>>, <<FetchFamilyType>>)

Parameters

<<ID>>	Mandatory	This is the Display ID (String/Field) for the Screen Mockups from which the embedded Link details

		will be fetched.
<<ID Prefix>>	Optional	Specify a comma separated ID Prefix to find Links to a particular Record Type.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID
ID Prefix	ID Prefix for the Linked Record
Project	Project name of the Linked Record
Project ID	Project ID of the Linked Record (This is an internal property and will not be visible to the user).
Folder	Folder name of the Linked Record
Control ID	Unique ID of the Flow Element to which the Record is linked. **NOTE**: A Record can be linked to multiple Flow Elements. Consequently, Multiple Rows will be fetched for one Linked Record (one Row for each Flow Element).

	The Control ID cannot be generated in the report. It can be used for any other secondary command where the Unique ID of the Flow Element is given as an input. E.g. **Fetch_Widget_Properties**

Examples

\Fetch_ Linked_Records_For_Screen_Mockup_By_ID(a:ID)\
\Fetch_ Linked_Records_For_Screen_Mockup_By_ID(a:ID,'SMK,BPD')\
\Fetch_ Linked_Records_For_Screen_Mockup_By_ID(a:ID,'SMK,BPD','Project')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\Fetch_Linked_Records_For_Screen_Mockup_By_ID(a:Id,'BPD,SMK','Type')\
\VAR(LastState) \\ LastState:='' \

Linked Record Name	**Project**

\scan(b)\
\if (LastState <> b : Type)\

Linked Record Type : \b : Type\

\endif\

\ b : Name \	\ b : Project \

\SET(LastState, b : Type)\
\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \

Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Fetch_Linked_Records_For_Widget

Compatibility: Desktop App Version 7.10 and above.

This secondary command is used to Fetch embedded Links info for Widgets of the specified ID.

\Fetch_ Linked_Records_For_Widget('<<ID>>','<<Fetch_Linked_Records_For_Field>>' , '<<Record Type ID Prefixes comma separated>>','<<Sort_by_Field>>', <<FetchFamilyType>>)

Parameters

<<ID>>	Mandatory	Specify the Control ID field of the Widget from which embedded details will be fetched.
<<Fetch_Linked_Records_For_Field>>	Optional	Speci Yes or No for this parameter to fetch embedded links for the linked entity field for a Widget.
<<Record Type ID Prefixes comma separated>>	Optional	Specify the ID Prefixes of Record Types for which you want to fetch embedded links. If the ID Prefixes are not specified, all embedded links will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the name is not specified, it will be sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a

		Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
If a single Record Type's ID Prefix is specified	Record Type specific columns are available. When ID Prefix is passed in the command, it fetches Record Type Specific fields. E.g. Fetch_Traced_Records_Of_Type('UC') For the above command ,it will fetch Use case specific fields.
Record Type ID Prefix is not specified or multiple ID Prefixes are passed	All the fields of a generic object type are available i.e. Fields displayed in the object list for all element record types.

Examples

\Fetch_Linked_Records_For_Widget(a:ID)\
\Fetch_Linked_Records_For_Widget(a:ID,'Yes', 'BREQ', 'Name')\

Examples

\scan(a) [,page] \

```
\ a : Id \ \  a : Name \
```

\Fetch_Widgets(a:Id)\
\scan(b)\

Widget : \b:Widget Name\ [\b:Widget Type\]

\Fetch_Linked_Records_For_Widget(b:Widget Id)\
\VAR(LastState) \\ LastState:='' \

Record Name	Project	Linked in Properties

\scan(c)\
\if (LastState <> c : Type)\

Linked Record Type : \c : Type\

\endif\

\ c : Name \	\c : Project \	\c : Properties\

\SET(LastState, c : Type)\ \endscan\
\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \
Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Fetch_Linked_Requirements_For_Widget

Compatibility: Desktop App Version 7.10 and above.

This secondary command is used to Fetch embedded Requirement Links info for Widgets of the specified ID.

\Fetch_Linked_Requirements_For_Widget('<<ID>>','<<Fetch_Linked_Records_For_Field>>','<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	Specify the Control ID Field of the Widget from which embedded details will be fetched.

<<Fetch_Linked_Records_For_Field>>	Optional	Specify Yes or No for this parameter for fetching the embedded links for a linked entity field for a Widget.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the name is not specified, it will be sorted by Linked Record Type.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record ID
ID Prefix	Display Preifx for the Linked Record
Project	Project Name of the Linked Record
Project ID	Project ID of the Linked Record
Folder	Owner Name of the Linked Record
Properties	Property names of the widget where this link has been created (as link in description/Rich custom properties)

Examples

\Fetch_Linked_Requirements_For_Widget(a:ID)\
\Fetch_Linked_Requirements_For_Widget(a:ID,'Yes', 'Name')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\Fetch_Widgets(a:Id)\
\scan(b)\

\if(b:Is Significant Widget = 'Yes')\

Widget : \b:Widget Name\ [\b:Widget Type\]

\Fetch_Linked_Requirements_For_Widget(b:Widget Id)\\VAR(LastState) \\ LastState:='' \

Record Name	Project	Linked in Properties

\scan(c)\
\if (LastState <> c : Type)\

Linked Record Type : \c : Type\

\endif\

\ c : Name \	\c : Project \	\c : Properties\

\SET(LastState, c : Type)\ \endscan\
\endif\
\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \
Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Fetch_Screen_Pages

Compatibility: Desktop App Version 7.10 and above.
This secondary command displays the contents of all tab pages and accordion panels. It can also be executed as a Master Report. The sorting default is set to 'Entity Name, Widget Name'.

\Fetch_Screen_Pages(<<Screen_ID>>)

Parameter

<<Screen ID>>	Mandatory	

Fields Available

Field	Description
Owner Widget	Name of the owner widget currently being iterated.
Active Child Widget	Name of the Child Widget currently being shown.

Examples

All Tab pages and Accordion Panels for Screen Mockup

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\Fetch_Screen_Pages(a:Id)\
\scan(b)\

\B: Owner Widget\ \B: Active Child Widget
\Insert_Diagram_Custom(b: Diagram)\

\endscan\
\endscan\

Widget Properties

Fetch_Widget_Field_Properties

Compatibility Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command is used to fetch Field Properties for the Field linked to a Widget in a Screen Mockup. It is used in conjunction with Fetch_Widgets.

\Fetch_Widget_Field_Properties('<<ID>>')

Parameter

<<ID>>	Mandatory	Specify the ID for the Widget from which Linked Field details will be fetched.

Fields Available

Field	Description
Field Name	
Field Type	
Unique	
Mandatory	
Display Caption	
Default Display Widget	
Size	
Precision	
Maximum Value	
Minimum Value	
Allowed	

Values	
List Type	

\ Get_Default_Value(b:Field Name)\: This special command that fetches the Default Value of the field. The Field Name is mandatory and should only be used under the **Fetch_Widget_Field_Properties** command**.**

Example

\Fetch_Widget_Field_Properties(b:Widget Id)\

Examples

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Widgets(a:Id)\
\scan(b)\

Widget: \b:Widget Name\ [\b:Widget Type\]

\Fetch_Widget_Field_Properties(b:Widget Id)\

Field Name	Field Type	Value

\scan(c)\

\ c : Field Name\	\c: Field Type\	\Get_Default_Value(c: Field Name)\

\endscan\
\endscan\
\endscan\

Insert_Widget_Field_Property_Value

Compatibility: Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command is used to Insert Value for a Field Property of a Field linked to a Widget. It is used in conjunction with Fetch_Widgets.

\Insert_Widget_Field_Property_Value(<<ID>>,'<<Field Property Name>>')

Parameters

<<ID>>	Mandatory	This is the Widget ID Field.
<<Field Property Name>>	Mandatory	This is the name of the Field Property from with the Value will be fetched.

NOTE : Any Field Property name specified in Fetch_Widget_Field_Properties can be given as an input parameter as <<Field Property Name>>.

The result of this command will be Simple Text/Rich Text/True-False/Numeric.

Fields Available

There are no fields available for this command.

Example

\Insert_Widget_Field_Property_Value(b:Widget ID,'Field Name')\

Examples

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\Fetch_Widgets(a:Id)\
\scan(b)\

Widget: \b:Widget Name\ [\b:Widget Type\]

Widget Name	Field Name	Type
\ b : Widget Name\	\ Insert_Widget_Field_Property_Value(b:Widget Id,'Field Name') \	\ Insert_Widget_Field_Property_Value(b:Widget Id,'Field Type') \

\endscan\
\endscan\

Fetch_Widget_Properties

Compatibility Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command is used to Fetch Properties for Screen Mockups of a specified Widget ID. It is used in conjunction with Fetch_Widgets. The sorting default is set to Display Sequence Name.

\Fetch_Widget_Properties('<<ID>>', '<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	This is the ID for the Widget from which details will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the name is not specified, it will be sorted by Display Sequence.

Fields Available

Field	Description
Name	Property Name
Type	Property Data Type (Text, Rich Text, Image)
Value	Property Value
Category	Property Category (Applicable to Custom Properties)
Display Sequence	Property Display Sequence (Applicable to Custom Properties)
Is Custom	Indicates if the Property is a Custom Property

Examples

\Fetch_Widget_Properties(a:ID)\
\Fetch_Widget_Properties(a:ID,'Type')\
\Fetch_Widget_Properties(a:ID,'Type,Name')\

Examples

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Widgets(a:Id)\
\scan(b)\

Widget: \b:Widget Name\ [\b:Widget Type\]

\Fetch_Widget_Properties(b:Widget Id,'Category')\\VAR(LastState) \\ LastState:='' \

Property Name	Data Type	Value

\scan(c)\
\if (LastState <> c : Category)\

Category: \c : Category\

\endif\

\ c : Name \	\c:Type \	\if (c: Type = 'Image')\ \Insert_Custom_Image(c : Value)\\elsif (c:Type = 'Rich Text')\\fRTF(c : Value)\\else\\c: Value\\endif\

\SET(LastState, c : Category)\ \endscan\
\endscan\
\endscan\

Insert_Widget_Property_Value

Compatibility: Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command is used to generate a Value of the specified Property of a Widget. It is used in conjunction with Fetch_Widgets.

\Insert_Widget_Property_Value(<<ID>>,'<<Widget Property Name>>')

Parameters

<<ID>>	Mandatory	This is a Widget ID Field.
<<Widget Property Name>>	Mandatory	Specify the Name of the Property for which the Value will be fetched.

NOTE: Any Field Property Name specified in Fetch_Widget_ Properties can be passed as an input parameter as <<Widget Property Name>>. The result of this command will be Simple Text/Rich Text/True-False/Numeric.

Fields Available

There are no fields available for this command.

Examples

\Insert_Widget_Property_Value(a:Widget ID,'Color')\
\Insert_Widget_Property_Value(a:Widget ID,b:Name)\

Examples

\scan(a) [,page] \

Mockup: \ a : Name \ [\ a : Id \]

\Fetch_Widgets(a:Id)\
\scan(b)\

Widget: \b:Widget Name\ [\b:Widget Type\]

Widget Border Visible: \ Insert_Widget_Property_Value (b:Id, 'Border Visible')\
\if (b:Widget Type = 'Rich Text')\

Widget Description

\fRTF(Insert_Widget_Property_Value (b:Id, 'Rich Text'))\ \endif\
\endscan\
\endscan\

Data Definition

Fetch_Field_Template

NOTE: This command "Fetch_Field_Template" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 7.00 and above.

This command was was previously called Fetch_Domain. However, both versions of this command will work the same. It is recommended to use the latest command name Fetch_Field_Template.

Fetch_Domain

NOTE: This command "Fetch_Domain" has been deprecated. As an alternative, use command Fetch_Repository_Objects_By_Condition.

Compatibility: Desktop App Version 6.20 and above.

This command fetches the Domain information of a Record in a list or as a Master Report. Depending upon the type of Domain, you can control which attributes are to be generated in the document.

\Fetch_Field_Template(<<Field Template field>>)

Parameter

<< Field Template field >>	Mandatory	Specify the Field Template Field Name. The default name of the Field is Field Template.

Fields Available

Field	Description
Type	
Default Display Widget	
Size	
Precision	
Maximum Value	
Minimum Value	
Allowed Values	
List Type	

Example

\ Fetch_Field_Template(a : Field Template)\

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Repository_Objects_By_Condition('DOM', '', 'Id')\
\scan(a)\
\ Fetch_Field_Template(a : Field Template)\
\if (! eof(b))\

\a:Name\ [\ a: Id\]

\scan(b)\

Type	\ b : Type \
Default Display Widget	\ b: Default Display Widget\

\if (b: Type = 'Text')\

Size	\ b: Size \

\endif\

\if (b: Type = 'Decimal')\

Size	\ b: Size \
Precision	\ b: Precision \
Maximum Value	\ b: Maximum Value \
Minimum Value	\ b: Minimum Value \

\endif\

\if (b: Type = 'Multiple Select List')\

Allowed Values	\ b: Allowed Values\
List Type	\ b: List Type\

\endif\

\if (b: Type = 'Selection List')\

Allowed Values	\ b: Allowed Values\
List Type	\ b: List Type\

\endif\

\if (b: Type = 'Number')\

Size	\ b: Size \
Maximum Value	\ b: Maximum Value \
Minimum Value	\ b: Minimum Value \

\endif\

\endscan\

\endif\

\endscan\

Fetch_Fields

Compatibility: Desktop App Version 6.20 and above.

This command fetches the Fields information of the Entity Record in a List or as a Master Report. It is used to output the Fields of Entity Records in a document. Depending upon the field type, you can control which attributes are to be generated in the document.

\ Fetch_Fields(<< Fields field of Entity>>, '<<Sort_by_Field>>')

Parameter

<<Field of Entity >>	Mandatory	Specify the Fields, Field of Entity Records. The default Field name is Fields.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted.

Fields Available

Field	Description
Field Name	
Field Type	
Unique	
Mandatory	
Display Caption	
Default Display Widget	
Size	
Precision	
Maximum Value	
Minimum Value	

Allowed Values	
List Type	
Description	

\ Get_Default_Value(b:Field Name)\: This special command fetches the Default Value of the Field. The Field Name is mandatory. It should only be used under the Fetch_Fields command.

Example

\ Fetch_Fields(a : Fields)\
\ Fetch_Fields(a : Fields,' Field Type, Field Name')\

Examples

\scan(a)\

\ a : Name \ [\ a: Id\]

\ Fetch_Fields(a : Fields)\
\if (! eof(b))\

Fields

\scan(b)\

Field Name: \b: Field Name\	
Display Caption	\ b: Display Caption \
Default Value	\ Get_Default_Value(b:Field Name)\

\endscan\
\endif\
\endscan\

Examples

\Fetch_Repository_Objects_By_Condition('ENT', '', 'Id')\
\scan(a) [,page] \

\a:Name\ [\ a: Id\]

\ Fetch_Fields(a : Fields)\

\if (! eof(b))\

Field List					
Field Name	**Field Type**	**Size**	**Is Unique**	**Is Mandatory**	**Display Caption**

\scan(b)\
\if (! eof(b))\

\ b : Field Name \	\ b : Field Type \	\ b : Size\	\ b:Is Unique \	\ b : Is Mandatory \	\b : Display Caption\

\endif\
\endscan\
\endif\

Fetch_Custom_Properties_For_Entity_Field

Compatibility: Desktop App Version 6.20 and above.

Use this command to generate Custom Properties for Fields of Entity Records in the document.

\ Fetch_Custom_Properties_For_Entity_Field ('<<Field Name>>', '<<Sort By>>')

Parameters

<<Field Name>>	Mandatory	Specify the Field Name.
<<Sort By>>	Optional	Specify the Field Names by which you want the results to be sorted. If the name is not specified, the data will sorted by the Display Sequence of the Custom Property.

Fields Available

Field	Description

Name	Property Name
Data Type	Data Type
Value	Property Value
Group	Property Group
Display Sequence	Property Display Sequence

NOTE: If the Data Type Property is Rich Text then use **fRTF** to get the rich text output.

e.g. \if (c:Data Type = 'Rich Text')\ \fRTF(c : Value)\\else\\c: Value\\endif\

Examples

\scan(a)\

Entit: \ a: Name \ [\ a : ID \]

\Fetch_Fields(a : Fields)\
\scan(b)\
\Fetch_Custom_Properties_For_Entity_Field(b : Field Name)\

Field Name: \b: Field Name\

\scan (c)\

\ c : Name \	\if (c:Data Type = 'Rich Text')\ \fRTF(c : Value)\\else\\c: Value\\endif\

\endscan\
\endscan\
\endscan\

Fetch_Field_Template_Custom_Properties

Compatibility: Desktop App Version 7.00 and above.

This command was was previously called Fetch_Domain_Custom_Properties. Both commands work the same. However, it is recommended to use the latest command name, Fetch_Field_Template_Custom_Properties.

Fetch_Domain_Custom_Properties

Compatibility: Desktop App Version 6.20 and above.

Use this command to generate the Custom Properties of Field Templates into the document.

\ Fetch_Field_Template_Custom_Properties ('<<Field Template field >>', '<<Sort By>>')

Parameters

<<Field Template field>>	Mandatory	Specify the Field Template. The default Field Template name is Field Template.
<<Sort By>>	Optional	Specify the Field Names by which you want the results to be sorted. If the name is not specified, the data is sorted by default on the Display Sequence of the Custom Property.

Fields Available

Field	Description
Name	Property Name
Data Type	Property Data Type
Value	Property Value
Group	Property Group
Display Sequence	Property Display Sequence

NOTE: If the Data Type Property is Rich Text then use **fRTF** to get the rich text output.

e.g. \if (c:Data Type = 'Rich Text')\ \fRTF(c : Value)\\else\\c: Value\\endif\

Examples

\scan(a)\

Domain: \ a: Name \ [\ a : ID \]

\Fetch_Field_Template_Custom_Properties(a : Field Template)\

Domain Properties

\scan (b)\

\ b : Name \	\if (b:Data Type = 'Rich Text')\ \fRTF(b : Value)\\else\\b: Value\\endif\

\endscan\
\endscan\

Fetch_Linked_Records_For_Entity_Field

Compatibility: Desktop App Version 7.10 and above.

Use this command to generate embedded Links data for Entity Fields in a document. Depending upon the Field Name, you can specify which embedded Links data can be generated in the document.

\Fetch_Linked_Records_For_Entity_Field('<<ID>>', '<<Record Type ID Prefixes comma separated>>','<<Sort_by_Field>>', ,<<FetchFamilyType>>)

Parameters

<<Field Caption>>	Mandatory	Specify the Field Caption.
<<Record Type ID Prefixes comma separated>>	Optional	Specify the ID Prefixes of Record Types for which you want to fetch embedded Links. If the ID Prefix is not specified, all embedded Links will be fetched.

<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Name is not specified, the data is sorted by default on Display Sequence of the Custom Property.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID
ID Prefix	ID Prefix for the Linked Record
Project	Project name of the Linked Record
Project ID	Project ID of the Linked Record (This is an internal property and will not be visible to the user).
Folder	Folder name of the Linked Record
Properties	Property names of the Widget where this Link was created (as Link in Description field/Rich Custom Properties).

Examples

\ Fetch_ Linked_Records_For_Entity_Field(a : Field Name)\

\ Fetch_Linked_Records_For_Entity_Field(a : Field Name, 'REQ', 'Name')\

\ Fetch_Linked_Records_For_Entity_Field(a : Field Name, 'REQ', 'Name', 'true')\

Examples

Entities

\scan(a) [,page]

\a:Name\

Id: \ a: Id\

Description

\ InsertRtf(a : Description)\

\Fetch_Fields(a : Fields)\\scan(b)\

\Fetch_ Linked_Records_For_Entity_Field(b : Field Name)\\VAR(LastState) \\ LastState:='' \

Field: \b :Field Name\

\scan(c)\

Record Name	**Project**	**Linked in Properties**

\if (LastState <> c : Type)\

Linked Record Type : \c : Type

\endif\

\ c : Name \	\c : Project \	\c : Properties\

\SET(LastState, c : Type)\

\endscan\

Record Information	
Version: \ a : Version \	**Project**: \ a : Project \
Crt by: \ a : Crt by \	**Crt dt**: \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt**: \ a : Upd dt \

\endscan\

Fetch_Linked_Requirements_For_Entity_Field

Compatibility: Desktop App Version 7.10 and above.

Use this command to generate the embedded Requirements Links data for the Fields of Entity Records in a document. Depending upon the Field Name, you can specify which embedded Llinks data can be generated in the document.

\Fetch_Linked_Requirements_For_Entity_Field('<<ID>>', '<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	This is the Display ID of the Entity Record. It can be a field or a string.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If not specified, the data is sorted by default Display Sequence of the Custom Property.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record ID
Display Prefix	Disp Prefix of the Linked Record
Project	Project Name of the Linked Record
Project ID	Project ID of the Linked Record (Internal Property, not to be exposed to the user)
Folder	Linked Record Folder Name
Properties	Property names of the widgets where this link has been created (as Link in Description/ Rich Custom Properties)

Fetch_Property_Value_For_Entity_Field

Compatibility: Desktop App Version 6.3 and above.

Use this command to generate the Value of a given Property for a field of an Entity Record within a document. You can specify the output Value of a particular Field Property. This command is useful when you need to generate a specific Property Value for a field as in Screen Mockups.

\ Fetch_Property_Value_For_Entity_Field(a : ID, b:Fieldname, <<Property name>>)

Parameters

ID	Mandatory	This is the Display ID of an Entity Record. It can be a field or a string.
Field Name	Mandatory	This is the Field Name of the Entity Record. It can be a field or a string.
Property Name	Mandatory	This is the field Property Name. It can be a field or a string.

Fields Available

Field	Description
Field Name	
Field Type	
Unique	
Mandatory	
Display Caption	
Default Display Widget	
Size	
Precision	
Maximum Value	
Minimum Value	

Allowed Values	
List Type	
<<Custom Property Name>>	This is the Custom Property name from which a Value will be fetched.

NOTE: If the Data Type Property is Rich Text, then use **fRTF** to generate the rich text output.

Examples

\scan(a) [,page]\
\a:Name\
Id: \ a: Id\

Examples

Description

\ InsertRtf(a : Description)\
\Fetch_Fields(a : Fields)\\scan(b)\

\b:Field Name\ : \Fetch_Property_Value_For_Entity_Field(a:ID, b : Field Name, 'Display Caption')\

\endscan\

Record Information	
Version: \ a : Version \	**Project**: \ a : Project \
Crt by: \ a : Crt by \	**Crt dt**: \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt**: \ a : Upd dt \

\endscan\

Fetch_Screen_Entity_Fields

Compatibility: Desktop App Version 7.09 and above

This secondary command fetches the Entity Fields details for Entity Fields that are used in Mockups. By default, it will sort on Entity Name, Widget Name.

\Fetch_Screen_Entity_Field_Details(<<Screen ID>>,<<Sort by Field>>)

Parameters

<<Screen ID>>	Mandatory	Specify the ID Field Name.
<<Sort by Field>>	Optional	Specify the comma separated list of Field Names for sorting.

Fields Available

Field	Description
Entity ID	This is for internal use only.
Entity Name	
Project Name	Entity Project Name
Entity Disp ID	This is for internal use only.
Field Name	
Field Type	
Field Type Caption	Field Type Display Caption
Unique	
Mandatory	
Display Caption	
Default Display Widget	
Size	
Precision	

Maximum Value	
Minimum Value	
Allowed Values	
List Type	

Example

\Fetch_Screen_Entity_Field_Details(a:Id)\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\\ <Example Command> \

Examples

\PROJECT_NAME\

Linked Entity Field Details for a Screen Mockup

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\ Fetch_Screen_Entity_Field_Details (a:Id, 'Entity Name, Field Name')\\VAR(LastState) \\ LastState:='' \
\scan(b)\\if (b : Entity Name <> '')\\if (LastState <> b : Entity Name)\

Entity: \b :Entity Name\	Project : \b: Project Name\

\endif\
\Fetch_Custom_Properties_For_Entity_Field(b : Field Name)\

Field: \b :Field Name\ **Type:** \b:Field Type Caption\

Property Name	Property Type	Value

\scan(c)\

\ c : Name \	\ c: Data Type \	\ c : Value\

\endscan\
\SET(LastState, b : Entity Name)\ \endif\\endscan\

\endscan\

Insert_Property_Value_For_Entity_Field

Use this command to generate the Value of a given Property for an Entity Record field within the document. This command is useful when you need to generate a specific Property Value for a field as in Screen Mockups.

\ Insert_Property_Value_For_Entity_Field(a : ID, b:Fieldname, <<Property name>>)

Parameters

ID	Mandatory	Display ID of the Entity Record. Can be a field or a string.
Field Name	Mandatory	Field Name of the Entity Record.
Can be a field or string		

Fields Available

Field	Description
Field Name	
Field Type	
Unique	
Mandatory	
Display Caption	
Default Display Widget	
Size	
Precision	

Maximum Value	
Minimum Value	
Allowed Values	
List Type	
<<Custom Property Name>>	For a given Custom Property Name, the respective Property Value will be fetched.

NOTE: If the Data Type Property is Rich Text, then use **fRTF** to generate the rich text output.

Examples

\scan(a) [,page]\
\a:Name\

Id: \ a: Id\

Description
\ InsertRtf(a : Description)\

\Fetch_Fields(a : Fields)\\scan(b)\

\b:Field Name : \Insert_Property_Value_For_Entity_Field(a:ID, b : Field Name, 'Display Caption')\

\endscan\
\endscan\

OLE Objects

Insert_OLE_Object

Compatibility: Desktop App Version 6.20 and above.

This command inserts the OLE Objects of specified Fields into the generated output.

\Insert_OLE_Object(<<Field Caption>>)

Parameter

<<Field Caption>>	Mandatory	OLE Field Caption

Fields Available

There are no fields available for this command.

Example

\ Insert_OLE_Object(a : Diagram)\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Repository_Objects_By_Condition('FEAT', '', 'Id')\
\scan(a) [,page] \

\a:Name\ [\ a: Id\]

\ Insert_OLE_Object(a : Diagram)\
\endscan\

Does_Text_Contain_Table ()

Compatibility: Desktop App Version 6.40 and above.

This command is used to check if a table is present in the specified rich text field. It returns a Boolean value (**TRUE** or **FALSE).** If the table is present in the field, the result is **TRUE**. If the table is not present, this result is **FALSE**.

This command is needed because the DocProcessor generates distorted tables if there is a table in the rich text field and a field tag is used inside a table cell within the DocProcessor template.

**\ if (Does_Text_Contain_Table (<<Rich text Field>>)) **

Parameter

<<Rich text Field>>	Mandatory	**Rich text field name SUBSTR**

Fields Available

There are no fields available for this command.

Examples

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Requirements_Tree_By_Document_Id_By_Condition('$DEFAULT_REQUIREMENTS_DOCUMENT$')\
\scan(a)\
\if(Does_Text_Contain_Table(a:Description))\

\a: wbs \. [\ a : Id \] \a : Title

\InsertRTF(a: Description)\
\endIf\
\if(! Does_Text_Contain_Table(a:Description)) \

**\a: wbs **	**[\ a : Id \ ] \ a : Title **
\ InsertRtf(a : Description)\	

\endIf\
\endscan\

Business Process Diagram

Fetch_Business_Process_Flows

Compatibility Desktop App Version 6.20 and above.

This primary command is used to Fetch Business Process Flows (Links) between different flow elements on the Business Process Diagram.

\Fetch_Business_Process_Flows('<<ID>>', '<<Flow type>>','<<From ID>>', '<<To ID>>', '<<From Name>>', '<<To Name>>', '<<Sort By Field>>')

Parameters

<<ID>>	Mandatory	This is the Display ID for the Diagram from which the details will be fetched.
<<Flow Type>>	Optional	Flow Type of the Flow (Link) Allowed Values: 'Default Flow' / 'Conditional Flow' / 'Sequence Flow' / 'Message Flow' / 'Association Flow'
<<From ID>>	Optional	From Element ID
<<From Name>>	Optional	From Element Name
<<To ID>>	Optional	To Element ID
<<To Name>>	Optional	To Element Name
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Names are not specified, the data is sorted by default on Owner Pool and Flow Element Type.

Fields Available

Field	Description
Name	Name of the Flow
Type	Type of the Flow (Message/Sequence/Association/etc.)
Text	Text of the flow

From ID	ID of the From shape
From Name	Name of the From shape
From Text	Text of the From shape
From Type	Type of the From shape
From Pool	Owner Pool of the From shape
From Lane	Owner Lane of the From shape
To ID	ID of the To shape
To Name	Name of the To shape
To Text	Text of the To shape
To Type	Type of the To shape
To Pool	Owner Pool of the To shape
To Lane	Owner Lane of the To shape

Examples

\Fetch_Business_Process_Flow(a:ID)\
\Fetch_Business_Process_Flow(a:ID,'Type')\
\Fetch_Business_Process_Flow(a:ID,'Type,Name')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Business_Process_Flows(a:Id)\

Name	Type	Text	From Text	From Type	To Text	To Type

\scan(b)\

\ b : Name \	\b:Type \	\b:Text\	\b : From Text\	\b : From Type\	\b : To Text\	\b: To Type\

\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \
Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Business_Process_Flows(a:Id)\\VAR(LastState) \\ LastState:='' \
\scan(b)\
\if (LastState <> b : From ID)\

From Flow Object : \b: From Text\ [\b:From Type\]			
Name	**Type**	**Text**	**To Flow Object**

\endif\

\ b : Name \	\b : Type \	\b : Text\	\b: To Text\ [\b:To Type\]

\SET(LastState, b : From ID)\\endscan\

Record Information	
Version: \ a : Version \	**Project:** \ a : Project \
Crt by: \ a : Crt by \	**Crt dt:** \ a : Crt dt \
Upd by: \ a : Upd by \	**Upd dt:** \ a : Upd dt \

\endscan\

Fetch_Business_Process_Properties

Compatibility Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command fetches Custom Properties for Business Process Flow Elements for the specified Flow Element ID. It is used in conjunction with Fetch_Business_Process_Flow.

\Fetch_Business_Process_Properties('<<ID>>', '<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	This is the ID of the Flow Element from which details will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Names are not specified, the data is sorted by default on Display.

Fields Available

Field	Description
Name	Property Name
Data Type	Property Data Type
Value	Property Value
Category	Property Category
Display Sequence	Property Display Sequence

NOTE: If the data type property is Rich Text, then use **fRTF** to generate the rich text output.
e.g. \if (c:Data Type = 'Rich Text')\ \fRTF(c : Value)\\else\\c: Value\\endif\

Examples

\Fetch_Business_Process_Properties(a:ID)\
\Fetch_Business_Process_Properties(a:ID,'Data Type')\

\Fetch_Business_Process_Properties(a:ID,'Data Type,Name')\

Examples

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Business_Process_Flow(a:Id,'Type')\
\scan(b)\

Flow Element : \b:Name\ [\b:Type\]

\Fetch_Business_Process_Properties(b:Id,'Category')\\VAR(LastState) \\ LastState:='' \

Property Name	Data Type	Value

\scan(c)\
\if (LastState <> c : Category)\

Category: \c : Category\

\endif\

\ c : Name \	\c:Data Type \	\if (c:Data Type = 'Rich Text')\ \fRTF(c : Value)\\else\\c: Value\\endif\

\SET(LastState, c : Category)\ \endscan\
\endscan\
\endscan\

Insert_Business_Process_Property_Value

Compatibility Desktop App Version 6.20 (Advanced Edition only) and above.

This secondary command is used to Insert a Property Value of a Property for the Business Process Flow Element of a specified Property Name in a Business Process Diagram. It is used in conjunction with Fetch_Business_Process_Flow. It can also be used with Fetch_Business_Process_Flow and Fetch_Business_Process_Custom_Properties.

\Insert_Business_Process_Property_Value(<<ID>>,'<<Property Name>>')

Parameters

<<ID>>	Mandatory	This is the Flow Element ID field.
<<Property Name>>	Mandatory	This is the Name of the Property from which the Value will be fetched.

Fields Available

The Property Value is generated automatically and therefore there are no fields.
NOTE : The result of this command is type Text.

NOTE: If the Data Type Property is Rich Text, then use **fRTF** to generate the rich text output.
e.g. \fRTF(\Insert_Business_Process_Property_Value(b:ID,' Rich Text')\)\

Examples

\Insert_Business_Process_Property_Value(b:ID,'Property Name')\
\Insert_Business_Process_Property_Value(b:ID,b:PropertyNameField)\

Examples

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Business_Process_Flow (a:Id,'Type')\
\scan(b)\

Flow Element : \b:Name\ [\b:Type\]

\Fetch_Business_Process_Properties (b:Id)\
\scan(c)\
\if (c : Data Type = 'Rich Text')\

Property Name : \c: Name\

Property Value : \fRTF(Insert_Business_Process_Property_Value (b:Id, c: Name))\

\endif\\endscan\
\endscan\
\endscan\

Fetch_Business_Process_Elements

Compatibility: Desktop App Version 7.10 and above.
*New alias added for **'Fetch_Business_Process_Flow'**

This primary command is used to Fetch Business Process Elements and their details for the specified ID on a Business Process Diagram.

Fetch_Business_Process_Flow

Compatibility: Desktop App Version 6.10 and above.

\Fetch_Business_Process_Elements('<<ID>>', '<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	This is the Display ID for the Diagram from which the details will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Name is not specified, the data is sorted by default on Owner Pool and Flow Element Type.

Fields Available

Field	Description
ID	Unique ID for the Flow Element
Name	Flow Element Name
Type	Element Type
Description	Element Description
Pool Name	Owner Pool Name

Display Sequence	Flow Element Display Sequence

Examples

\Fetch_Business_Process_Elements(a:ID)\
\Fetch_Business_Process_Elements(a:ID,'Type')\
\Fetch_Business_Process_Elements(a:ID,'Type,Name')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\ Insert_Diagram_Custom(a : Diagram)\

\Fetch_Business_Process_Elements(a:Id,'Type')\

Name	Type	Pool

\scan(b)\

\ b : Name \	\b:Type \	\b:Pool Name\

\endscan\
\endscan\

Fetch_Business_Process_Flows

Compatibility: Desktop App Version 6.20 and above.

This primary command is used to Fetch Business Process Flows (Links) between different Flow Elements on a Business Process Diagram.

\Fetch_Business_Process_Flows('<<ID>>', '<<Flow type>>','<<From ID>>', '<<To ID>>', '<<From Name>>', '<<To Name>>', '<<Sort By Field>>')

Parameters

<<ID>>	Mandatory	This is the Display ID for the Diagram from which details will be fetched.
<<Flow Type>>	Optional	This is the Flow Type of the Flow Link. Allowed Values are: • Default Flow • Conditional Flow • Sequence Flow • Message Flow • Association Flow
<<From ID>>	Optional	From Element ID
<<From Name>>	Optional	From Element Name
<<To ID>>	Optional	To Element ID
<<To Name>>	Optional	To Element Name
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Name is not specified, the data is sorted by default on Owner Pool and Flow Element Type.

Fields Available

Field	Description
Name	Flow Name
Type	Flow Type (Message/Sequence/Association etc.)
Text	Flow Text

From ID	From shape ID
From Name	From shape Name
From Text	From shape Text
From Type	From shape Type
From Pool	From shape owner Pool
From Lane	From shape owner Lane
To ID	To shape ID
To Name	To shape Name
To Text	To shape Text
To Type	To shape Type
To Pool	To shape owner Pool
To Lane	To shape owner Lane

Examples

\Fetch_Business_Process_Flows(a:ID)\

\Fetch_Business_Process_Flows(a:ID,'Type')\

\Fetch_Business_Process_Flows(a:ID,'Type,From ID)\

Sample Template 1

\scan(a) [,page] \

```
\ a : Id \ \  a : Name \
```

\ Insert_Diagram_Custom(a : Diagram)\

\Fetch_Business_Process_Flows(a:Id)\

Name	Type	Text	From Text	From Type	To Text	To Type

\scan(b)\

\ b : Name \	\b:Type \	\b:Text\	\b : From Text\	\b : From Type\	\b : To Text\	\b : To Type\

\endscan\
\endscan\

Sample Template 2

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\ Insert_Diagram_Custom(a : Diagram)\

\Fetch_Business_Process_Flows(a:Id)\\VAR(LastState) \\ LastState:='' \
\scan(b)\
\if (LastState <> b : From ID)\

From Flow Object : \b: From Text\ [\b:From Type\]			
Name	**Type**	**Text**	**To Flow Object**

\endif\

\ b : Name \	\b : Type \	\b : Text\	\b: To Text\ [\b:To Type\]

\SET(LastState, b : From ID)\\endscan\
\endscan\

Fetch_Linked_Records_For_Business_Process

Compatibility: Desktop App Version 7.10 and above.

This primary command fetches embedded Links data for all the Flow Elements of a Business Process Diagram.

\Fetch_Linked_Records_For_Business_Process(<<Diagram Field>>,'<<ID Prefix>>', '<<Sort by Field>>', ,<<FetchFamilyType>>)

Parameters

<<Diagram>>	Mandatory	This is the Diagram Field for the Business Process Diagram from which embedded Links details will be fetched.
<<ID Prefix>>	Optional	Specify the comma separated ID Prefix to display Links to specified Record Types.
<<Sort by Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Name is not specified, the data is sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID
ID Prefix	Linked Record ID Prefix
Project	Linked Record Project Name
Project ID	Linked Record Project ID (This is internal property and is not visible to the user).
Folder	Linked Record Folder Name
Control ID	This is the unique ID of the Flow Element to which the Record is linked.

	NOTE: A record can be linked to multiple Flow Elements. Multiple Rows will be fetched for one Linked Record in such case (one Row for each Flow Element). **NOTE:** The Control ID is not generated in the report. It can be used for any other secondary command where the unique ID of the Flow Element is given as an input.

Examples

\Fetch_Linked_Records_For_Business_Process(a:ID)\
\Fetch_Linked_Records_For_Business_Process(a:ID,'CTX,BPD')\
\Fetch_Linked_Records_For_Business_Process(a:ID,'CTX,BPD','Project')\
\Fetch_Linked_Records_For_Business_Process(a:ID,'CTX,BPD','Project', 'True')\

Examples

\scan(a) [,page] \

\ a : Id \ \ a : Name \

\Fetch_Linked_Records_For_Business_Process(a:Diagram,'BPD,CTX', 'Type')\\VAR(LastState) \\ LastState:=''
\

Linked Record Name	**Project**

\scan(b)\
\if (LastState <> b : Type)\

Linked Record Type : \b : Type\

\endif\

\ b : Name \	\ b : Project \

\SET(LastState, b : Type)\\endscan\
\endscan\

Fetch_Linked_Records_For_Business_Process_By_ID

Compatibility: Desktop App Version 7.10 and above.

This primary command fetches embedded Links details for the Business Process Diagram.

\Fetch_Linked_Records_For_Business_Process_By_ID('<<ID>>','<<ID Prefix>>', '<<Sort_by_Field>>' >>',<<FetchFamilyType>>)

Parameters

<<ID>>	Mandatory	This is the Display ID (String/Field) for the Business Process Diagram from which embedded Link details will be fetched.
<<ID Prefix>>	Optional	Specify the comma separated ID Prefix to display Links of the particular Record Type.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Name is not specified, the data is sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID

ID Prefix	Linked Record ID Prefix
Project	Linked Record Project Name
Project ID	Linked Record Project ID (This is internal property and is not visible to user).
Folder	Linked Record Folder Name
Control ID	This is the unique ID of the Flow Element to which the Record is linked. **NOTE:** A record can be linked to multiple Flow Elements. Multiple Rows will be fetched for one Linked Record in such case (one Row for each Flow Element). **NOTE:** The Control ID is not generated in the report. It can be used for any other secondary command where the unique ID of the Flow Element is given as an input.

Examples

\Fetch_Linked_Records_For_Business_Process_By_ID(a:ID)\
\Fetch_Linked_Records_For_Business_Process_By_ID(a:ID,'CTX,BPD')\
\Fetch_Linked_Records_For_Business_Process_By_ID(a:ID,'CTX,BPD','Project')\
\Fetch_Linked_Records_For_Business_Process_By_ID(a:ID,'CTX,BPD','Project', 'True')\

Examples

\scan(a) [,page] \

```
\ a : Id \ \  a : Name \
```

\Fetch_Linked_Records_For_Business_Process_By_ID(a:Id,'BPD,CTX', 'Type')\\VAR(LastState) \\ LastState:=''
\

Linked Record Name	Project

\scan(b)\

\if (LastState <> b : Type)\

Linked Record Type : \b : Type\

\endif\

\ b : Name \	\ b : Project \

\SET(LastState, b : Type)\\endscan\
\endscan\

Fetch_Linked_Records_For_Business_Process_Element

Compatibility: Desktop App Version 7.10 and above.

This secondary command fetches embedded Links data for the Business Process Flow Element for the specified ID of a Business Process Diagram..

\Fetch_Linked_Records_For_Business_Process_Element(<<ID Field>>, '<<Record Type ID Prefix comma separated>>',‘<<Sort_by_Field>>’), >>’,<<FetchFamilyType>>)

Parameters

<<ID>>	Mandatory	This is the Control ID field for the Flow Element from which embedded Link details will be fetched.
<<Record Type ID Prefix comma separated>>	optional	Specify the Record Type ID Prefix for which you want to fetch embedded Links. If the ID is not specified, all embedded Links will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If the Field Names are not specified, the data is sorted by Linked Record Type.
<<FetchFamilyType>>	Optional	This is a Boolean parameter (Default = True) for specifying whether to fetch all

		family type linked records for a Widget. E.g. It will fetch all linked requirement types for a widget even if you have specified one requirement type such as "BRU" and pass this parameter as True.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID
ID Prefix	Linked Record ID Prefix
Project	Linked Record Project Name
Project ID	Linked Record Project ID (This is an internal property and is not visible to users).
Folder	Linked Record Folder Name
Properties	These are the Property names of the BP Element where this Link has been created (as Link in Description/Rich Custom Properties).

Examples

\Fetch_Linked_Records_For_Business_Process_Element(a:ID)\
\Fetch_Linked_Records_For_Business_Process_Element(a:ID,'ATY','Project')\

Examples

\scan(a) [,page] \

```
\ a : Id \ \  a : Name \
```

\Fetch_Business_Process_Flow(a:Id,'Type')\
\scan(b)\

Flow Element : \b:Name\ [\b:Type\]

\Fetch_Linked_Records_For_Business_Process_Element(b:Id)\\VAR(LastState) \\ LastState:='' \

Record Name	Project	Linked in Properties

\scan(c)\
\if (LastState <> c : Type)\

Linked Record Type : \c : Type\

\endif\

\ c : Name \	\c : Project \	\c : Properties\

\SET(LastState, c : Type)\ \endscan\
\endscan\
\endscan\

Fetch_Linked_Requirements_For_Business_Process_Element

Compatibility: Desktop App Version 7.10 and above.

This secondary command fetches embedded Requirement Links data for the Business Process Flow Element for the specified ID of a Business Process Diagram.

\Fetch_Linked_Requirements_For_Business_Process_Element(<<ID Field>>, '<<Sort_by_Field>>')

Parameters

<<ID>>	Mandatory	This is the Control ID field for the Flow Element from which embedded Link details will be fetched.
<<Sort_by_Field>>	Optional	Specify the Field Names by which you want the results to be sorted. If The Field Names are not specified, the data is sorted by Linked Record Type.

Fields Available

Field	Description
Name	Linked Record Name
Type	Linked Record Type
ID	Linked Record Display ID

Insert_Custom_Image

Compatibility: Desktop App Version 6.20 (Advanced Edition only) and above.

This specialized command is used to generate an Image and is to be used in conjunction with commands like Fetch_Widgets, etc.

\Insert_Custom_Image(<<Image Field>>)

Parameter

<<Image Field>>	Mandatory	This field contains Image data.

Fields Available

There are no fields available for this command.

Examples

\Insert_Custom_Image(c:Value)\

Example

\scan(a) [,page] \

\ a : Name \ [\ a : Id \]

\ Insert_Diagram_Custom(a : Diagram)\
\Fetch_Widgets(a:Id)\
\scan(b)\

Widget: \b:Widget Name\ [\b:Widget Type\]

\Fetch_Widget_Properties(b:Widget Id,'Category')\\VAR(LastState) \\ LastState:='' \

Property Name	Data Type	Value

\scan(c)\
\if (LastState <> c : Category)\

Category: \c : Category\

\endif\

\ c : Name \	\c:Type \	\if (c: Type = 'Image')\ \Insert_Custom_Image(c : Value)\\elsif (c:Type = 'Rich Text')\\fRTF(c : Value)\\else\\c: Value\\endif\

\SET(LastState, c : Category)\ \endscan\
\endscan\
\endscan\

Utility Commands

Date / Time Format Commands

Now

Compatibility: Desktop App Version 3.35 and above.

The NOW command returns the current Date and Time.

\NOW()

Parameters

There are no parameters for this command.

Examples

\Var(LDate)\
\Var(Filter_Condition, LDateStr)\

```
\LDate := Now()\
\LDateStr := DateToStr(Now()) \
\Filter_Condition := '"CSTM Date and Time" > "' + LDateStr +'"' \
\Fetch_Requirements_By_Condition(Filter_Condition)\
\scan(a)  \
\ a : Id \
\ a : Title \
\endscan\
```

DATE

Compatibility: Desktop App Version 3.35 and above.

The DATE command returns the Current Date only.

\DATE()

Parameters

There are no parameters for this command.

TIME

Compatibility: Desktop App Version 3.35 and above.

The TIME command returns the Current Time only.

\TIME()

Parameters

There are no parameters for this command.

DateToStr

Compatibility: Desktop App Version 3.35 and above.

The DateToStr command converts a Date value to a string.

\DateToStr(date)

Parameter

<<date>>	Mandatory	This is the Date value that needs to be converted to a string.

Examples

```
\Var(LDate)\
\Var(Filter_Condition, LDateStr)\
 \LDate := Now()\
 \LDateStr := DateToStr(Now()) \
 \Filter_Condition := '"CSTM Date and Time" > "' + LDateStr +'"' \
 \Fetch_Requirements_By_Condition(Filter_Condition)\
\scan(a)  \
\ a : Id \
\ a : Title \
\endscan\
```

DateTimeToStr

Compatibility: Desktop App Version 3.35 and above.

The DateTimetoStr command converts the Date and Time values to a string.

\DateTimeToStr(datetime)

Parameter

<<datetime>>	Mandatory	This is the Date and Time values that need to be converted to a string.

TimeToStr

Compatibility: Desktop App Version 3.35 and above.

The TimeToStr command returns a string that represents a Time value.

\TimeToStr(time)

Parameter

<<time>>	Mandatory	This is the Time value that needs to be converted to a string.

StrToDate

Compatibility: Desktop App Version 3.35 and above.

The StrToDate command converts a string to a Date value. The Time part is set to 0.

\StrToDate(string)

Parameter

<<string>>	Mandatory	This is the String value that needs to be converted to a Date. The Time part is set to 0.

StrToDateTime

Compatibility: Desktop App Version 3.35 and above.

The StrToDateTime command converts a String to Date and Time values.

\StrToDateTime(string)

Parameter

<<string>>	Mandatory	This is the String value that needs to be converted to a Date. The Time part is set to 0.

StrToTime

Compatibility: Desktop App Version 3.35 and above.

The StrToTime command converts a String to a Time value.

\StrToTime(string)

Parameter

<<string>>	Mandatory	This the String value that needs to be converted to a Time value.

YEAR

Compatibility: Desktop App Version 3.35 and above.

The Year command returns the Year of the specified Date.

\YEAR(date)

Parameter

<<date>>	Mandatory	

MONTH

Compatibility: Desktop App Version 3.35 and above.

The Month command returns the Month of specified Date.

\MONTH(date)

Parameter

<<date>>	Mandatory	

DAY

Compatibility: Desktop App Version 3.35 and above.

The Day command returns the Day of specified Date.

\DAY(date)

Parameter

<<date>>	Mandatory	

SYEAR

Compatibility: Desktop App Version 3.35 and above.

The SYEAR command returns the Year of the Date in a string.

\SYEAR(date)

Parameter

<<date>>	Mandatory	

SDAY

Compatibility: Desktop App Version 3.35 and above.

The SDAY command returns the Day in a string. Days earlier than 10, have zero in place of first digit. For e.g. "01", "02" and so on.

\SDAY(date)

Parameter

<<date>>	Mandatory	This is the Date and Time argument returned in a string format.

DTOS

Compatibility: Desktop App Version 3.35 and above.

The DTOS command converts the Date to a string, formatted as yyyymmdd.

\DTOS(date)

Parameter

<<date>>	Mandatory	This is the Date and Time returned in the yyyymmdd format.

STOD

Compatibility: Desktop App Version 3.35 and above.

The STOD command converts Strings formatted as yyyymmdd to Date values.

\STOD(string)

Parameter

<<string>>	Mandatory	

Date Format Commands

Fdtm

Compatibility: Desktop App Version 3.35 and above.

The Fdtm command formats the Date and Time parameters and returns the values in the specified format. If the String specified by the Format String parameter is empty, the Date and Time values will be formatted as if a 'c' format specifier had been supplied.

\Fdtm(<<Date>>, '<<Format String>>')

Parameters

<<Date>>	Optional	In this field, you can set a Date variable.
<<Format String>>	Optional	This the date format in which you want to display the Date.

Format Strings

Specifier	Displays
c	Displays the Date using the format given by the ShortDateFormat global variable, followed by the Time using the format given by the LongTimeFormat global variable. The Time is not displayed if the date-time values indicate midnight.
d	Displays the Day as a number without a leading zero (1-31).
dd	Displays the Day as a number with a leading zero (01-31).
ddd	Displays the Day as an abbreviation (Sun-Sat) using the strings given by the ShortDayNames global variable.
dddd	Displays the Day as a full name (Sunday-Saturday) using the strings given by the LongDayNames global variable.
ddddd	Displays the Date using the format given by the ShortDateFormat global variable.
dddddd	Displays the Date using the format given by the LongDateFormat global variable.
e	Displays the Year in the current period/era as a number without a leading zero (Japan, Korea and Taiwan only).
ee	Displays the Year in the current period/era as a number with a leading zero (Japan, Korea and Taiwan only).
g	Displays the period/era as an abbreviation (Japan and Taiwan only).

gg	Displays the period/era as a full name (Japan and Taiwan only).
m	Displays the Month as a number without a leading zero (1-12). If the **m** specifier immediately follows an **h** or **hh** specifier, the Minute rather than the Month is displayed.
mm	Displays the Month as a number with a leading zero (01-12). If the **mm** specifier immediately follows an **h** or **hh** specifier, the Minute rather than the Month is displayed.
mmm	Displays the Month as an abbreviation (Jan-Dec) using the strings given by the ShortMonthNames global variable.
mmmm	Displays the Month as a full name (January-December) using the strings given by the LongMonthNames global variable.
yy	Displays the Year as a two-digit number (00-99).
yyyy	Displays the Year as a four-digit number (0000-9999).
h	Displays the Hour without a leading zero (0-23).
hh	Displays the Hour with a leading zero (00-23).
n	Displays the Minute without a leading zero (0-59).
nn	Displays the Minute with a leading zero (00-59).
s	Displays the Second without a leading zero (0-59).

ss	Displays the Second with a leading zero (00-59).
z	Displays the Millisecond without a leading zero (0-999).
zzz	Displays the Millisecond with a leading zero (000-999).
t	Displays the Time using the format given by the ShortTimeFormat global variable.
tt	Displays the Time using the format given by the LongTimeFormat global variable.
am/pm	Uses the 12-hour clock for the preceding **h** or **hh** specifier, and displays **'am'** for any Hour before noon, and **'pm'** for any Hour after noon. The **am/pm** specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
a/p	Uses the 12-hour clock for the preceding **h** or **hh** specifier, and displays **'a'** for any Hour before noon, and **'p'** for any Hour after noon. The **a/p** specifier can use lower, upper, or mixed case, and the result is displayed accordingly.
ampm	Uses the 12-hour clock for the preceding **h** or **hh** specifier, and displays the contents of the TimeAMString global variable for any Hour before noon, and the contents of the TimePMString global variable for any Hour after noon.
/	Displays the Date Separator Character given by the DateSeparator global variable.
:	Displays the Time Separator Character given by the TimeSeparator global variable.

'xx'/"xx"	Characters enclosed in single or double quotes are displayed as is, and do not affect formatting.

Example

\ DateVar := NOW \
\ FormatVar :='"The meeting is on" dddd, mmmm d, yyyy, "at" hh:mm AM/PM' \
\ Fdtm(DateVar, FormatVar) \

DateTime

Compatibility: Desktop App Version 13 and above.

The DateTime command is used to format DateTime type fields to the desired format. This is a miscellaneous command and can be used inside Scan-Endscan.

\Format_Date_Time('<<DateTime Field Name>>','<<Format_String>>')

Parameter

<<DateTime Field Name>>	Mandatory	Specify the field names to format DateTime value to the desired format. If not specified, this command will not work.
<<Format_String>>	Optional	Specify the Format_String to format DateTime value to the desired format. If not specified, this command will return the current value of the field.

Example

\ Format_Date_Time(a : Crt dt , ' dddd, mmmm d, yyyy,') \

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\

\scan(a) \

**[\ a: Id\]-\ a : Name **

'DD-MMM-YYYY'

\ Format_Date_Time(a : Crt dt , 'DD-MMM-YYYY')\

MMM dd, yyyy

\ Format_Date_Time(a : Crt dt , ' MMM dd, yyyy')\

'm/d/yyyy'

\ Format_Date_Time(a : Crt dt , 'm/d/yyyy')\

dddd, mmmm d, yyyy,

\ Format_Date_Time(a : Crt dt , ' dddd, mmmm d, yyyy,')\

\endscan\

String Format Commands

STR

Compatibility: Desktop App Version 3.35 and above.

The Str command formats a string and returns it to a variable. It converts an Integer-type decimal expression to a String according to the Width and Decimal formatting parameters.

\STR(Number, Length, Decimals)

Parameters

<<Number>>	Mandatory	This specifies the numeric expression.
<<Length>>	Optional	This specifies the Length of the Character String that the command should return. If passed as 0, but at the same time, Decimals are not zero, the resulting String is trimmed with the trim() function.
<<Decimals>>	Optional	This specifies the number of Decimal places in the Character String that the command should return. If you specify fewer Decimal places than are in the numeric expression, the return value is rounded up. If Decimals aren't included, the number of Decimal places defaults to zero.

SUBSTR

Compatibility: Desktop App Version 3.35 and above.

The SUBSTR command returns characters from the given source string '**S**'.

**\ SUBSTR(S, Start Position, Length) **

Parameters

<<S>>	Mandatory	Parameter '**S**' specifies the character expression from which the character string is returned.
<<Start Position>>	Mandatory	StartPos specifies the position in the character expression from where the character string is returned. The first character of '**S**' is position 1. If StartPos is greater than the number of characters in source string, the empty string is returned.
<<Length>>	Optional	Optional count specifies the number of characters to return from string. If you omit count, characters are returned until the end of the source string is reached.

Example

```
StringVar := "This is non-numeric Value."
\SUBSTR(StringVar, 9, 11)\
```

Above example with output non-numeric.

VAL

Compatibility: Desktop App Version 3.35 and above.

The Val command converts a string to a numeric expression.

\VAL(s)

Parameter

<<s>>	Mandatory	If string value '**S**' is passed as a parameter then it converts to its equivalent numeric representation. If '**S**' is an invalid String, an exception is raised.

Example

```
StringVar := "This is not a numeric Value. VAL will raise error."
\VAL(StringVar)\
```

UPPER

Compatibility: Desktop App Version 3.35 and above.

The UPPER command returns the specified Character expression in uppercase.

\UPPER(s)

Parameter

<<s>>	Mandatory	

LOWER

Compatibility: Desktop App Version 3.35 and above.
The LOWER command returns the specified Character expression in lowercase.

\LOWER(s)

Parameter

<<s>>	Mandatory	

COPY

Compatibility: Desktop App Version 3.35 and above.

The Copy command copies a Character from the specified Position in the '**S**' String. The parameter StartPos and Optional Count determine the Position from which the Characters would return from the source string as well as the Number of Characters from the source String.

\COPY(S, Start Position, Count)

Parameters

<<S>>	Mandatory	'**S**' specifies the Character Expression from which the Character String is returned.
<<Start Position>>	Mandatory	This specifies the Position in the Character Expression from where the Character String is returned. The first character of '**S**' is position 1. If StartPos is greater than the Number of Characters in the Source String, the empty String is returned.
<<Count>>	Optional	This specifies the Number of Characters to return from the String.

POS

Compatibility: Desktop App Version 3.35 and above.

The POS command searches the String 'Substr' within String '**S**' and returns an Integer value. The returned value is the index of the first character of 'Substr' with '**S**'. The POS is case-sensitive. If the Substr is not found, POS returns zero.

\POS(Substr, S)

Parameters

<<Substr>>	Mandatory	This is the part of the String which is searched from within the Source String '**S**'.
<<S>>	Mandatory	This is the Source String out of which part of the Substr is returned.

TRIM

Compatibility: Desktop App Version 3.35 and above.

The TRIM command removes leading and trailing spaces and control characters from a string.

\TRIM(s)

Parameter

<<s>>	Mandatory	This is the Source String from which leading and trailing spaces and control characters are removed.

flnk

Compatibility: Desktop App Version 3.35 and above.

The FLNK command generates a clickable Link. The Link caption is the same as the Link parameter that is passed.

\flnk(var)

Parameter

<<var>>	Mandatory	URL

Example

\Set_Project('$CURRENT_PROJECT$')\
\Fetch_Use_Cases_By_Condition(' "State" = "All Open" ', 'Priority, Name') \
\scan(a)\

Project : \a:Project\
Name : [\a:Name\]
ID : [\a:Id\] **State**: \a:state\ **Priority** : \a:priority\

\flnk('http://www.Google.com')\
\endscan\

Mathematical Commands

FormatFloat

Compatibility: Desktop App Version 3.35 and above.

The FORMATFLOAT command formats a Floating Point value given by the value using the String given by FormatString.

\FormatFloat(FormatString, Value)\

Parameters

<<FormatString>>	Mandatory	
<<value>>	Mandatory	

Specifier	Represents
0	This is the Digit placeholder. If the value being formatted has a Digit in the position where the **'0'** appears in the format String, then that Digit is copied to the output String. Otherwise, a **'0'** is stored in that position in the output String.
#	This is the Digit placeholder. If the value being formatted has a Digit in the position where the **'#'** appears in the format String, then that Digit is copied to the output String. Otherwise, nothing is stored in that position in the output String.
.	This is the Decimal Point. The first '**.**' character in the format String determines the location of the Decimal Separator in the formatted value. Any additional '**.**' characters are ignored. The actual character used as the Decimal Separator in the output String is determined by the Decimal Separator global variable. The

	default value of the Decimal Separator is specified in the number format of the International section in the Windows Control Panel.
,	This is the Thousand Separator. If the format String contains one or more '**,**' characters, the output will have the Thousand Separators inserted between each group of three digits to the left of the Decimal Point. The placement and number of '**,**' characters in the format String does not affect the output, except to indicate that Thousand Separators are wanted. The actual character used as the Thousand Separator in the output is determined by the Thousand Separator global variable. The default value of the Thousand Separator is specified in the number format of the International section in the Windows Control Panel.
E+	This is a scientific notation. If any of the Strings **'E+', 'E-', 'e+', or 'e-'** is contained in the format String, the number is formatted using a scientific notation. A group of up to four **'0'** characters can immediately follow the **'E+', 'E-', 'e+', or 'e-'** to determine the minimum number of Digits in the exponent. The **'E+**' and **'e+**' formats cause a plus sign to be outputted for positive exponents. The **'E-**' and **'e-**' cause a minus sign to be outputted for negative exponents.
'xx'/"xx"	The characters enclosed in single or double quotes are outputted as is, and do not affect formatting.
;	The semicolon separates sections for positive, negative, and zero numbers in the format String.

ROUND

Compatibility: Desktop App Version 3.35 and above.

The ROUND command rounds a Real-type value to an Integer. A Real-type value is always processed to the largest Integer.

\ROUND(n,decimals)

Parameters

<<n>>	Mandatory	It is a Real-type value.
<<decimals>>	Mandatory	It is an Integer value.

INT

Compatibility: Desktop App Version 3.35 and above.

The INT command returns the Integer part of a Real Number in a parameter. The INT command rounds the Real Number to zero.

\INT(number)

Parameter

<<number>>	Mandatory	This is a Real-type value.

FRAC

Compatibility: Desktop App Version 3.35 and above.

The FRAC commands returns a Fractional part from the Real-type Number in a parameter.

\FRAC(number)

Parameter

<<number>>	Mandatory	It is a Real-type value.

POWER

Compatibility: Desktop App Version 3.35 and above.
The POWER command raises the Base to any Power. If the Fractional exponents or exponents are greater than MaxInt, the Base must be greater than 0.

\POWER(base, exponent)

Parameters

<<base>>	Mandatory	
<<exponent>>	Mandatory	

INTPOWER

Compatibility: Desktop App Version 3.35 and above.

The INTPOWER command calculates the Integral Power of a Base value. The IntPower raises the Base to the Power specified by exponent.

\INTPOWER(base, exponent)

Parameters

<<base>>	Mandatory	
<<exponent>>	Mandatory	