TopTeam Requirements DocProcessor:

Commands Reference

Revised: April 23rd, 2025

Techno Solutions

Table of Contents

Table of Contents 2

Introduction 8

- Overview 8
- What is this guide about? 8

Learning "How to build report templates" 8

Command Syntax 9

- Fetch Commands 9
- Fetch Command Parameters 9
- Case Insensitivity 10
- Enclosing Backslash 10
- Open-Closed Parenthesis 10
- Commands Should Not be Split with a New Line or Enter 11
- Command Parameters 11

System Field Tags 12

System Variable Tags 12

- Fetch_Large_Field_Data 12

Set and Clear Context Commands 13

- Set_Project 13
- Set_Project_By_ID 15
- Clear_Project 15

Repository Objects Commands 16

- Fetch_Repository_Objects_By_Conditions 16
- Fetch_Repository_Object_By_ID 17
- Fetch_Repository_Objects_By_Hierarchy_By_Condition 17
- Fetch_Link_Hierarchy_For_Repository_Object 22

OneView Commands 24

- Fetch_One_View_Section_By_Name 24
- Fetch_One_View_Pages and Records 26

Commands for Use Case Scenarios 27

InsertFlows ... 28
Fetch_Main_Success_Scenario ... 28
Fetch_Extensions ... 29
Fetch_Variations ... 30
Fetch_Main_Flow_of_Events ... 31
Fetch_Alternate_Flow ... 32
Commands to Fetch Steps within each Flow ... 33
Fetch_Steps_For_Flow ... 33
Insert_Activity_Diagram_Custom_In_CMs ... 36
Insert_Activity_Diagram_Custom_In_Inches ... 36
Use Case Diagram Commands ... 38
Special Fields Available ... 38
Sub-reports ... 38
Diagram Commands ... 39
Fetch_Diagram_Compare ... 39
Commands for Inserting Diagrams into a Document ... 42
Insert_Diagram_Custom ... 42
Fetch Sub-report Commands for Repository Objects and Tracking Items ... 45
Comments Sub-report Commands ... 45
Fetch_Comments_By_Condition ... 45
Attachments Sub-report Commands ... 48
Fetch_Attachments_By_Condition ... 48
Insert_Attachment ... 49
Insert_Attachments ... 50
Workflow Comments Sub-report Commands ... 51
Fetch_WorkFlow_Comments_By_Condition ... 51
Version History Sub-report Commands ... 53
Fetch_Versions_By_Condition ... 53
Audit Log Sub-report Commands ... 54
Fetch_Audit_Logs_By_Condition ... 54
Fetch_Audit_Log_Detail ... 55
Traceability Sub-report Commands ... 56
Fetch_Traced_Records_of_Type_by_Condition ... 57
Variants Sub-Reports Commands ... 60
Fetch_Variants_By_Condition ... 60
Linked Issues Sub-report Commands ... 62

Fetch_Linked_Issues_By_Condition *62*

Tracking Items Commands 62

Fetch Tracking Items Commands 62

Fetch_Tracking_Item_By_ID *62*

Fetch_Tracking_Items_By_Condition *63*

Fetch_Tracking_Items *64*

Test Management Commands 65

Fetch_Test_Case_Steps 65

Fetch_Test_Results_In_Test_Run 67

Fetch_Test_Runs_By_Condition 70

Project Based Commands 72

Fetch_Projects 72

Fetch_Project_By_ID 74

Fetch_Project_Inclusion_Types 76

Fetch_Project_Team_Members 77

Fetch_Roles 80

Fetch_Roles_Granted_To_Team_Members 81

Fetch_Privileges_Granted_To_Roles 83

Fetch_Team_Members_Privileges 84

Fetch_Project_Baselines 87

User Based Commands 88

Fetch_Users 88

Fetch_System_Privileges 91

Fetch_Projects_For_User 92

Fetch_User_Groups 93

Fetch_User_Group_Members 94

Fetch_Groups_Where_User_Is_Member 97

Conditional Output using Sections 98

Pre-selecting Sections 98

Dividing Document into Sections 99

Commonly Used Commands 100

InsertRTF 100

Insert_Rich_Text_Using_Word 101

FRTF_Using_Word() 102

InsertRtfAsText 105
Insert_Indented_Rtf 106
Is_Field_Non_Empty 106
Is_Field_Empty 107
Is_Value_Selected 107
Insert_URL_For_Record 109
Insert_URL_for_Record_ID 110
Insert_URL(Caption, Address) 112
Insert_Permalink 113
Insert_Bookmark_For_Record 116
Insert_Formatted_Title 118
Insert_Record_Location 119
Insert_Linked_Records_Field 120
Insert_External_Artifact_URL 122
Insert_Component_Location 123
Insert_User_Image 124
Insert_Multi_Value_Field 126
Comments 127
Execute_Template_For_Id 127
Replace_Text 129

Record Type Commands 130

Fetch_Record_Type_Details 130
Fetch_Record_Type_Associations 132
Fetch_Record_Type_Project_Inclusions 133
Fetch_Record_Type_States 135
Fetch_Record_Type_State_Allowed_Action 136
Fetch_Record_Type_State_Transition 137
Fetch_Record_Type_Versioning_Setting 139
Fetch_Record_Type_Columns 140
Insert_Record_Type_Image 142
Fetch_Record_Fields 144
Fetch_Report_Section 146

Commands to Insert Application Logo and Company Logo 148

Insert_Application_Logo 148
Insert_Company_Logo 148

Baseline Commands 150

Fetch_Compare_Records 150
Fetch_Compare_Versions 154
Insert_Differences 156
Insert Diagram in Compare Versions 157
Insert_Field_As_Diagram_In_CMs 157
Insert_Field_As_Diagram_In_Inches 157
Declare Variable 158
VAR 158
Declare_Variable 159
Declare_Variable_Filter 160
Declare_Variable_ID 162
Insert_Custom_Variable 165
Review Packages Commands 166
Fetch_Package_Contents 166
Fetch_Approvals_For_Repository_Object_By_Condition 168
Fetch_Package_Approvals_By_Condition 170
Fetch_Compare_Package_Baselines 173
Fetch_Package_Baselines 176
Fetch_Review_Comments_By_Condition 177
Fetch_Package_Contents_With_Comments_Modified_After_Date 179
Fetch_Package_Contents_Modified_After_Approval 181
Fetch_Package_Contents_Modified_After_Date 183
Business Process Diagram 186
Fetch_Business_Process_Flows 186
Fetch_Business_Process_Properties 189
Insert_Business_Process_Property_Value 191
Fetch_Business_Process_Elements 192
Fetch_Business_Process_Flow 192
Fetch_Business_Process_Flows 194
Fetch_Linked_Records_For_Business_Process 197
Fetch_Linked_Records_For_Business_Process_By_ID 199
Fetch_Linked_Records_For_Business_Process_Element 201
Fetch_Linked_Requirements_For_Business_Process_Element 204
Insert_Custom_Image 204
Utility Commands 206

Date / Time Format Commands 206
Now 206
DATE 206
TIME 207
DateToStr 207
DateTimeToStr 207
TimeToStr 208
StrToDate 208
StrToDateTime 208
StrToTime 209
YEAR 209
MONTH 209
DAY 210
SYEAR 210
SDAY 210
DTOS 211
STOD 211
Date Format Commands 211
Fdtm 211
DateTime 215
String Format Commands 217
SUBSTR 218
VAL 218
UPPER 219
LOWER 219
COPY 220
POS 220
TRIM 221
flnk 221
Mathematical Commands 222
FormatFloat 222
ROUND 224
INT 224
POWER 225
INTPOWER 225

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 based on your needs.

What is this guide about?

This guide lists the commands available in the DocProcessor document generation engine to develop 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.

Example

\Fetch_Repository_Objects_By_Condition('UC', ' "Priority" = "Very High" ', 'Priority, Name') \

Fetch Command Parameters

<<ID Prefix>>	Specify the Record Type Prefix, such as 'UC', that appears in front of the identifier. For example, in the identifier UC-2348, UC is the ID Prefix and represents Use Cases.
<< Filter Name>>	Specify the Filter. The <<Filter Name>> specified here should be defined in the List/Tree editor for the current specified record type. The command will fetch all records for the set Project or Baseline if not specified.
<< 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. The command will fetch the records that satisfy the specified filter condition. The command will fetch all records for the set Project or Baseline if not specified.

Case Insensitivity

Command names are not case-sensitive.

Enclosing Backslash

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

Example

\Fetch_Repository_Objects_By_Condition('UC', ' "State" = "Approved" ')\

Open-Closed Parenthesis

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

Example

\Fetch_Repository_Objects_By_Condition('')\
\Fetch_Repository_Objects_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. Occasionally, 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','Traces 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_Repository_Objects('UC', 'Approved Status', '')\

Multiple parameter values must be separated by commas.

Example

\Fetch_Repository_Objects("UC', '', '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_Repository_Objects('', '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: Disp 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.

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('$PARENT_OF_CURRENT_PROJECT$')\

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()\

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 that can 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 'STORY', and 'DOC' that appear in front of the identifiers.
<< Filter Condition>>	Optional	
<< Sort Order>>	Optional	

Examples

\ Fetch_Repository_Objects_By_Condition('STORY', ' "Priority" = "High"', 'State') \

This command fetches all User Story Types of records that have the Priority set to High, and sorted by State.

\ 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_By_Hierarchy_By_Condition

Compatibility: Desktop App Version 21.0 and above.

This command retrieves 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_Condition>>','<<WBS_Code>>', ' <<Show_Parent>>','<<Linked_Record_field< Names>>')

Parameters

<<ID_Prefix>>	- Mandatory - Optional	Specify the ID Prefix of Record Type you want to retrieve records hierarchy for the record type prefix specified from the current project set in the report. E.g., 'UC',

		'TC', etc. • This is mandatory if the second parameter **<<Starting_Record_ID >>** is not provided. • This is optional if the second parameter **<<Starting Record_ID >>** is specified.
<<Starting_Record_ID >>	- Mandatory - Optional	This fetches the hierarchy of the record for the starting Record ID specified. • This is mandatory if the first parameter **<<ID_Prefix>>** is not provided. • This is optional if the first parameter **<<ID_Prefix>>** is provided.
<<Filter_Condition>>	Optional	This parameter retrieves desired records such as records with Priority = High.
<<WBS_Code>>	Optional	The WBS Code parameter determines the starting point for the generated records with the specified value. • If WBS Code is specified, The WBS Code for the generated records will start with the specified value as the base value. E.g., If WBS code value is 5, the generated records will start from 5.1.

		• If this parameter is not specified, the WBS Code for each record type will default to the settings defined in the project settings. • Additionally, the WBS Code changes based on the records fetched: If a data filter used prevents parent records from being fetched, the WBS Code for child records will adjust accordingly.

<<Show_Parent>>	Optional	This Boolean parameter accepts the values of **TRUE** and **FALSE.** By default, it is set to **FALSE**.
		
		• **If the filter condition is not specified**:
		  ○ If FALSE, header records are ignored and not inserted in the report.
		  ○ If TRUE, the command fetches the header records in the report.
		• If the filter condition is specified:
		  ○ If FALSE, it ignores parent records that do not satisfy the filter condition of child records that do satisfy the filter condition.
		  ○ If TRUE, the command fetches the parent records of child records that satisfy the filter condition, even if the parent

		records do not satisfy the filter condition.
<<Linked_Record_field< Names>>	Optional	Specify comma separated display captions for Linked record fields to be displayed in the report.

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 special fields are available for hierarchy records:

\ a : Indentation Level \

\ a : WBS \

These fields are not accessible in the editor but 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('GOBJ', '', '','5', 'True','LRFD1, LRFU1')\
- \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(‘’, ‘343’, ‘’,‘5’, 'True','LRFD1, LRFU1’)\

Example

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\ Comments(“Fetch record hierarchy for the specified record type ”)\

Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘GOBJ’)
\Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘GOBJ’) \
\scan(a)\

\a:Wbs \a:Id\ \a:Title\

\endscan\

\ Comments(“Fetch record hierarchy for the specified record type ”)\

Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘GOBJ’,‘’, ‘Crt by = Me’,‘2’)
\Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘GOBJ’,‘’, ‘Crt by = Me’,‘2’) \
\scan(a)\

\a:Wbs \a:Id\ \a:Title\

\endscan\

\ Comments(“Fetch record hierarchy for the specified record type ”)\

Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘BRULE’)
\ Fetch_Repository_Objects_By_Hierarchy_By_Condition(‘BRULE’)\
\scan(a) \
\if (! eof(a))\
\if (a : Indentation Level = ‘1’)\

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

\elsif (a : Indentation Level = ‘2’)\

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

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

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

\else\

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

\endif\

\endif\

\endscan\

Fetch_Link_Hierarchy_For_Repository_Object

Compatibility: Desktop App Version 16 and above.

This command can be used to generate a link hierarchy for the specified record.

\Fetch_Link_Hierarchy_For_Repository_Object('<<Starting Record ID >>','<<Link Type>>', '<<Hierarchy Level>>')

Parameter

<<Starting Record ID>>	Mandatory	Fetches record hierarchy for the specified starting record ID, which is the "Record Identifier" of the record. This parameter can be used for both Repository Objects Or Tracking Items and can be either a Field or String. If the specified record <<ID>> is not present, this command will generate an error.
<<Link Type>>	Optional	Fetches linked records of a specified type. If not specified, it will fetch a linked record hierarchy of Trace type.

<<Hierarchy Level>>	Optional	Fetches linked record hierarchy up to specified level. If not specified, by default, it will fetch a linked record hierarchy upto 15 levels.

Fields Available

Field	Description
Name	This is the name of the associated Record.
ID	
Link Type	
WBS	
Indentation Level	
Disp.ID	
TraceDirection	

Example

\Fetch_Link_Hierarchy_For_Repository_Object(a:ID)\

\Fetch_Link_Hierarchy_For_Repository_Object(a:ID, 'Composed Of')\

\Fetch_Link_Hierarchy_For_Repository_Object(a:ID, 'Composed Of','10')\

Example

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\ Comments("Fetch record link hierarchy for the specified record ")\

Fetch_Link_Hierarchy_For_Repository_Object(a:ID)

\Fetch_Link_Hierarchy_For_Repository_Object(a:ID)\

\scan(a)\

\a:Wbs \a:Id\ \a:Name\

\endscan\

\ Comments("Fetch record hierarchy for the specified record and link type ")\

Fetch_Link_Hierarchy_For_Repository_Object(a:ID, 'Composed Of','10')

\ Fetch_Link_Hierarchy_For_Repository_Object(a:ID, 'Composed Of','10')\
\scan(a) \
\if (! eof(a))\
\if (a : Indentation Level = '1')\

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

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

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

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

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

\else\

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

\endif\
\endif\
\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 the 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 the 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,, 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: Disp ID)\] \ d : Name \

\elsif (d : Indentation Level = 2)\

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

\elsif (d: Indentation Level = 3)\

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

\else\

\d: wbs \ [\Insert_Permalink(d: Disp 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.

This is a 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 the 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 is not present, 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: Disp ID)\] \ c : Name \

\elsif (c : Indentation Level = 2)\

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

\elsif (c: Indentation Level = 3)\

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

\else\

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

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

Commands for Use Case Scenarios

The following commands are used to insert Use Case Scenarios into a document.

InsertFlows

Compatibility: Desktop App Version 3.35 and above.

This command inserts the entire Use Case Scenarios 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_Repository_Objects_By_Condition, Fetch_ Repository_Objects _by_ID,** 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 Scenarios field is Scenarios.

Examples

\Fetch_Repository_Objects ('UC')\
\ scan(a) \]\

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

\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 Scenario.

\ Fetch_Main_Success_Scenario()

Fetch_Extensions

Compatibility: Desktop App Version 4.20 and above.

This command fetches the Header Records for the Alternate Scenarios, Alternate Paths, or Variations present in a Use Case Scenarios.

\ 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 Scenarios.

\ 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 Scenarios.
FLOW_TYPE	Flow Types: M – Main Flow-of-Events (or Main Path or Main Success Scenario) E – Extensions (or Alternate Scenarios 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 : Scenarios)\

Examples

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

\Fetch_Main_Flow_of_Events (a : Scenarios)\

\scan(b)\\if (! Eof(b))\
\FRtf(b: FLOW_NAME_Rich_Text)\
\endif\
\ Fetch_Steps_For_Flow(a : Scenarios, 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 Scenarios 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 Scenarios of the Main Flow.

Example

\Fetch_Alternate_Flow(a : Scenarios)\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

\Fetch_Repository_Objects_By_Condition('UC', ' "State" = "Casual" ')\
\scan(a) \

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

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

\FRtf(b: Flow_Name_Rich_Text)\

\endif\

\Fetch_Steps_For_Flow(a : Scenarios, 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 fetch each Step of the Flow individually using the commands below. 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_LEVEL	This is the indentation level of the Step. The indentation 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 : Scenarios)\

Flow

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

\b: FLOW_NAME \

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

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

\endscan\ \endscan\ \endscan\

Examples

\Fetch_Repository_Objects_By_Condition('UC')\
\scan(a) \

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

\ Fetch_Extensions(a : Scenarios)\

Flow

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

\b: FLOW_NAME \

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

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

\endscan\ \endscan\ \endscan\

Examples

\Fetch_Repository_Objects_By_Condition('UC')\
\scan(a) \

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

\ Fetch_Variations(a : Scenarios)\

Flow

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

\b: FLOW_NAME \

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

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

\endscan\ \endscan\ \endscan\

Insert_Activity_Diagram_Custom_In_CMs

Insert_Activity_Diagram_Custom_In_Inches

Compatibility: Desktop App Version 4.50 and above.

NOTE: You can use the **Insert_Custom_Diagram** command described in this document.

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	Mandatory	The Activity Diagram is generated for Use Case Flows

Case>>		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: scenarios, 'JPEG')\
\Insert_Activity_Diagram_Custom_In_Inches(a: scenarios, 'PNG', '7', '')\
\Insert_Activity_Diagram_Custom_In_CMs(a: scenarios, 'GIF', '', '20')\

Examples

\scan(a) \

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

\ InsertFlows(a : Scenarios)\

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

Examples

\scan(a) \

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

\ InsertFlows(a : Scenarios)\

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

Examples

\scan(a) \

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

\ InsertFlows(a : Scenarios)\

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

Use Case Diagram Commands

Special Fields Available

Refer to the **Insert_Custom_Diagram** command described in this document.

Sub-reports

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

Diagram Commands

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.

\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	Optional	Specify whether to show differences on the shapes graphically on the diagram – True/False – Default:

on Diagram>>		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 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: 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\

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, it will generate the default diagram field of the current record. If the passed field contains invalid data, 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 the 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 larger size, the other parameter will be reduced to a proportionate scale. • If both parameters are specified, the image will still

		be proportionate. In this case, whichever is smaller (Width or Height), will be taken as a reference for scaling down 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: Scenario Diagram)\
\Insert_Diagram_Custom(a: Activity Diagram)\
\Insert_Diagram_Custom(a: Swimlane Diagram)\
\Insert_Diagram_Custom(a: Diagram)\

Examples

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\Fetch_Repository_Objects_by_Condition('BPD','' ,'Name')\
\scan(a)\

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

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

Fetch Sub-report Commands for Repository Objects and Tracking Items

Comments Sub-report Commands

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 you want to load the context of a comment, 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 available fields.

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 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_Repository_Objects_By_Condition(‘UC’)\
\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_Repository_Objects_By_Condition(‘UC’)\
\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_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 parameter 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_Repository_Objects_By_Condition('UC', ' "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\

Insert_Attachments

Compatibility: TopTeam Requirements Version 23.03 and above.

This command fetches all the attachments along with its contents of the record.
Note: Attachments as a Field.

This is the miscellaneous command. This is used under the primary command, to fetch the attachments of the record.

\Insert_Attachments ('<<Field Name>>')

Parameter

<<Field Name>>	Mandatory	Specify the field names as "Attachments" to fetch attachments.

Example

\Insert_Attachments(a : Attachments)\

Examples

\scan(a)\

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

Attachments

\Insert_Attachments(a : Attachments)\
\endscan\

Workflow Comments Sub-report Commands

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_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(' "Note" 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.
Note	This is the text or remark added for 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_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\

Fetch_Audit_Log_Detail

Compatibility: Desktop App Version 6.20 and above.

This command displays the Audit Log details, such as Old Values and their 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\]

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\

Traceability Sub-report Commands

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

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 'UC', 'UC' will appear in front of the identifier. For example, in the identifier UC-2348, UC 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, it will fetch linked records with 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, it will fetch linked records having links of any type.
<< Filter Condition>>	Optional	If Filter Condition is specified, it will fetch linked records that satisfy the specified filter condition. If Filter Condition is not specified, 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_Repository_Objects_By_Condition('UC')\
\ 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.

Variants Sub-Reports Commands

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	
Record Type	

Examples

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

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

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

Sample Template

\scan(a)\

\a:Name\

\Insert_Permalink(a: Disp 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\

Linked Issues Sub-report Commands

Fetch_Linked_Issues_By_Condition

Compatibility: Desktop App Version 3.35 and above.

This secondary command fetches Linked Tracking Items (Issues, CRs, Action Items, 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 before 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. Action Items, 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 before 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 descending order.

\Fetch_Tracking_Items('', 'Assigned to Me') \

This command fetches all Tracking Items Records for the specified filter Assigned to Me.

Test Management Commands

Fetch_Test_Case_Steps

Compatibility: TopTeam Requirements Version 23 and above.

This secondary command retrieves Test Case's steps.

\Fetch_Test_Case_Steps('<<Steps Field>>')

Parameters

<< Steps Field>>	Mandatory	This field contains Test Case 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

All columns in Steps Table, field name is same as column name except Step No.

Sample Template 1

\Fetch_Test_cases_By_Condition()\

Test Cases

\scan(a) \

\a:Name - [\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:Name - [\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_Results_In_Test_Run

Compatibility: TopTeam Requirements Version 23 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
Name	
Run Date	
Run By	
Result	
ID	
Defects	

Sample Template 1

\Set_Project('$CURRENT_PROJECT$')\

\Fetch_Test_Runs_By_Condition('')\
\scan(a)\

**\a : ID\ - \ a : Name \ - State: \a: State\ Result: \a: Result **

[\Insert_Permalink(a: ID)\]

Created On : \ a: Created On\
Priority: \a: Priority\
Start Dt: \a: Planned Start Date\
Run By: \a: Run by\

Target Release: \ a: Target Release \
Owner : \a: Owner\
End Date: \a: Planned End Date\

\Fetch_Test_Results_In_Test_Run(a:id)\

Executed	Name	Result	Run by
\scan(b)\ \if (! eof(b))\			
\ b : Executed\	\ b : Name \	\b: Result \	\ b: Run by\

\endif\ \endscan\\endscan\

Sample Template 2

Test Results

\Fetch_Test_Results_In_Test_Run('TR-4356')\

Contents

\a:Name\ 3

Test Results

Executed	Name	Result	Executed by	Execution dt	Defects
\scan(a)\ \if (! eof(a))\					
\a: Executed\	\ a : Name \	\a: Result \	\ a: Run by\	\ a: Run End Date\	\a: Defects\

\endif\

\endscan\

Fetch_Test_Runs_By_Condition

Compatibility: Desktop App Version 23 and above.

This primary command retrieves Test Runs in the system based on 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	
Name	
Assigned To	
Result	
Planned Start Date	
Planned End Date	

Run By	
Remarks	
Run Date	
Project	
Priority	
Stability	
State	
Updt By	
UPdt Dt	
Crt By	
Crt Dt	

Example

Fetch_Test_Runs_By_Condition (' "Result" = "Passed" ','Title')\

Sample Template 1

Test Runs

\Fetch_Test_Runs_By_Condition(' "State" = "Completed" ','Priority','TS-3401')\

Contents

\a:Name\ 3

Test Runs

ID	Name	Result	Owner	State

\scan(a)\

\if (! eof(a))\

\ a : ID\	\ a : Name \	\a: Result \	\ a: Owner\	\ a: State \

\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: Display Seq\	\if (b : Security Enforced = 'Y')\Y \endif\	\if(b : Versioned = 'Y')\Y\endif\	\ b : WBS Start No\	\if(b : 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**, 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 Role 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**, 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, 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**, 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 : Record Type\ \b: action\ \b:User Name\

\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
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

\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 has been granted System Administrator permission, 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 license 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 a specified user is a member and can be used within the 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\

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 a 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 \

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\

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.

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'\

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. When a document is generated, you can use the IF command to test if the Section has been turned ON.

**\if(<<Section_Name>>) **

The Fetch commands, Fields, etc., for this Section go here:

\endif

Examples

\if(Use_Cases) \

\Fetch_Repository_Objects_By_Condition('UC')\

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\

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 the second parameter
<<Font Size>>	Optional	Specify the Size of the Font as the 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 a 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 the second parameter
<<Font Size>>	Optional	Specify the Size of the Font as the 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_Repository_Object_By_Condition('BRULE','' ,'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

\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_Repository_Objects_By_Condition('UC','' ,'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')\

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.

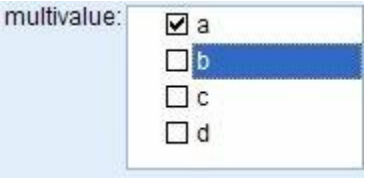
**\ Is_Value_Selected(<<CheckList Field>>, '<<One Value from the List>>') **

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_Repository_Objects_By_Condition('UC','' ,'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 document user can 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 document user can 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 a 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_Repository_Objects_By_Condition('UC','' ,'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 command will give the 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 >>')

Compatibility: Version 19.11 onwards **- parameter << URL Text>> - provision for new value <<SHOW_URL>>.**

Parameters

<<ID>>	Mandatory	The record identifier can be **ID_With** or **Without_Prefix**. This parameter can be used for both Repository Objects and Tracking Items and can be either 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. Use new value **<<SHOW_URL>>** to display actual record URL as link URL text.
<< Is_WIN_URL >>	Optional	Allows 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	By default, it 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 : Disp ID)\

\Insert_Permalink (a : Disp ID, a : Title)\

\Insert_Permalink(a : Disp 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 : Disp ID)

\Fetch_Repository_Objects_By_Condition('UC', '' ,'Name')\

\scan(a)\

\a:id\ \a:Name\

\Insert_Permalink (a : Disp ID)\

\endscan\

Items ID within a Scan Loop

Insert_Permalink (a : Disp ID)

\Fetch_Tracking_Items_By_Condition('', '', 'Id')\

\scan(a)\

\a:id\ \a:Title\

\Insert_Permalink (a : Disp 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 command will point to the bookmark of a record ID/link in the output document. If no bookmark is specified for a record ID/link, 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 : Disp ID)
\Fetch_Repository_Objects_By_Condition('UC','' ,'Name')\
\scan(a)\
\Insert_Bookmark_For_Record(a : ID)\\a:Name\
\Insert_Permalink (a : Disp ID)\
\endscan\

Sample Template 2

\ PROJECT_NAME \

Use Cases

\scan(a)\\if (! EOF (a))\

\a:Name\

\Insert_Permalink(a: Disp 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: Disp 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 a formatted record title in the output document based on 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 based on 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('$CURRENT_PROJECT$')\

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/OneView Section*, 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('$CURRENT_PROJECT$')\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\
\scan(a) \
[\ a: Id\]-\ a : Name \
\Insert_Record_Location(a : ID)\
\endscan\

Insert_Linked_Records_Field

Compatibility: Desktop App Version 21.00 and above.

This command inserts a web URL link for records or artifacts in a *Referential, Linked Records, Backlog* and *Sprints* fields. It is a miscellaneous command and can be used inside Scan-Endscan.

\Insert_Linked_Records('<<Linked_Record_Field>>' ,'<< Is_Show_On_New_Line>>', '<<Is_Show_As_Text >>')

Parameters

<<Linked_Record_Field>>	Mandatory	This parameter supports *Linked Records* field. If the specified **<<** Linked_Record_Field **>>** parameter is not a linked field, this command will generate an error.
<<Is_Show_On_New_Line>>	Optional	By default, it shows artifacts in the relational fields as comma-separated text. If set to TRUE, it will display each artifact in the linked field on a new line.
<<Is_Show_As_Text >>	Optional	By default, it show artifacts in the linked field as web URLs. Set to TRUE to display field data as simple plain text.

Examples

Use within scan loop

Insert Referential Field

\Linked Record Field (a : RFMV2)\

Insert MULTIPLE Referential Field - Comma Sep, As Text

\Insert_Linked_Records (a : RFMV2, False,True)\

Insert MULTIPLE Referential Field - New Line, As Text

\Insert_Linked_Records (a : RFMV2, True, True)\

Insert MULTIPLE Referential Field - Comma Sep, As Link

\Insert_Linked_Records (a : RFMV2)\

Insert MULTIPLE Referential Field - New Line, As Link

\Insert_Linked_Records (a : RFMV2, True)\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\scan(a) \

\a:Name\

\a: Record Type\, \Insert_Permalink(a : Disp ID)\

RFSingle-V2 – Field Text
\a: RFSV2\

RFSingle-V2 – UDF- Link

\Insert_Linked_Records (a : RFSV2)\

RFMultiple-V2 - Field Text
\a: RFMV2\

Insert MULTIPLE Relational Field - New Line , As Text

\Insert_Linked_Records (a : RFMV2, True, True)\

Insert MULTIPLE Relational Field - Comma Sep, As Link

\Insert_Linked_Records (a : RFMV2)\
\endscan\

Insert_External_Artifact_URL

Compatibility: Desktop App Version 21.00 and above.

This command inserts a web URL link for an external URL in the URL field. It is a miscellaneous command and can be used inside Scan-Endscan.

\Insert_External_Artifact_URL('<<External_URL_Field>>' ,'<<Is_Show_As_Text >>')

Parameters

<<External_URL_Field>>	Mandatory	This parameter supports only External URL Field. If the specified **<<** External_URL_Field **>>** parameter is not a external field, this command will generate an error.
<<Is_Show_As_Text >>	Optional	By default, it shows the external link in the URI field as a WEB URL. Set to TRUE to display field data as simple plain text.

Examples

Use within scan loop

Insert External_URL Field

\Insert_External_Artifact_URL (a : URI)\

Sample Template

\Set_Project('$CURRENT_PROJECT$')\

\ PROJECT_NAME \

\scan(a) \

\a:Name\

\a: Record Type\, \Insert_Permalink(a : Disp ID)\

External URL field - As Link

\Insert_External_Artifact_URL (a : URI)\

Insert External_URL Field - As Text

\Insert_External_Artifact_URL (a : URI, True)\

External URL field - As Link
\Insert_External_Artifact_URL (a : Name)**//caption will be used from Name**

\endscan\

Insert_Component_Location

Compatibility: Desktop App Version 16 and above.

This command inserts records owner component's name, path, or location in the output document. It is a miscellaneous command that can be used inside the Scan-Endscan loop.

\Insert_Component_Name('<<ID Field Name>>')
\Insert_Component_Path('<<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_Component_Name(a : ID)\
\Insert_Component_Path(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_Components_Path(a : ID)\
\Insert_Components_Name(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 Path >>	Optional	If the user name/image path is blank, the logged-in user's image will display by default.
<<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 image's actual size 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 the Scan-Endscan loop.

\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 this command's short form, 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 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 a template is not available for current Record Type, 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', '')\
\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 the Scan-Endscan loop.

\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 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 fetches 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 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 fetches 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 their 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 generates 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 the desired Width of the diagram to display in the output.
<<Height>>	Optional	Set the desired height of the diagram to display 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 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')\

Fetch_Record_Fields

Compatibility: Desktop App Version 16 and above.

This command is used to insert record-specific fields for a particular record type. It is a miscellaneous command that can be used inside a scan-endscan loop.

Fetch_Record_Fields('<<ID Field Name>>')

Parameters

<< ID Field Name>>	Mandatory	Specify the ID field to insert record-specific fields for that record type. If this parameter is not specified, the command will not work.

Fields Available

No fields are available for this command.

Examples

\Fetch_Record_Fields(a : ID)\

Example

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\

\scan(a) \

**[\ a: Id\]-\ a : Name **

\Fetch_Record_Fields(a: Id)\ \LisFieldDiagram := 'N'\\scan(b)\\if (! eof(b))\\LisFieldEmpty := 'Y'\\if(b : Type = 'RTF')\\if (Is_Field_Non_Empty (b : ValueRtf))\\LisFieldEmpty := 'N'\\endif\\endif\
\LIsPrintValue:= True\\LCaption := ''\\if(b: IsShowCaption = 'Y')\\LCaption := Trim(b: Field)\\endif\

\if(b : IsHeader = 'Y' && LCaption = 'Record Information')\\LCaption := ''\\endif\\if(b: Field = 'ID')\\LIsPrintValue:= False\\endif\\if(b: Field = 'Name')\\LIsPrintValue:= False\\endif\\if(LIsPrintValue)\
\if(b : Type = 'Diagram' && LCaption = '') \\LisFieldDiagram := 'Y'\

\Insert_Diagram_Custom(b:ValueText,5,7,'INCHES','JPEG')\

\elsif(b : Type = 'Diagram' && LCaption <> '')\\LisFieldDiagram := 'Y'\

\LCaption\ \Insert_Diagram_Custom(b:ValueText,5,7,'INCHES','JPEG')\

\elsif(b : Type = 'RTF' && LisFieldEmpty = 'N' && LCaption = '')\

\LCaption\ \Insert_Rich_Text_Using_Word(b: ValueRtf)\

\elsif(b : Type = 'RTF' && LisFieldEmpty = 'N')\

\LCaption\ \Insert_Rich_Text_Using_Word(b: ValueRtf)\

\elsif(b : Type = 'Actors')\

Actors \Execute_Template_For_Id(a:ID, '', 'Form Layout Use Case Actors Report')\

\elsif(b : IsHeader = 'Y' && (LCaption = 'Record Information' || b : ValueText = 'Record Information'))\
\elsif(b : IsHeader = 'Y' && LCaption <> '')\

\LCaption

\elsif(b : ValueText != '' && LCaption <> '')\

\LCaption\ \b : ValueText\

\elsif(b : ValueText != '')\

\LCaption\ \b : ValueText\

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

\endscan\

Fetch_Report_Section

Compatibility: Desktop App Version 16 and above.

This command is used to insert records sub-tab sections in the output. It is a miscellaneous command that can be used inside a scan-endscan loop.

Fetch_Report_Sections('<<ID Field Name>>')

Parameters

<< ID Field Name>>	Mandatory	The parameter is needed for working of the command. It requires ID field as a parameter.

Fields Available

No fields are available for this command.

Examples

\Fetch_Report_Sections(a : ID)\

Example

\Set_Project('$CURRENT_PROJECT$')\

\PROJECT_NAME\

Use Cases

\Fetch_Repository_objects_by_Condition('UC')\

\scan(a) \

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

\Fetch_Report_Sections(a: Id)\
\scan(b)\\if(!eof(b))\\IsHeader := 'True'\\Attachments := 'False'\\Linked_Issues := 'False'\\Traced_Records := 'False'\\Comments:= 'False'\\Links:= 'False'\\Approvals := 'False'\\ Version_History := 'False'\\ Branches := 'False'\\ Workflow_History := 'False'\\Audit_Log_History := 'False'\

\if(b : ValueText = 'Attachments')\\Attachments := 'True'\\endif\\if(b : ValueText = 'Comments')\\Comments := 'True'\\endif\\if(b : ValueText = 'Links')\\Links := 'True'\\endif\\if(b : ValueText = 'Traced_Records')\\Traced_Records := 'True'\\endif\\if(b : ValueText = 'Linked_Issues')\\Linked_Issues := 'True'\\endif\\if(b : ValueText = 'Approvals')\\ Approvals := 'True'\\endif\
\if(b : ValueText = 'Version_History')\\ Version_History := 'True'\\endif\\if(b : ValueText = 'Branches')\\ Branches := 'True'\\endif\\if(b : ValueText = 'Workflow_History')\\ Workflow_History := 'True'\\endif\\if(b : ValueText = 'Audit_Log_History')\\ Audit_Log_History := 'True'\\endif\
\If(Comments)\\Fetch_Comments_By_Condition('','')\\if(!eof(c))\\LIsSubReportSelected := 'Y'\

Comments

\scan(c)\
\if(!eof(c))\
\if(c : Indentation Level = 1)\


\c : Person\, \c : Date\
\c : Comment\

\else\


\c : Person\, \c : Date\
\c : Comment\

\endif\\endif\\endscan\\endif\\endif\\If(Attachments)\\Fetch_Attachments_By_Condition('', 'File name')\\if(! eof(c))\\LIsSubReportSelected := 'Y'\

Attachments

File Name	Person	Added on
\Insert_Attachment(c : Filename, 'Tru \ c : File name \	\c : Person \	\c:Added on \

e')\		

\endif\\endscan\\endif\\endif

\endif\\endscan\

\endscan\

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, it will display the 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_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**, 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')\
\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>>') **
\ Insert_Field_As_Diagram_In_Inches(<<Diagram Field>>, '<<Width>>', '<<Height>>')

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.

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')\
\Insert_Field_As_Diagram_In_CMs(b:Value2,'2','6')\
\Insert_Field_As_Diagram_In_CMs(b:Value1,'10','15')\

\Insert_Field_As_Diagram_In_Inches(b:Value2,'12','9')\
\Insert_Field_As_Diagram_In_Inches(b:Value1,'','')\

Declare Variable

VAR

Compatibility: Desktop App Version 3.00 and above.

This command creates Variables that 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 opens a variable prompt window where you can specify value of that variable. 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.

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',)\
\Declare_Variable('Filter_Name', 'Text', 'Enter Filter Name to filter Requirement records', '')\
\Declare_Variable('WBS_Code', 'Float', 'Enter Wbs Code for Requirement records', '1')\
\Declare_Variable('Show_Parent', 'Flag', 'Check Show Parent option to display Header records and child record satisfies the filter condition', 'False')\

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.

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_Action_Item,'AITM', '', 'Set Filter for Action Item list')\
\Declare_Variable_Filter('Filter_Tree_Action_Item','AITM', 'True', 'Set Filter for Action Item 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 ')\
\VAR(LVariable_Filter) \

Examples

\ LVariable_Filter := Filter_Value \
\Fetch_Use_Cases(LVariable_Filter)\
\scan(a) \

\a:Name\ [\a:id\]

\endscan\

Declare_Variable_ID

Compatibility: Desktop App Version 5.x and above.

This command creates variables along with their values at runtime.

Declare_Variable_ID: Command is used to create Variables.

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 '*' before the Variable for the Mandatory Value selection. Set **FALSE** to hide '*' before the Variable for the Optional Value selection. Set **TRUE** to display '*' before 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)\

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\

Insert_Custom_Variable

Compatibility: Desktop App Version 8.15 and above.

This command inserts a Custom Variable.
This command 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', '$CURRENT_PROJECT$')\

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')\
\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_Package_Approvals_By_Condition

Compatibility: Desktop App Version 21.50 and above.

This command is used to fetch approvals created for the current master package artifact.

\Fetch_Package_Approvals_By_Condition('<<Filter_Condition>>', '<<Sort_Order>>')

Parameters

<<Filter_Condition>>	Optional	Filter condition based on the required values of an artifact. If not specified, all artifacts will be fetched.
<<Sort_Order>>	Optional	Specify the field names by which you want the results to be sorted. If not specified, the data will be is sorted by default based on baseline and create date.

Fields Available

- Reviewer/Approver
- Review Status/Approval Status
- Finish Date/Review Date
- Digital Signature Verified
- Request Date
- Due Date
- ID
- Title
- Type
- Baseline
- Review Gate
- Requested By
- Modified By
- Modified On

For Progress Bar

- Pending - \b: Records Pending\
- Approved - \b: Records Approved\
- Needs Revision - \b: Records Needs Revision\
- Abstained - \b: Records Abstained\

Examples

\Fetch_Package_Approvals_By_Condition()\
\Fetch_Package_Approvals_By_Condition(' "Reviewer" = "Steve" ')\
\Fetch_Package_Approvals_By_Condition('','Finish Date')\
\Fetch_Package_Approvals_By_Condition(' "Review Status" = "Approved" ','Finish Date')\

Examples

Review Packages

\scan(a) [,page] \\if (! eof(a))\

\a:Name\

Id: \ a: Id\

Description
\ InsertRtf(a : Description)\

\VAR(LastBaseline)\\LastBaseline:=''\\Fetch_Package_Approvals_By_Condition('', '')\\if (! eof(b))\

Package Approvals History

Reviewer	Review Progress	Review Status	Finish Date	Digital Signature Verified	Due Date

\scan(b)\
\if (! eof(b))\
\if (LastBaseline <> b : Baseline)\

Baseline: \b : Baseline\ (**Review Gate**: \b : Review Gate\)

\endif\

Reviewer: \b: Reviewer \	Pending - \b: Records Pending\ Approved - \b: Records Approved\ Needs Revision - \b: Records Needs Revision\ Abstained - \b: Records Abstained\	Review Status: \b: Review Status \	Finish Date: \b : Finish Date\ Review Date: \b : Review Date\ Request Date: \b : Request Date\	\b : Digital Signature Verified\	Due Date: \b : Due Date \ Id: \b : Id\ Title: \b : Title\ Type: \b : Type\ Requested By: \b : Requested By \ Modified By: \b : Modified By \ Modified On: \b : Modified On \

\SET(LastBaseline, b : Baseline)\\endif\\endscan\\endif\

\endif\

\endscan\

Sample Output:

Package Approvals History

Reviewer	Review Progress	Review Status	Finish Date	Digital Signature Verified	Due Date
Baseline: R01 - Business Requirements Document (**Review Gate**: Formal Review)					

Steve	Pending - **25** Approved - **10** Needs Revision - **2** Abstained - **0**	Finished	6/9/2021 4:17:09 PM	Y	10/1/2021 12:00:26 AM
Baseline: R01 - Business Requirements Document - [0] (**Review Gate**: Formal Review)					
Tom	Pending - **28** Approved - **9** Needs Revision - **1** Abstained - **0**	Pending	Finish Date:	N	10/1/2021 12:00:26 AM

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')\
\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’)\
\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')\
\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')\
\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')\

\ *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')\
\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 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	
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')\

\ *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')\	\ a : Version \

	\scan(b)\ \if (! EOF(b))\ • **\ b : Date ** \ b : Person \ \ b : Comment \ \endif\ \endscan\	

\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\

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, 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, 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).
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).
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).
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_Repository_Object_By_Condition('BRULE',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_Rpository_Object_By_Condition('BRULE', 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 the specified Date.

\MONTH(date)

Parameter

<<date>>	Mandatory	

DAY

Compatibility: Desktop App Version 3.35 and above.

The Day command returns the Day of the 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 the first digit. 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 afternoon. 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 afternoon. 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 afternoon.
/	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 the 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, 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 part of the String which is searched from within the Source String '**S**'.

<<S>>	Mandatory	This is the Source String, from 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 controls 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_Repository_Objects_By_Condition('UC',' "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 do 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	