GENERIC TOOL FOR AUTOMATION USER GUIDE

AIRCRAFT APPLICABILITY	All
ATA APPLICABILITY	All
CONFIDENTIALITY	Company Use Only
DOCUMENT LEVEL	2

SUMMARY:

The Generic Tool for Automation (GTA) allows the creation, modification, execution and results reporting of test Procedures on ground Test Means.

The Tool has an Editor module, a Scheduler module and modules to handle Lab Test Means (Ex. Image recognition).

The objective of this document is to support the usage of the GTA Tool Editor Module.

	NAME	**SIGLUM - FUNCTION**	**DATE**	**SIGNATURE**
AUTHOR(S)	Fy Ny Aina ANDRIANONY Adrien CHARLES Lionel EON	SSG on behalf of Airbus	28 February 2025	
APPROVAL	PASIAN Anthony	1VMDY – Test Means Developer	28 February 2025	
AUTHORIZATION	PASIAN Anthony	1VMDY – Test Means Developer	28 February 2025	

LIST OF DISTRIBUTION

DEPARTMENT/ COMPANY	***NAME***	***P.O. BOX***	***COVER PAGE ONLY***	***NOTE WITHOUT ATTACHMENT***	***NOTE WITH ATTACHMENT***
1VMIA	Medhi DEMEULENAERE				
1VVAM	Wilfried TECOUERES				
1YYSW	Louis ANDREA SALBAN				
NO. OF COPIES			***0***	***0***	***3***

RECORD OF REVISIONS

ISSUE	DATE	EFFECT ON		REASONS FOR REVISION
		PAGE	PARA	
1.0	01/12/2015	All	All	Creation of the document
2.0	05/02/2016	N/A	N/A	Scrum 12.5 Updates: ✔ 3.4.2.1 – One Click Launch Application /Pin Programming updates. ✔ 3.4.2.7 – VMAC release parameter list prepared from already used VMAC parameters in the procedure. ✔ 3.4.2.4.3 – 'Check Audio Alarm' added for Condition command. ✔ 3.4.2.10 – Confirmation time for Check & Conditions. ✔ 3.4.2.10.2 – Added color input for Check FWS message. ✔ 3.6.1 – 'Find & Replace' ✔ 3.6.2 – 'Add to Favorites' option ✔ 3.6.3 – 'Export' Option ✔ 3.6.4 – 'Dump List' Option ✔ 3.6.5 – 'SVN' Option ✔ 3.6.6 – 'Show/Hide Toolbar menu option'
3.0	04/03/2016	N/A	N/A	Scrum13.5 Updates ✔ State names are changed from s0 to action name. ✔ "Confirmation Time" state is removed from SCXML. ✔ "Wakeup Launcher" state is removed from SCXML. ✔ One Click Stop Application Enhancements/Modifications as requested. ✔ "Print Value" command is renamed to "Print" ✔ 3.3 – "Fav" Option, add to Favorite ✔ 3.3 – new Element options in context menu ✔ 3.4 – "Dump List" column in Editor ✔ 3.4.2.1 – "One Click" Stop Application ✔ 3.4.2.12 – Robotic Arm Command

ISSUE	DATE	EFFECT ON		REASONS FOR REVISION
		PAGE	PARA	
4.0	15/04/2016	N/A	N/A	Scrum14.5 Updates Command Changes: ✔ 3.4.2.10.1 – CHECK ONLY VALUE ✔ 3.4.2.11 – Set Command – Added FS_NO_CHANGE ✔ 3.4.2.13 – Print Table ✔ 3.4.2.14 – F/CTL Kinematic Multi Output ✔ 3.4.2.1.2 – One Click Application o Added Stop application and Start Application tick boxes. ✔ 3.4.2.1.3 – One Click Pin Prog. o Added Start and Stop options. ✔ 3.4.2.1.1 – One Click Environment o One Click HMI Battery related changes. Result View Page Related Changes: ✔ 3.5 – Dump list is added in the Result View ✔ 3.5 – Get Last Execution Report. Other Changes: ✔ Configuration: Debug Mode Added to detail the logging of parameters ✔ Create Sequence from element browser. ✔ 3.4.1 – Show Clipboard. ✔ 3.4.1 – FORCE DB update ✔ 3.6 – Added More Description on SVN with GTA.
5.0	10/05/2016	N/A	N/A	Scrum15 Updates Command Changes: ✔ 3.4.2.15 – Generic Command ✔ 3.4.2.4 – Do –While Condition. ✔ 3.4.2.1.4 – One Click Title. Result View Page Related Changes: ✔ 3.5 – Added Check boxes to each command for export selection. Other Changes: ✔ “Check FWC Warning” confirmation time ✔ Added transitions in nested states. ✔ Added nested states in one click. ✔ Check compatibility: DB Parameter not present in Database check. ✔ New Attribute in SCXML data model for VMAC parameters “Actor”.
6.0	31/05/2016	N/A	N/A	Scurm16 Updates Command Changes ✔ 3.4.2.16 – Maths Command
7.0	04/11/16	N/A	N/A	Scrum 18 Updates Read only feature

ISSUE	DATE	EFFECT ON		REASONS FOR REVISION
		PAGE	PARA	
8.0	10/01/19	N/A	N/A	Scrum 24 Updates Find and replace Session management Indentation Switch to edit mode
9.0	29/04/2019	N/A	N/A	Scrum 25 Updates Delta difference between two commands Selective execution of steps Quick access menu to action/check commands Scheduler plugin Debug mode Printing a procedure
10.0	29/07/2019	N/A	N/A	Scrum 26 Updates Document Viewer Dashboard Updating for managing usable documents. Application Log Window to check basic application logs
11.0	10/09/2021	N/A	N/A	Get the User Guide (PDF format) Global Update since 23/11/2020 Manage For Each command, Typing variables, New features. Update picture of the GTA Editor
12.0	08/06/2022	N/A	N/A	Global update Modification table of contents Architecture of the document (red line) Add the current release notes (30.04.03)
13.0	16/01/2023	N/A	N/A	Split view / procedure editor
14.0	22/05/2023	N/A	N/A	GUI Clean up Terminal commands
15.0	30/06/2023	N/A	N/A	Csv to procedure conversion
16.0	28/08/2023	N/A	N/A	Test Report in csv format
17.0	27/10/2023	N/A	N/A	Enable Subscribe/Unsubscribe Enable Manual Action Find/Replace Equipment Name Minor Updates on the Equipment Settings
18.0	26/01/2024	N/A	N/A	Git integration section Updated icons picture to include git icons Possibility to use Equations from a Foreach loop Possibility to Set FS (SSM) from a Foreach loop Possibility to use ManAct from a Foreach Loop
19.0	29/02/2024	N/A	N/A	SVN options removal (Decommissioned)
20.0	29/03/2024	N/A	N/A	External action for SET/IF and Git config
21.0	12/06/2024	N/A	N/A	Addition of SET command with MCDU request
22.0	03/07/2024	N/A	N/A	Addition of Version Tools Bench information
23.0	09/08/2024	N/A	N/A	Addition LTRA Purpose and conclusion Update Enable/Disable Unsubscribe Addition GTA Controller dashboard Update Header customization

ISSUE	DATE	EFFECT ON		REASONS FOR REVISION
		PAGE	PARA	
24.0	04/09/2024	N/A	N/A	Update for Show dependencies Update Header customization Update GTA path settings Update Git icon Update Figure Architecture
25.0	03/10/2024	N/A	N/A	Update Procedure Dashboard Launcher Update Export .pro to xml file Update Path configuration settings
26.0	04/11/2024	N/A	N/A	Update Result view - Purpose and conclusion Update Git
27.0	06/12/2024	N/A	N/A	Update for Equation Update WriteBack function Update procedure export function
28.0	31/12/2024	N/A	N/A	Addition of “Know limitations” § Addition of recommendations for fill csv file
29.0	31/01//2025	N/A	N/A	Addition Log window view Addition warning for equation Addition reserved words Addition of result condition
30.0	28/02/2025	N/A	N/A	Addition of call an external tool command on a Check KO or OK Update keyword for MCDU button

TABLE OF CONTENTS

1. Introduction

The GTA (Generic Tool for Automation) aims to allow the creation, modification and execution of test Procedures on ground test means, by piloting various LTM Tools.

The architecture of the tool is based on a set of several modules with dedicated functions. All the modules will be piloted by a unique graphical user interface, the GTA editor. The editor allows user to:

- Choose the test rig that will execute the automatic procedure (Test Configuration).
- Create, configure, modify or import one or several actions to be performed on the test rig.
- Save these actions as a test procedure in an intermediate file (SCXML).
- Reuse the previously saved procedures/objects/functions in a new test procedure/sequence.
- Execute the test procedure in the targeted test rig.
- Analysis of test procedure execution results.

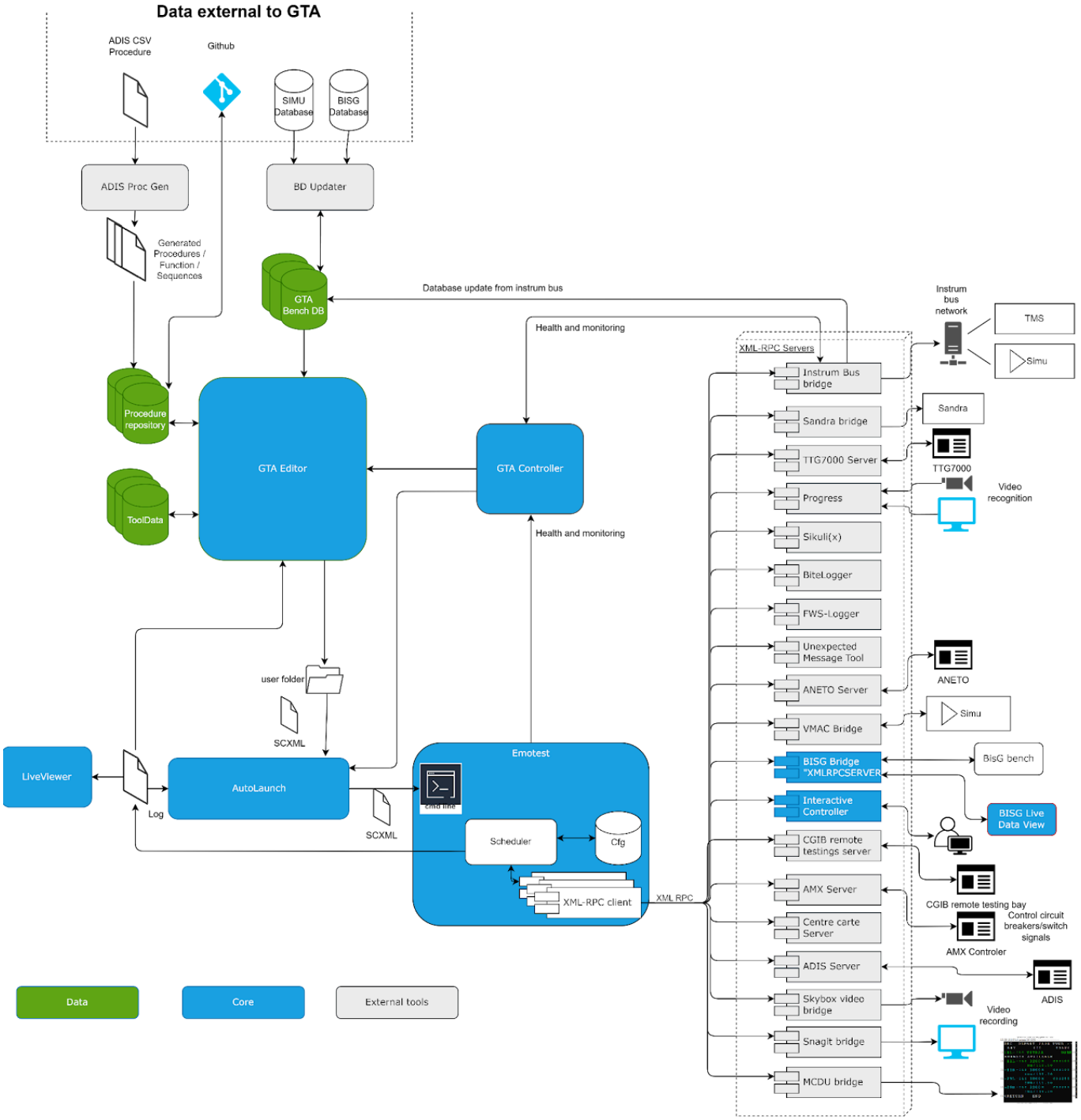


Figure: GTA architecture overview

This user guide defines the terms of use of the GTA editor to achieve automatic tests.

1.1.Reference documents

N°	Name
[R1]	LR89RQ1501173_PTS_Generic_tool_for_Automation_WP2-ed-2.0.docx
[R2]	RTF ATAXX Tests Automation X73RQ1302692_V3 0.docm

1.2. Terminology – Abbreviations

Term	Definition
BISG	Banc d'Intégration Système Générique. GSIB in english (Generic System Integration Bench)
BITE	Built In Test Equipment
BDW	Warning DataBase
CSV	Comma-Separated Values
EMOTEST	Name of GTA Scheduler Module
FWC	Flight Warning Computer
FS	Functional Status
GTA	Generic Tool for Automation
HMI	Human Machine Interface
IC	Interactive Controller
ICD	Interface Control Document
IRT	Image Recognition Tool
LTRA	Lab Test Report Analyses
LTM	Laboratory Test Means
MCDU	Multifunction Control Display Unit
PIR	Parameter Internal
PMR	Parameter Model on Request
SCXML	State Chart eXtensible Markup Language
SFS	Shared File System
VMAC	Versatile Monitor And Command Protocol

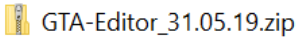
1.3. Known limitations

Allowed characters in procedure see here
Forbidden characters in Manual action see here
Forbidden characters in Print message see here
Forbidden characters in Title command see here
Limitation about cell content in a CSV see here

2. GTA Editor Setup

2.1. Folder Structure

The software is encapsulated in a folder **GTA-Editor_Vxxx** (xxx = version number), and consists of:



For example older "**GTA-Editor_31.05.19.zip**" consists of:

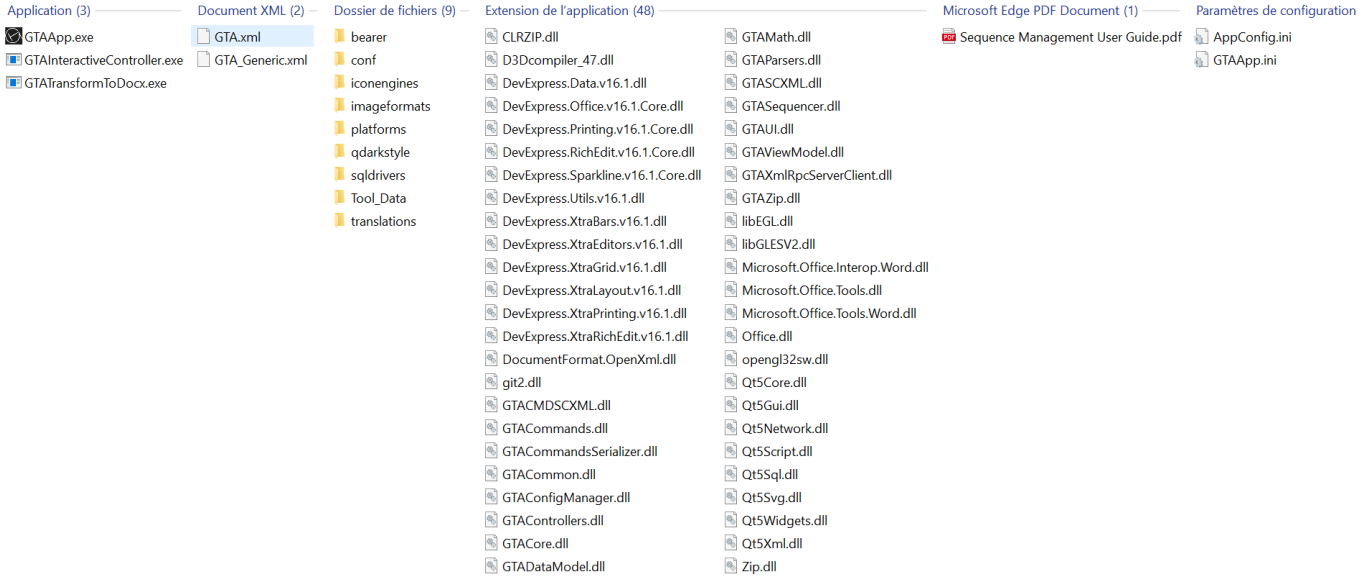


Figure: GTA Editor Folder Structure

NB: The GTA Editor software is included in the GTA Studio package delivery which is the new installer for all the GTA ecosystem softwares.

2.2. Start GTA Editor

Following steps define the sequence to follow to open the GTA Editor HMI:

STEP 1: The folder where the executable is located should contain a configuration file for the application (AppConfig.ini). This file contains the folder of one or several bench configuration files and the name of a default bench configuration file. GTA works with and without an argument, which is the bench configuration file. If the bench configuration file does not exist or is not valid, the following window appears.It is mandatory for the user to enter the tool configuration settings as shown in sample below:

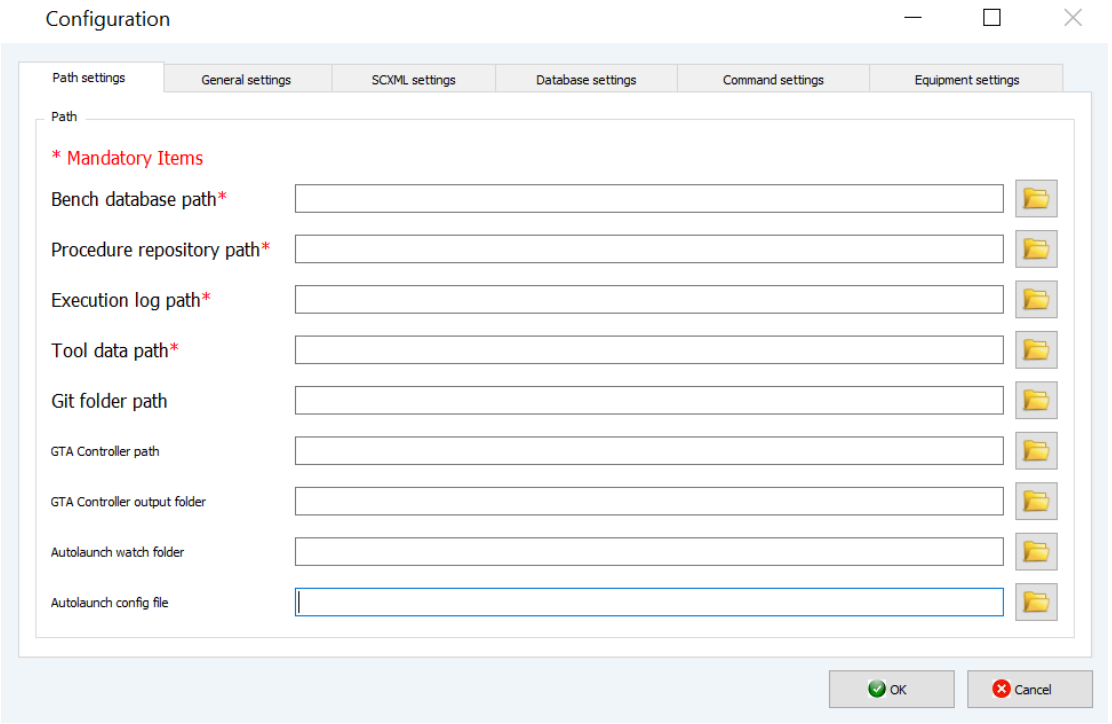


Figure: Configuration path settings

STEP_2: GTA automatically creates the following folders in the bench database path:

Figure: Database repository

STEP_3: GTA automatically creates the following folders in the tool repository path:

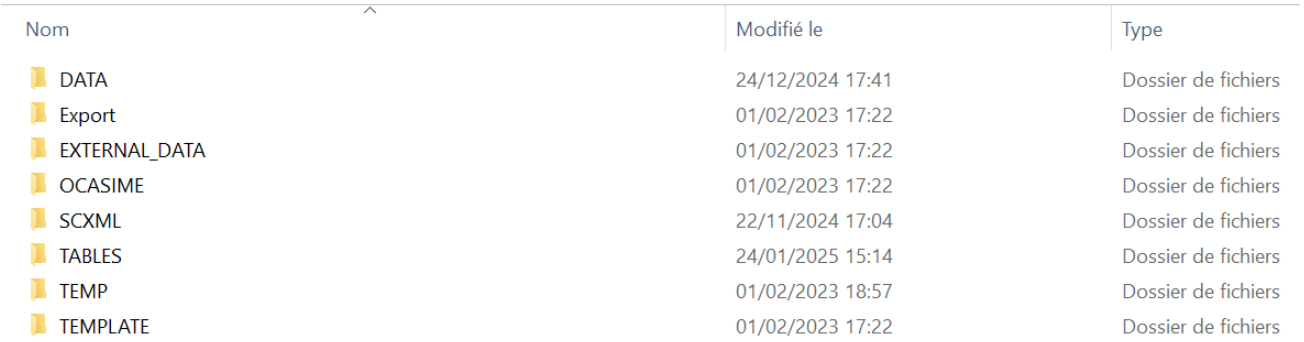
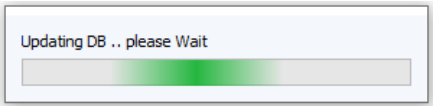


Figure: Tool repository

STEP_4: User needs to explicitly put appropriate contents into the folders at the bench database path.

STEP_5: Once the user provides the bench database files, the user just needs to press the ***right*** button on the editor to allow the tool to build entirely the associated database for its use.

The complete database creation may take a long time the first time; use the left button to update the database afterwards.



Note: The *left* button performs a partial database update, based on any file modification or addition.
Note: The *right* button performs a full database rebuild.
Note: Tool configuration settings can be updated anytime to provide a new set of bench database and repository locations for the GTA tool to work from.

GTA Editor is now ready for use!

2.3. Databases of GTA

During the execution of GTA Editor, users will manage several databases. On the one hand, the database contains the file architecture of the DATA folder (Workspace). On the other hand, the database is built to contain all the AC_Parameter (Bench).

2.3.1. Work_db

Once the Workspace (Procedure repository) path is defined for the first time, a database file “Work_db” is going to be created and will contain the file architecture of the DATA folder that is considered as the “root”.

In this file, are stored different and useful information handling to build/rebuild the Tree View of the Document Viewer, if needed.

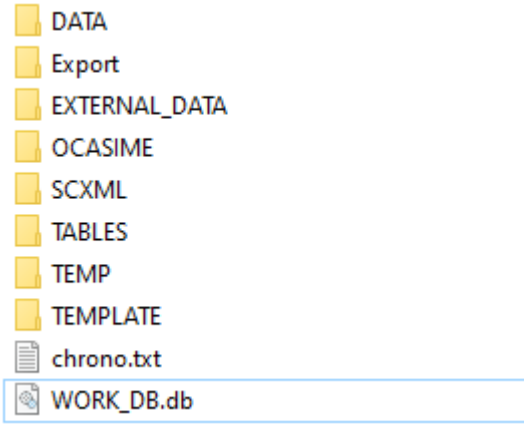


Figure: Database repository

In this figure, the content of the “WORK_DB” file.

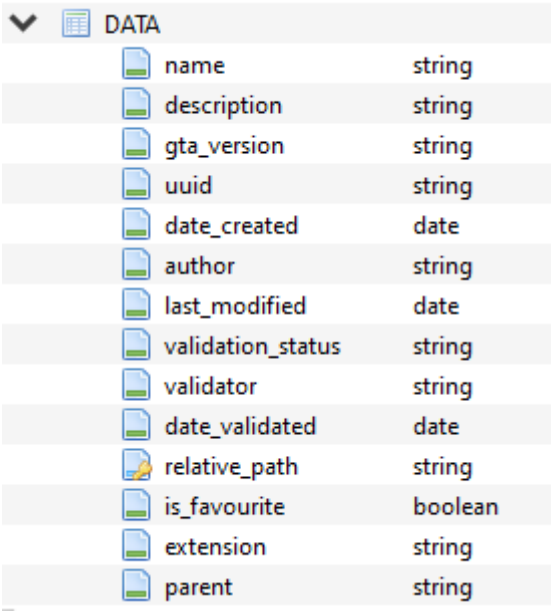


Figure: Work DB file content

2.3.2. Parameter_db

Once the Bench (Database) path defined for the first time, a database file “Parameter_db” is going to be created and will contain all the Aircraft Parameters provided by the different types of file such as *.CSV, *.PIR, *.XML.

In this file, are stored the useful information about Aircraft Parameters. These data are specific to users and can be used through GTA Editor to create procedure files.

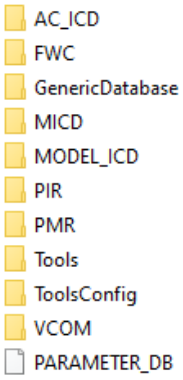


Figure: Database repository

In the following figure, the content of the “PARAMETER_DB” file.

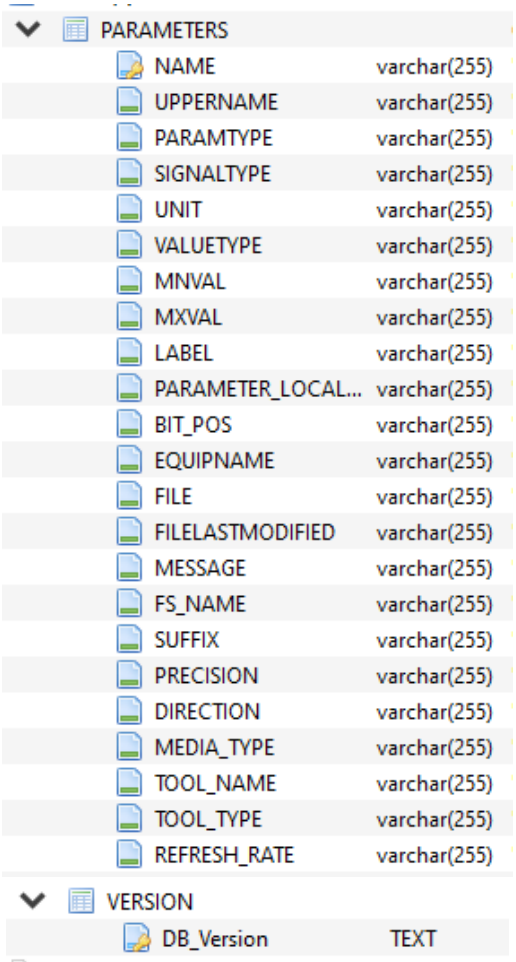


Figure: Parameter DB file content

3. GTA Editor

3.1. Basic Information

3.1.1. GTA main sections

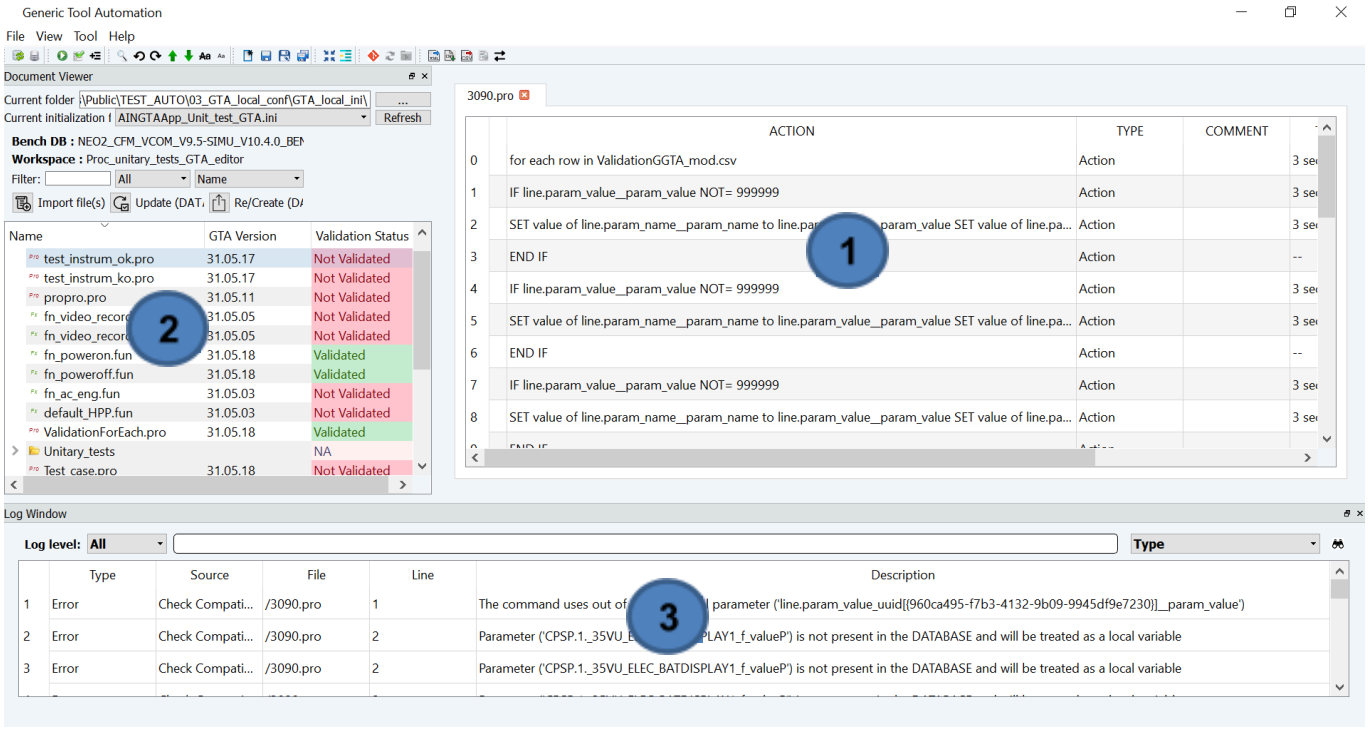


Figure: Editor View

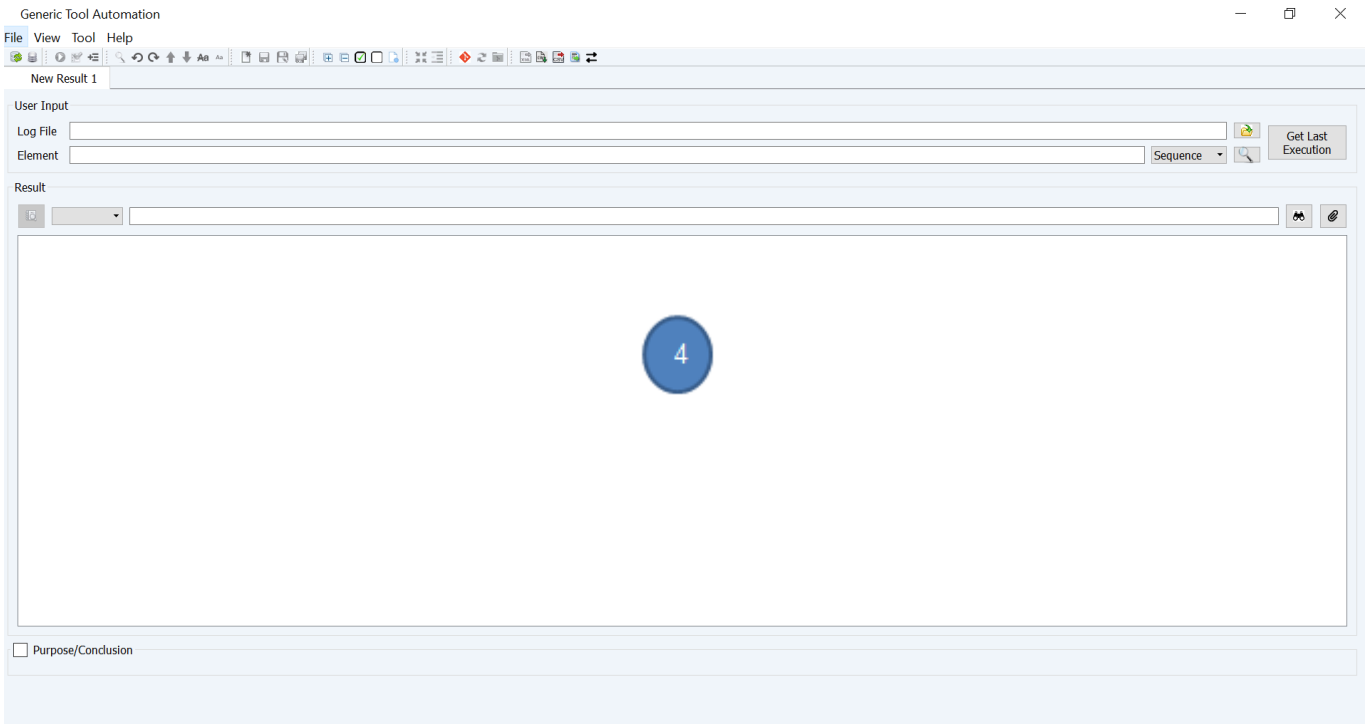


Figure: Result View

The GTA Editor has 4 main sections: (Also shown in figure above)

1. **Editor View** – The editor view comprises multiple editors. The user can open multiple documents and edit them.

2. **Document Viewer** – Provides an element browser to easily search and select specific Sequence/Procedure/Function and Object. The user can also select another bench configuration (*.ini) file, either by selecting one in the current folder (combo box) or by selecting another one from a different folder (browse button).

3. **Log Viewer –** All kinds of information and error logs will be displayed in this view.

4. **Result Viewer –** Allows the analyses of test procedure/sequence execution results.

3.1.2. GTA Elements

Definitions of GTA Elements:

✔ **Object**: Group of basic steps that primarily includes signal acquisition and generations.

✔ **Function**: Similar to an object but may also include conditional statements and parameters that contain placeholders which get replaced with values during function invocation. (Function may include call to other functions and objects)

✔ **Procedure**: Series of test steps which may include calls to functions and objects. (No specific restrictions for a procedure definition)

✔ **Sequence**: Ideally, a sequence is a collection of test procedures that needs to be executed back to back. However, GTA Editor also allows calls to Functions, Objects and conditional statements to enable creation of a dynamic sequence.

3.1.3. Allowed characters

Please be aware that only the following special characters are properly handled by GTA Ecosystem :

<	>	<=	>=	=	.	$	@	#	-	?

The default language of GTA is English therefore any other characters may not be compatible.

The users should consider that :
- ➔ an element name must not contain any accents
- ➔ use preferentially underscore “_” instead of dot “.”, space “ “ or dash “-” as separator

3.2. Configuration Menu

GTA Editor provides a tool menu, available from Tool > Configuration.

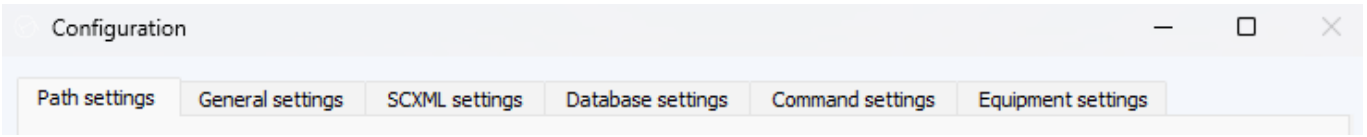


Figure: Configuration window

We are going to explain each tab and their role.

3.2.1. Path Settings

This tab contains the main paths to be configured. The Bench database and the Workspace paths are mandatory.

The first contains input parameters with the bench configuration. The second indicates the file path to your workspace where all elements (file, procedure, sequence, etc…) will be stored.

The LOG path could be configured as a common folder (easy-to-access).

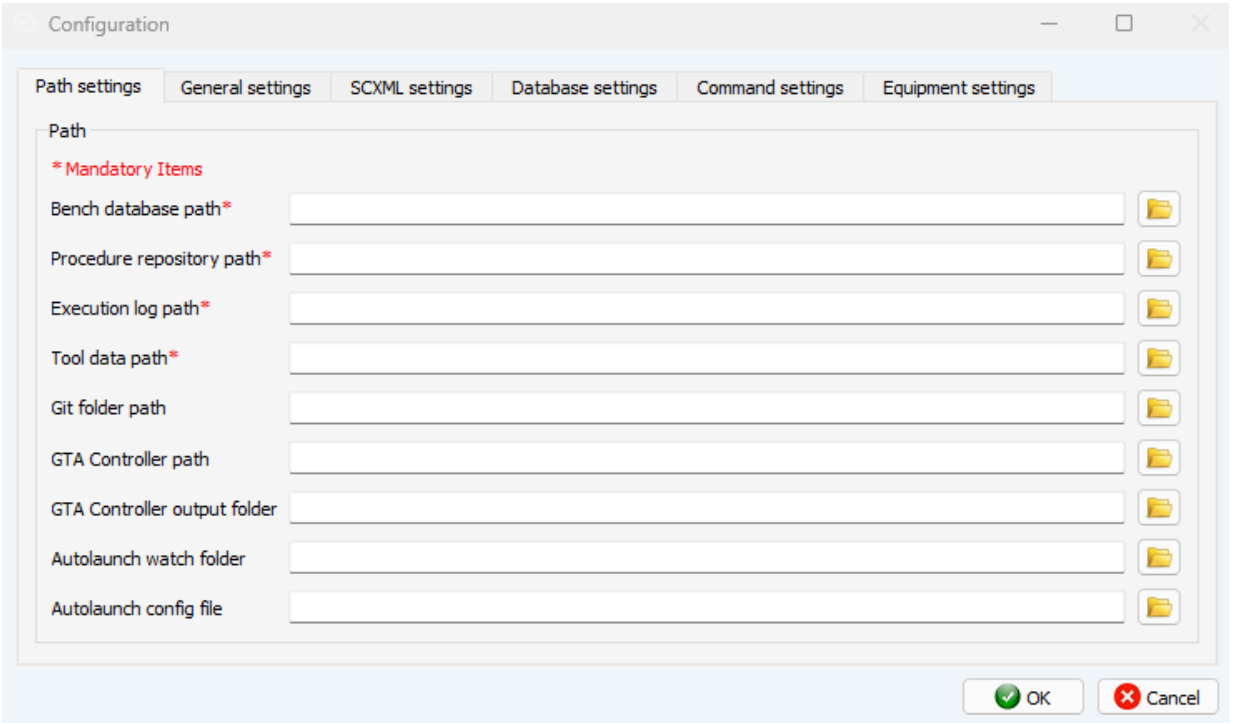
The Tool Data Path is evaluated when the GTA Editor installation.

Git Folder Path is used for integration with .git. This field is not mandatory and can be ignored if the user's Procedure repository path is not integrated with git. It should point to the procedure database folder which contains the .git folder.

GTA Controller Path is used for calling the Procedure dashboard from GTA Editor

AutoLauncher Watch Folder path is used for setting output folder for procedure dashboard

AutoLauncher Config File is used to add the execution time delays of AutoLauncher in procedure maximum execution time.



Paths settings

3.2.2. General Settings

This tab contains different options (checkboxes) that could be checked.

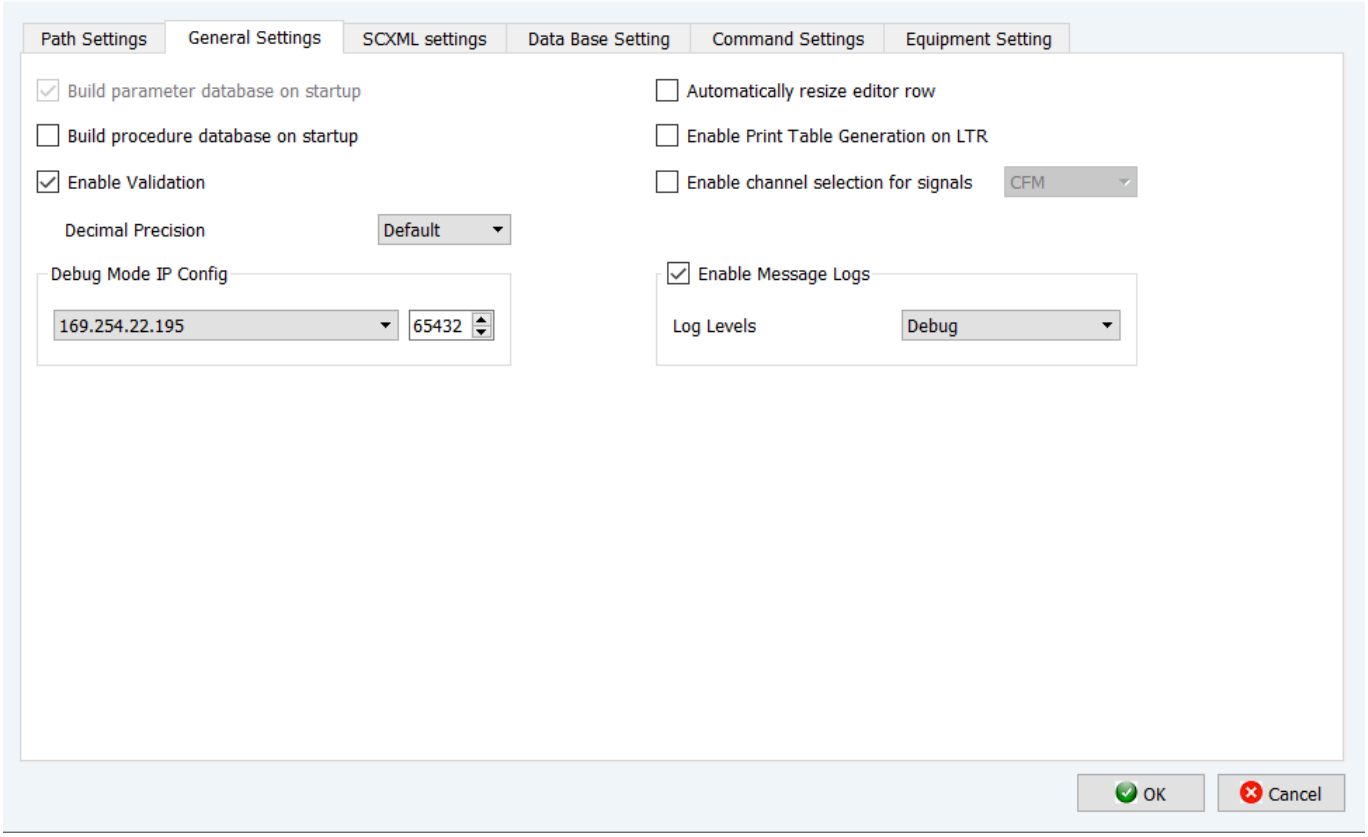


Figure: General settings

Following are the different settings that can be configured.

- **Database update on GTA start-up**
  This option allows users to update the bench database on GTA Editor start-up.

- **Enable Validation**
  This option allows the user to mark as validated each file that he would like to have as a reference. It is important when several files are called by another one.

- **Decimal Precision**
  This option can be changed; the user can choose another decimal precision (significant digits). Pay attention to using a 'dot' instead of a 'comma' if you want to set a Double/Float value.
  Check your decimal separator on your system parameters.

- **Automatically resize editor row**
  This option, once checked, allows automatically resizing a row depending on the size of the content for each row.

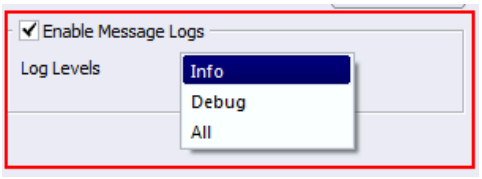
- **Enable Print Table Generation on LTR**

- **Enable channel selection for signals**

This option is applicable on search parameters (signal). Once checked, a drop down box with CFM, PW and RR engines are available. Selecting the expected engine and setting this choice gives the option of selecting the type of channel for a signal. Users will have to select the appropriate channel for the signal among all kinds of options.

- **Enable Message Logs**
  This option is now structured to be shown as warning, debug and all messages to the user. This can be used to identify errors such as Database connectivity loss and also information like time taken for an SCXML export.

  The current log levels defined are:
  - Info – Shows only Warnings generated from the tool
  - Debug – Shows only Debug messages generated from the tool
  - All – Shows warning, debug, critical and fatal messages generated from the tool.



- **Debug Mode IP Config**
  The Debug Mode can be called as an extended Interactive Controller (Manual Action) with enhanced functionalities. It also needs to start a server on an IP and Port to be able to control the execution flow.

3.2.3. SCXML Settings

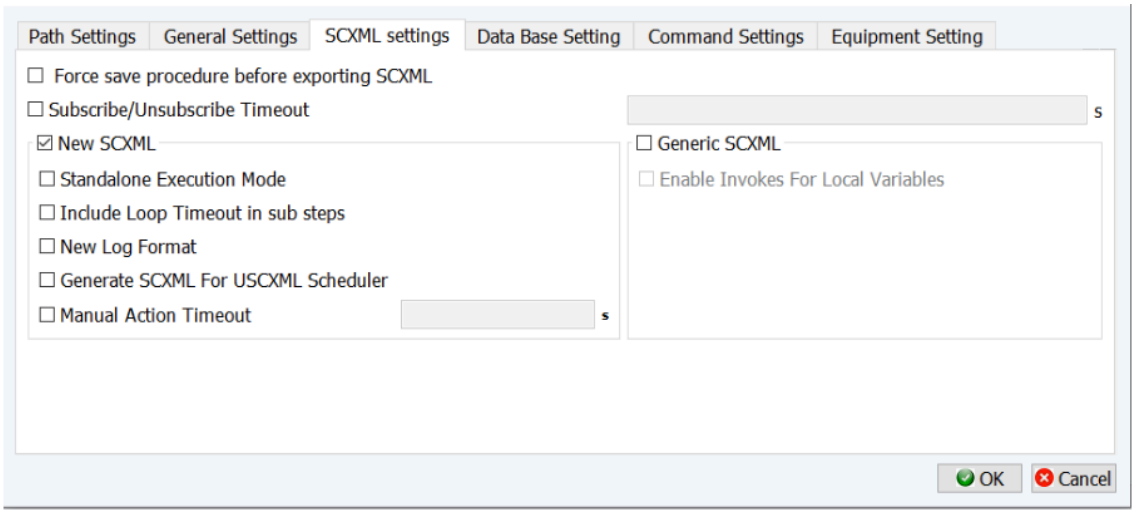


Figure: SCXML settings

Following are the different settings to be configured.

- **Force save procedure before exporting SCXML**
  This option is applicable for exporting a procedure to an SCXML file.

- **Subscribe/Unsubscribe Timeout**
  This option allows users to add a specific value in seconds for subscribe/unsubscribe timeout.

- **New SCXML**
  This option must be checked by default, it is recommended to not modify it.

3.2.4. Database Settings

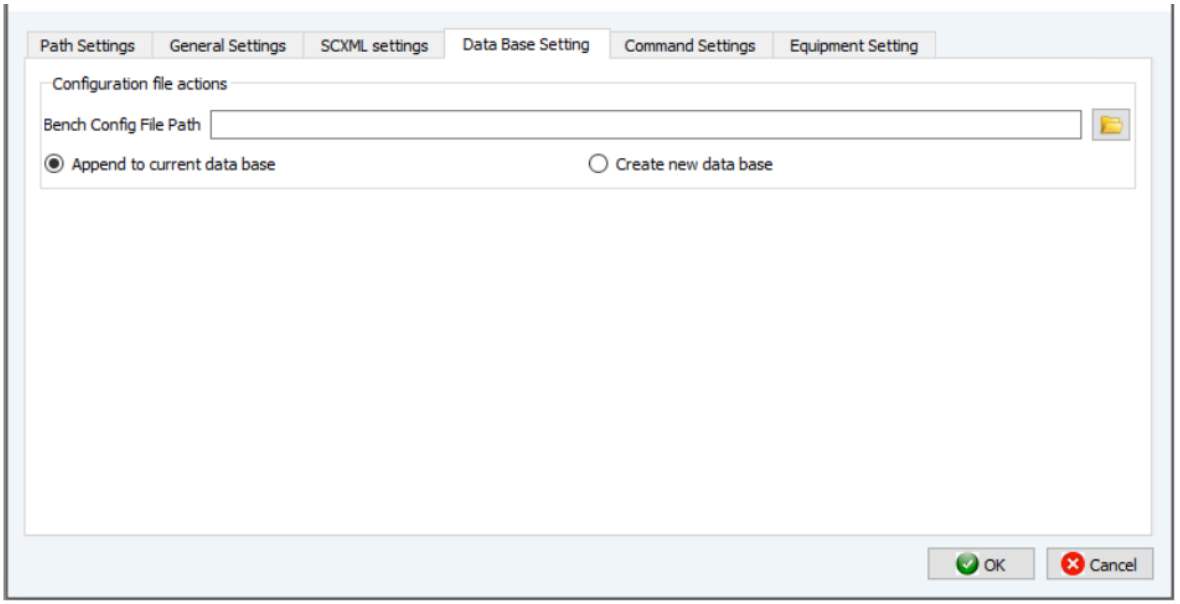


Figure: Data Base settings

3.2.5. Command Settings

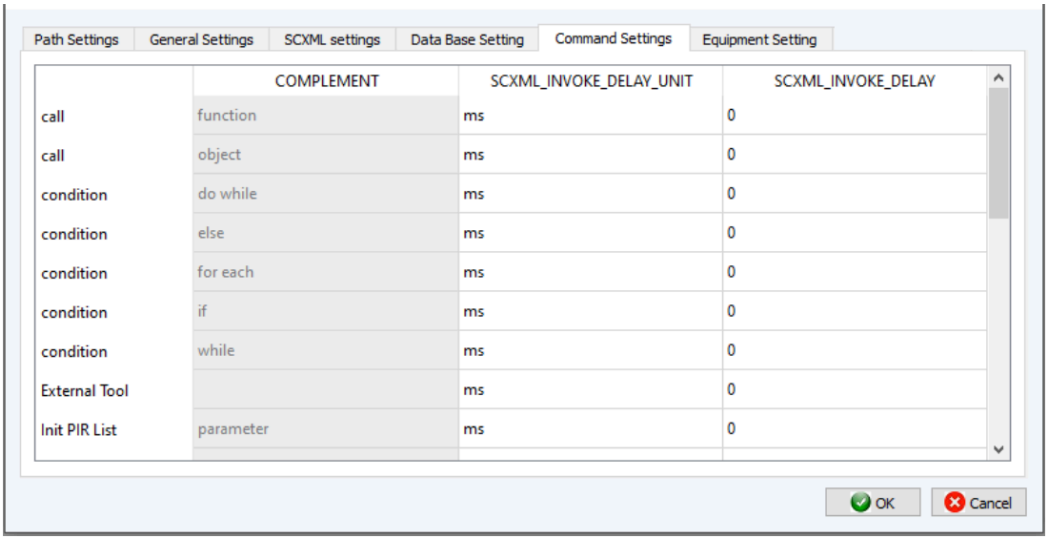


Figure: Command settings

In this tab, it is possible to configure a specific delay value for each command through GTA Editor.

3.2.6. Equipment Settings

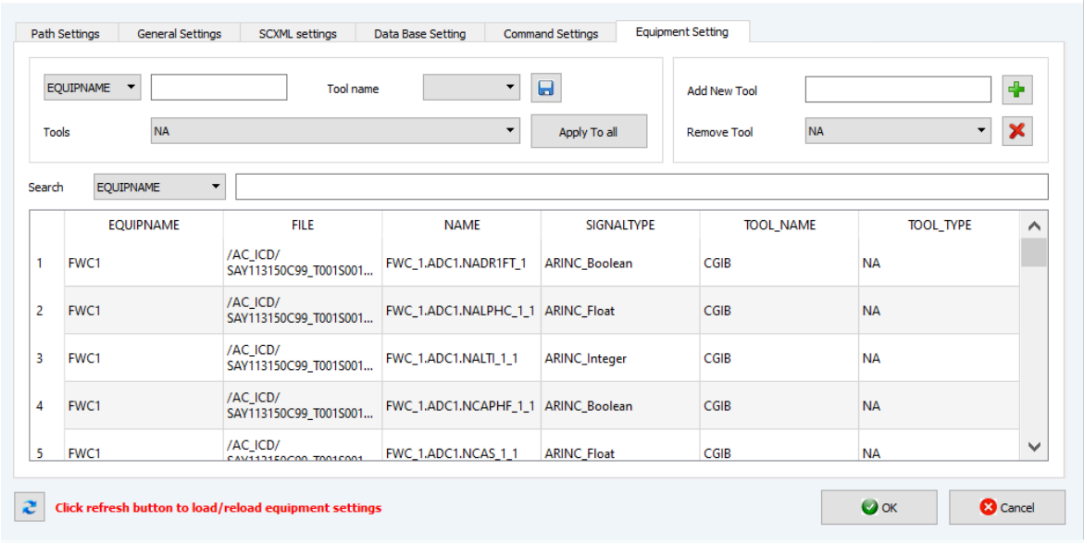


Figure: Equipment settings

The equipment settings have been added to allow the user to set tool names to equipment.
The figure above shows the equipment settings page.

- The table displays the equipment, file, file name, signal type and tool name from the database.
- The user can either set tool name to a single record by selecting from the table and clicking on  button or apply tool name to all the records using “apply to all” button

- A search option has also been provided to search the records. The search functionality can search records based on equipment name, file, file name, signal type and tool type. The user can select the search filter using the combo box just after the search label.

3.3. Section – “Editor View”

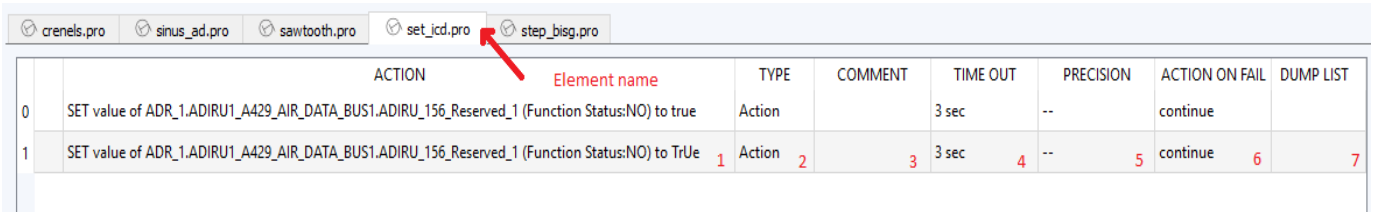


Figure: Editor View

Editor view presents the details of a test step under various fields as shown in above figure:

1. ACTION	Describes the command used in the test step.
2.TYPE	Either ACTION or CHECK.
3.COMMENT	User comment for the command used in the test step.
4.TIME OUT	User defined timeout for the command execution.
5.PRECISION	User defined or ICD value for range within which parameter value check must be performed.
6.ACTION ON FAIL	GTA allows users to define if the test execution must stop or continue based on the outcome of the current test step.
7.DUMP LIST	GTA allows users to add elements to the dump list during the command creation which will help the user to debug the command during failed execution, this information of the command can be seen under this column as well.

Note: User may keep the width of each column on the editor based on his convenience.

The editor view is a **multi-tabbed view**. The user can open multiple files in new tabs.
Additionally, the user can use a **split view** display of the documents via right clicking on a tab and selecting “Move to other view”:

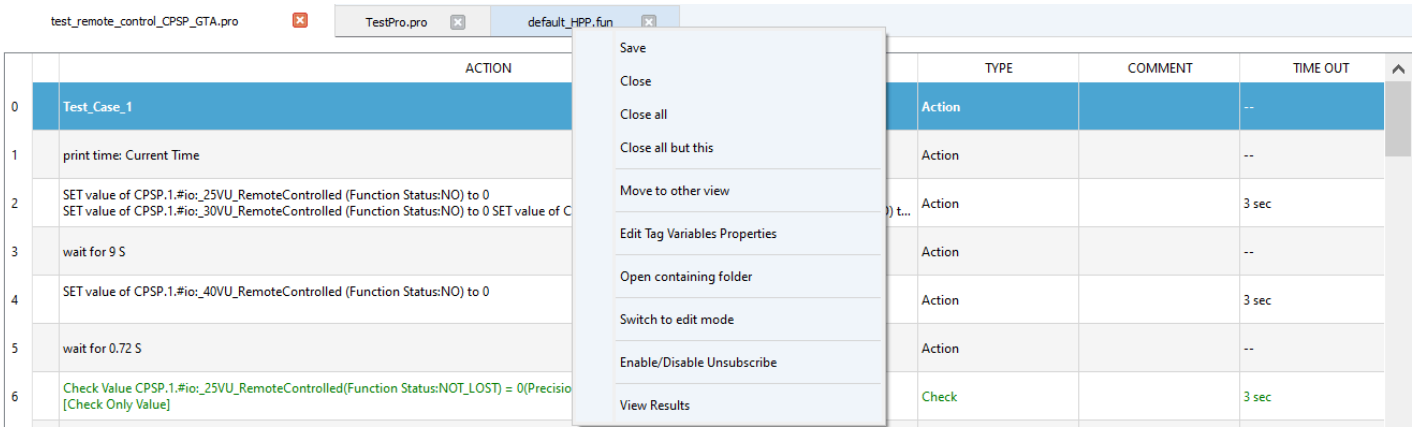


Figure: Editor multi-tab View 1

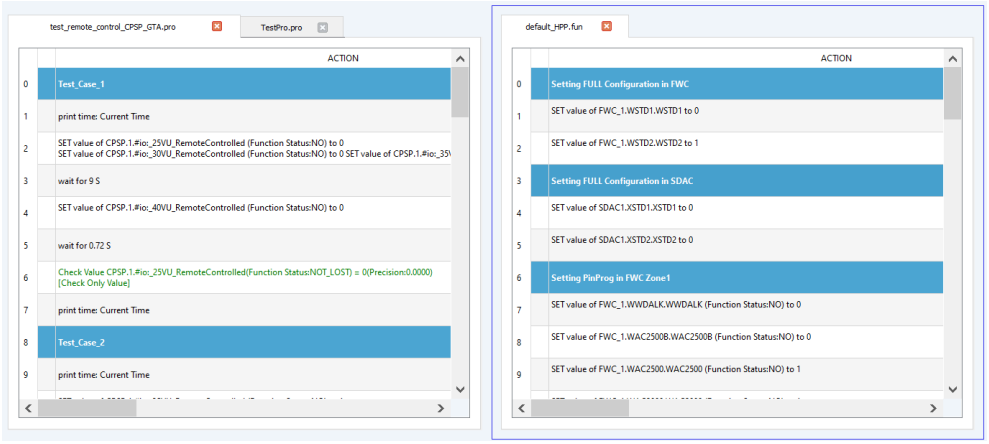


Figure: Editor multi-tab View 2

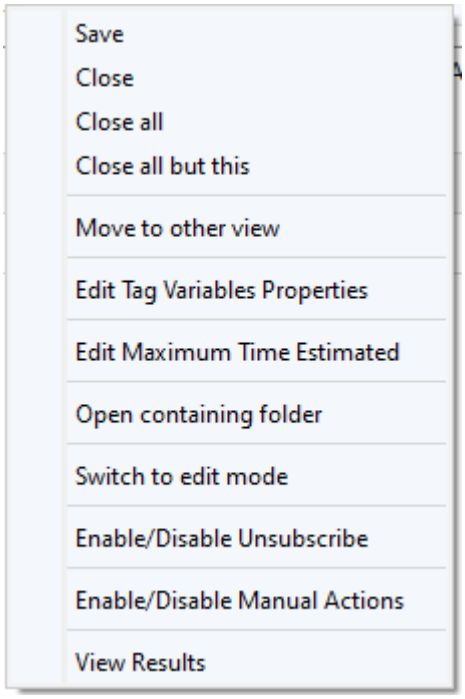


Figure: Editor multi-tab View option pop-up

Various actions are possible via the tabs context menu:

1. Save: Saves the current document.
2. Close: Closes the current editor window.
3. Close All: Closes all open editors.
4. Close all but this: Closes all open editors except for the current one
5. Open Containing folder: Opens the file in windows explorer.
6. Move to other view: Moves the tab to another view in a split view.
7. Edit tag variables properties: allows to modify variables properties of the document.

8. Edit Maximum Time Estimated: Changes that value by updating the related field in the <MODEL> node of the file if it exists, or by creating a new field if not.

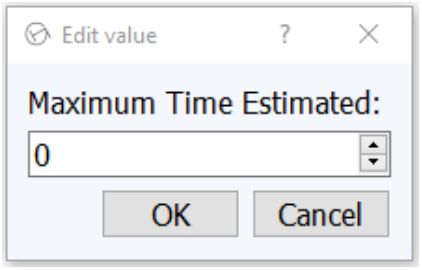


Figure: Maximum Time Estimated feature

9. Switch to edit mode
10. Enable/Disable Unsubscribe: it allows for each selected procedure to Enable or Disable the UnsubscribeAll at the start or the end of the .scxml file. Also there is option to unsubscribe file parameters only.

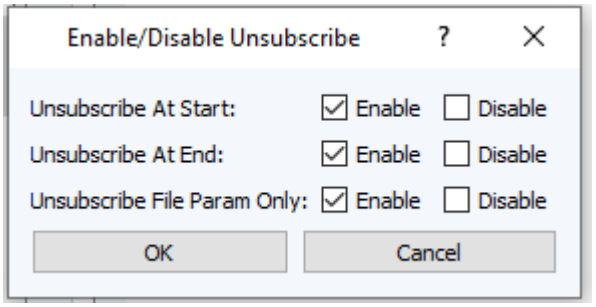


Figure: Unsubscribe feature

1. Enable/Disable Manual Actions: Select the manual actions of the current document to be ignored. If an ignored manual action has a checkValue just after it, that checkValue row is also ignored.

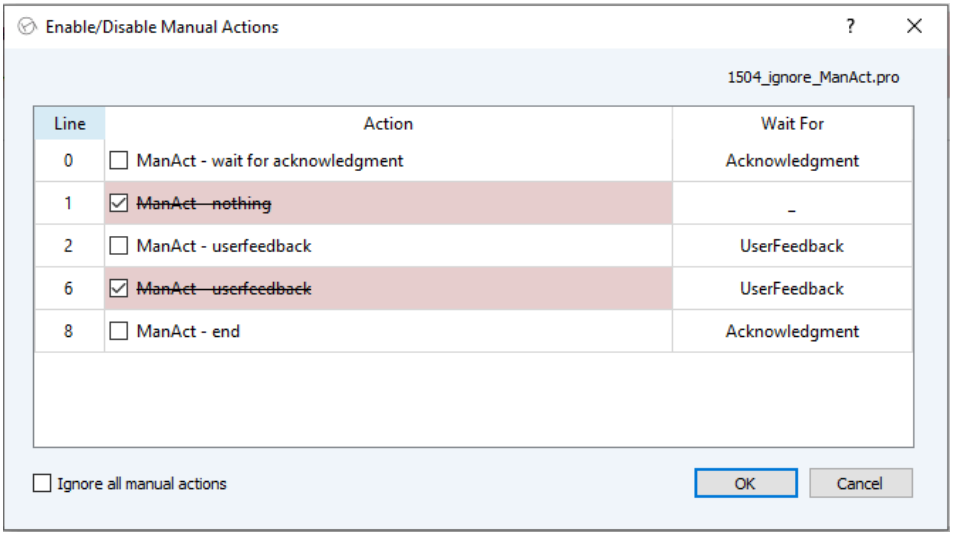


Figure: Manual action feature

2. View result: displays the results in result view (see below).

The user can create a new document by clicking on button in the toolbar. On clicking the new button a dialog as shown below will appear. The user can select the type of the element he/she wants to create, give a name and select a path in the DATA folder of GTA.

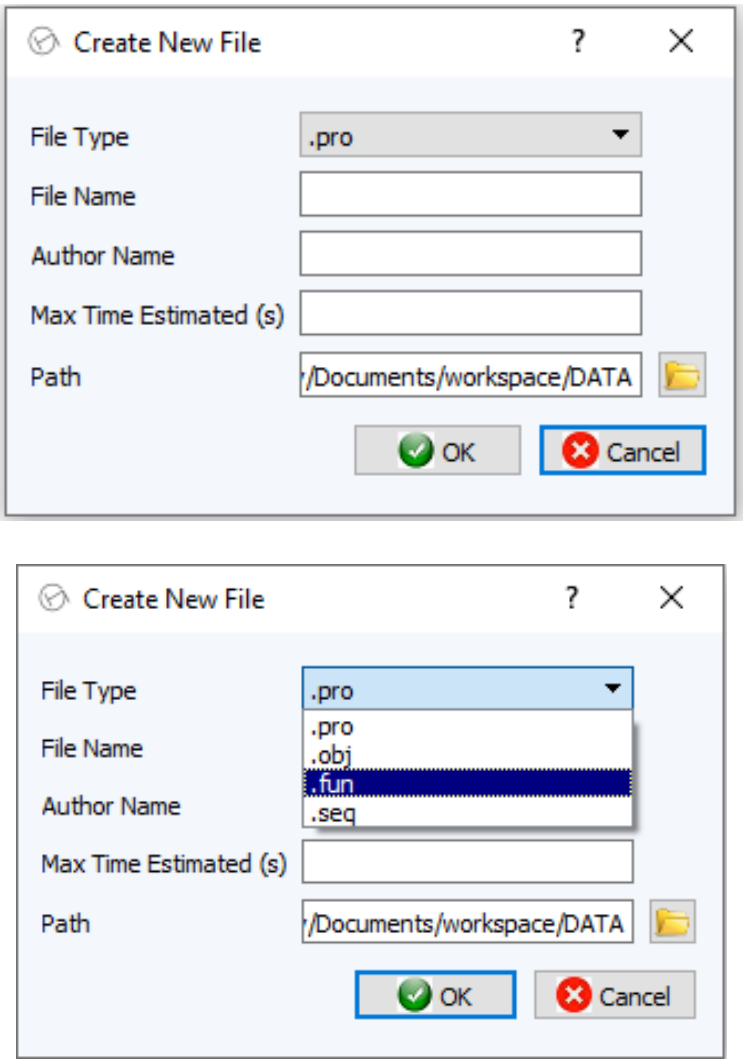


Figure: File creation pop-up

3.3.1. Editor Services

On selection of any row on the Editor View the tool provides user Right-Click context menu options as shown in figure below.

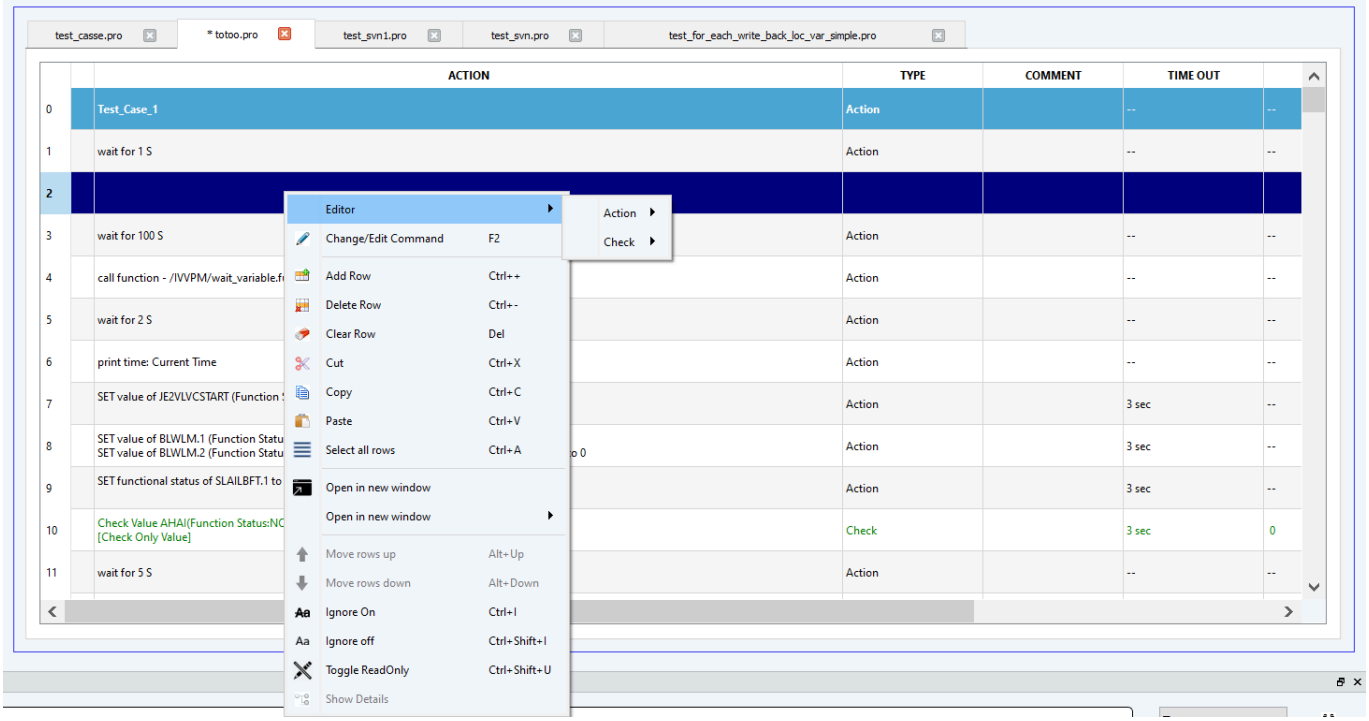


Figure: Editor context menu options

The Right-Click context menu features include the following services:

Action Selector: Detailed in next section.

Check Dialog: Detailed in next section.

Add Row – Allows users to add a new row.

Delete Row – Allows users to delete currently highlighted row(s).

Clear Row – Clears the content of the currently highlighted row.

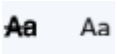
Copy – Copies the content of the currently highlighted row.
Note: In case of condition action, the content of next rows till END Condition is also copied.

Paste – Pastes the copied content to the currently highlighted empty row.
Note: Paste command is not applicable for overwriting the contents of a filled row, 'Only' applicable for an Empty row. Also, in case of condition action the content of next rows till END Condition is also pasted.

Cut Row – Cuts the row to paste it elsewhere.

Open in new window – Opens the selected call (*.pro/ *.fun/ *.obj) in a new window. This can be done by using a CTRL + Double-Click as well.

Move Rows – Allows users to reposition the currently highlighted row (Up or Down).
Note: Only ONE row can be repositioned at a time.



Toggle ignore on/ Toggle ignore off - This feature allows users to skip one or more commands. This is done by selecting commands that are to be ignored right click and click on toggle ignore on Commands will be struck out in the editor and these commands will be ignored while generating SCXML. Note that if a command has a parent (If, while, do while) that has been ignored then their children are automatically ignored.

Show Details – Allows users to view the details of a Call command. Details of the called Procedure/Object/Function are available in an additional window in tree view for a quick overview.

Note: Some of above-mentioned editor services are also available under the tool's 'Edit' option.

3.3.2. Tool Menu

GTA Editor provides a tool menu, available from Tool > Configuration.

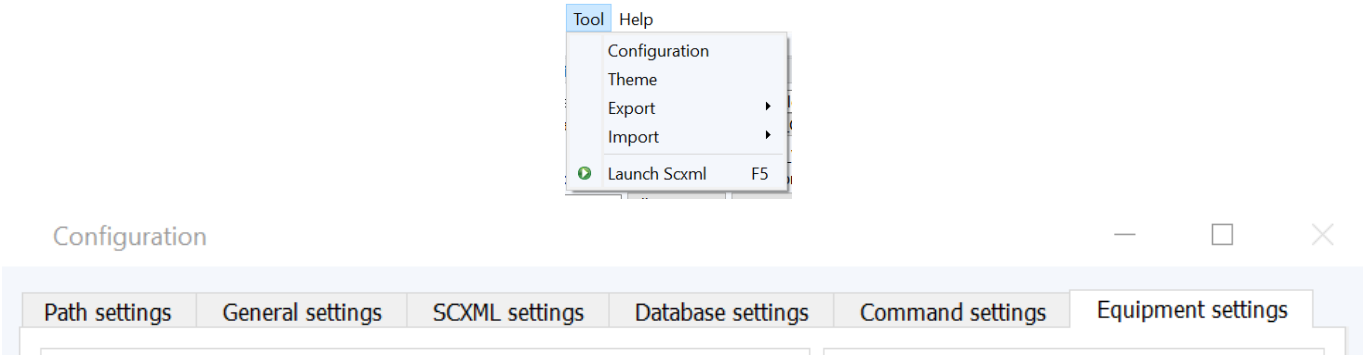


Figure: Tool menu

3.3.3. Tool Bar

GTA Editor provides a toolbar to enable users to quickly reach the GTA editor services.

Figure: Editor Tool Bar

3.3.3.1. Bench database building

Update DB: Performs a partial database update, based on any file modification or addition.

Rebuild entirely DB: Performs a full database rebuild.

3.3.3.2. Executing Tools

Run: This command enables the execution of the open Sequence or Procedure on the EMOTEST Scheduler module.

Check Compatibility: Allow to check that the content of the file is compliant before launching. This option validates the compatibility of the current document by analyzing to the following points:

- Ensures that Parameters used in elements are in sync with the Test Mean Configuration.
- Checks the presence of unauthorized characters in Variable/Parameter name.
- Ensures that the value assigned to the parameter is compliant to its parameter type.
- Ensures that a VMAC parameter is released after use.
- Ensures that a PIR parameter is released after use.
- Points to differences of parameter type with respective value (Based on ICD definitions)
- Ensures that unauthorized operations are not performed using local variables – Ex. CHECK FS, SET FS , SET Equations etc.
- Ensures that no ECMAScript reserved word (see below) is used as a parameter name:

"abstract", "arguments", "await", "boolean", "break", "byte", "case", "catch", "char", "class", "const", "continue", "debugger", "default", "delete", "do", "double", "else", "enum","eval", "export", "extends", "false", "final", "finally", "float", "for", "function", "goto", "if", "implements","import", "in", "instanceof", "int", "interface", "let", "long", "native", "new", "null", "package", "private", "protected", "public", "return", "short", "static", "super", "switch", "synchronized", "this", "throw", "throws", "transient", "true", "try", "typeof", "var", "void", "volatile", "while", "with", "yield"

Procedure Dashboard Launcher: Opens the Procedure dashboard Dialog. Procedure execution list can be maintained on it and then clicking on Launch button will launch the GTAController procedure dashboard.

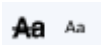
3.3.3.3. Command Editor

Find/Replace: Allows to find a text in the open Window or replace a character by another one.

Undo: Allows to go back in actions

Redo: allows to go forward in actions

Move Rows up/down: Allows users to reposition the currently highlighted row (UP or DOWN). Note: Only ONE row can be repositioned at a time.



Ignore On / Ignore Off: This feature allows users to skip one or more commands. This is done by selecting commands that are to be ignored right click and click on toggle Ignore ON Commands will be struck out in the editor and these commands will be ignored while generating SCXML. Note that if a command has a parent (If, while, do while) that has been ignored then their children are automatically ignored.

3.3.3.4. File Editor

New: Allows users to create a new element based on selected mode. (Sequence, Procedure, Function or Object).

Save: Allows users to save as a new element or by overwriting of contents to an existing element.

Save As: If a user wants to create a new element based on an existing element, he can use the 'Save As' feature. This allows the user to save the modifications to the element by providing a new element name, so the "Name" field on editor view shall be modifiable in this case.

Save all open files: Allows the user to save all currently open documents at once.

3.3.3.5. Result Toolbar (only for Result View)

Figure: Result Tool Bar

Expand all / Collapse all: Allows users to expand and collapse the contents displayed in the GTA Result View.

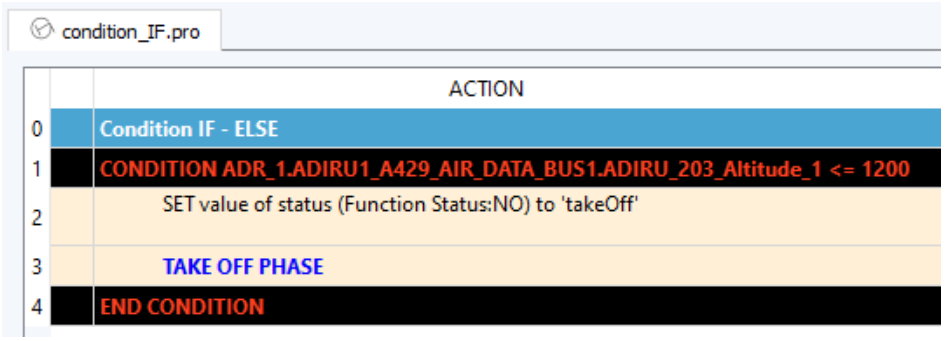
Select all / De-select all: Allows users to select/deselect all elements detected in the GTA Result View.

New Result page: Allows users to open a new tab in the GTA Result View.

3.3.3.6. View

Collapse to title: This feature allows users to view a summary of the complete procedure by minimizing contents based on title rows.

Indent/UnIndent view: This feature makes the content of the document more readable, elements in conditions action are indented and colored.



3.3.3.7. **Export Toolbar**

Export SCXML: Converts the selected file into an SCXML file

Generate Test Report: makes a test report in the form of a csv file in the form of a table of checks for each variable in each test case. The prerequisite for this function is a properly generated log file from valid procedure run.

Export LTR/LTRA: It's only active in Result View, it converts the log file loaded with the given pro file or sequence into an LTRA output file

Switch between Editor and Result view: The Switch to Edit Mode option is provided to move to the Editor mode after Search/Replace or Check Compatibility.

After performing either Search/Replace or Check Compatibility, the document(s) are highlighted and are in Search mode or Check compatibility mode, in order to remove the highlights and get back to the Editing mode, the user can Right-Click on the tab of the document and select the "Switch to edit mode" or simply click the ESC key.

Export csv to pro: Converts an input CSV file into a procedure

3.3.4. Shortcut Keys

User may use these shortcut keys to access the GTA editor services:

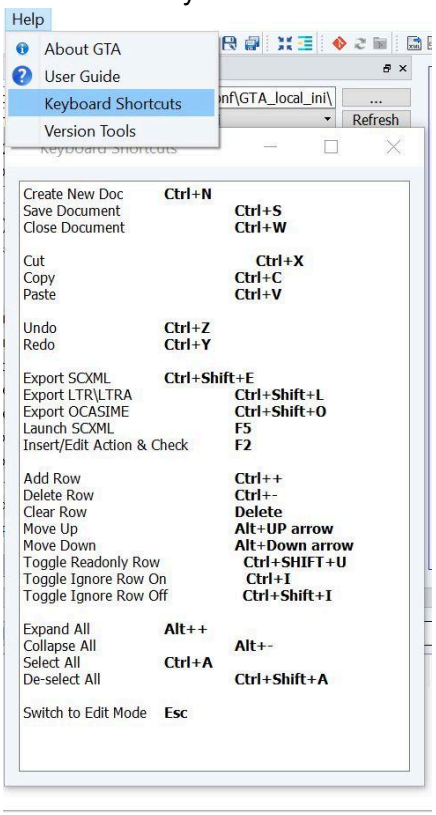


Figure: Shortcut Keys

	Action	F2
	Check	F2
	Add Row	Ctrl + +
	Delete Row	Ctrl + -
	Copy	Ctrl + C
	Paste	Ctrl + V
	Clear Row	Del
	New Element	Ctrl + N
	Find and Replace	Ctrl + F
	Save Element	Ctrl + S
	SCXML	Ctrl + Shift + E
	CSV	Alt + C
	LTRA	Ctrl + Shift + L
	Undo	Ctrl + Z
	Redo	Ctrl + Y

Note: These shortcut keys are also available under the tool's 'Help' menu.

3.3.5. Action Selector

The action Selector provides users the option to choose the command that he wishes to use for a test step. The action selector also allows users to enter comments and DOORs requirements.

The Action Selector dialog can be opened in any of the following ways:

- ✔ Double Click on the Action Cell on a row
- ✔ Right Click on the Row > Editor > Action
- ✔ 'Edit' menu > Action
- ✔ Shortcut is 'F2'

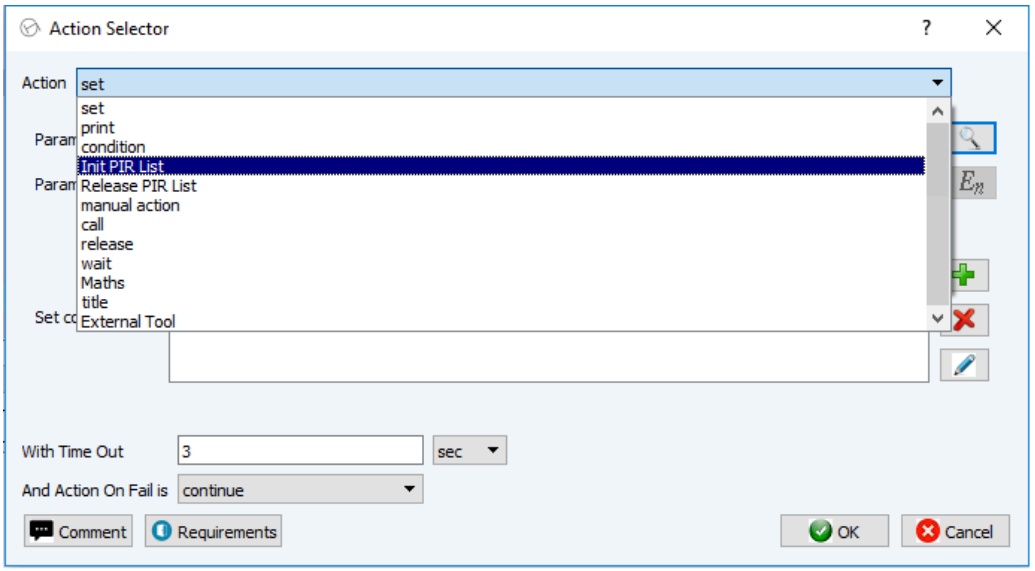


Figure: Action Selector - Procedure

The number of actions in the selector dropdown is dependent on the element mode currently selected:

- ✔ **Procedure** has the modes items as shown in figure above
- ✔ **Function** modes action selector items as shown in figure below.

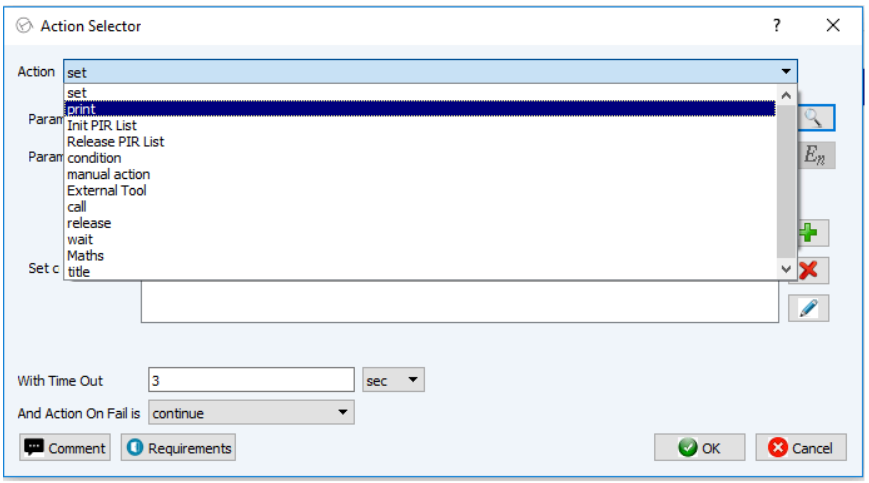


Figure: Action Selector - Function

✔ **Sequence** mode action selector content is as shown below

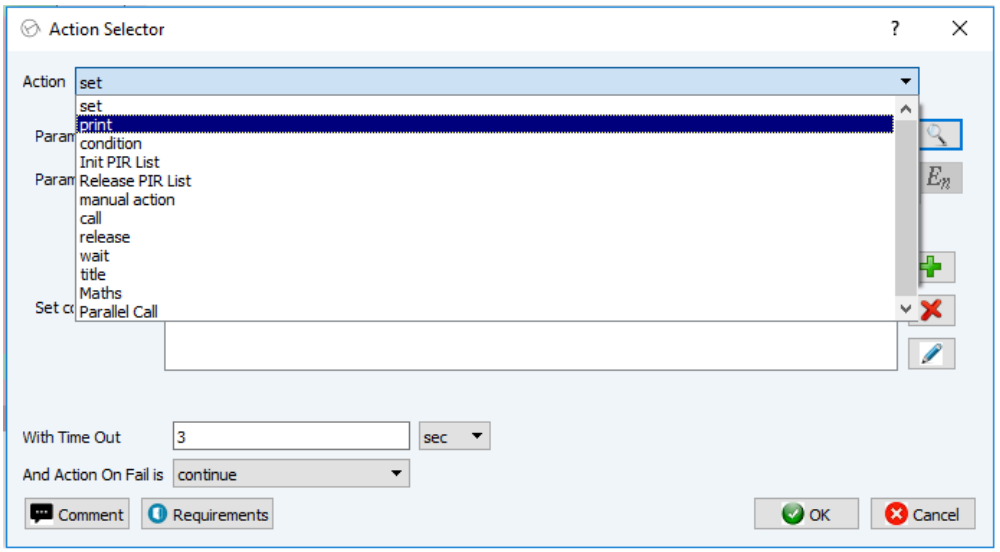


Figure: Action Selector items - Sequence

✔ **Objects** mode action selector content is as shown below

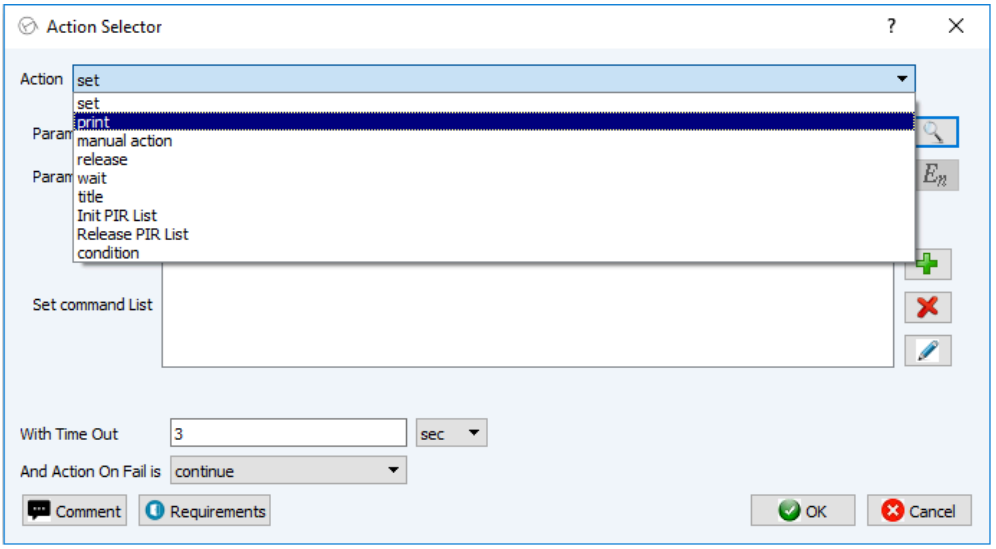
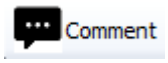
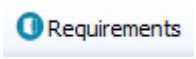


Figure: Action Selector items – Object

The comment button allows the user to enter the comments for the current command.



Below is the figure that shows the HMI for entering comments.



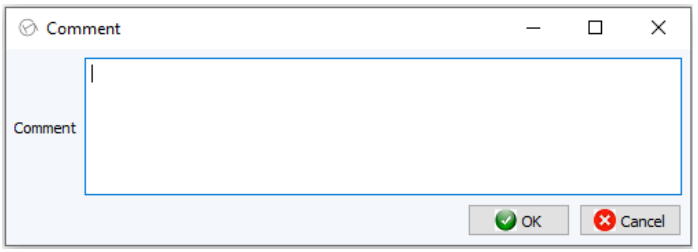


Figure: Action Selector - Comments

Similarly, the requirement button allows users to add requirements for a command. Following is the HMI for adding requirements.

Figure: Action Selector - Requirements

3.3.6. Parameter Selector

The parameter selection window helps users to search for a specific bench parameter or local variable.

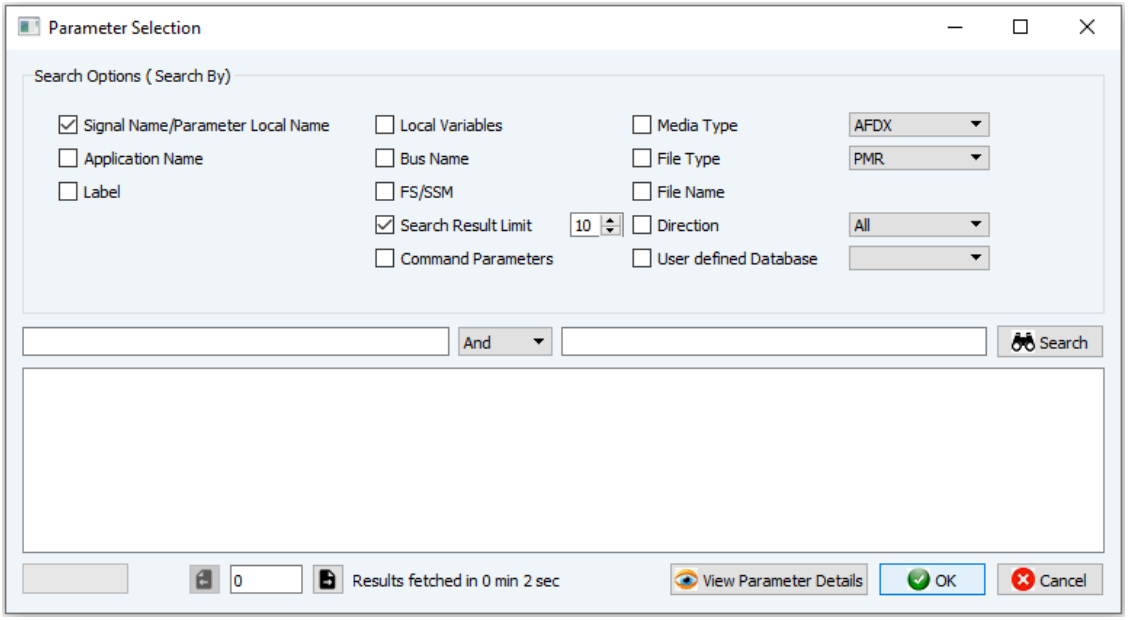
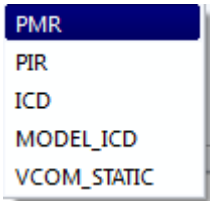


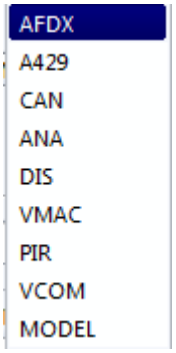
Figure: Parameter Selection widget

The widget permits user to perform filtering based on the following:

- ✔ File Type



- ✔ Media Type



- ✔ File Name
- ✔ Local Variables (From history of all previously used local variables)
- ✔ Bus Name (ICD Column)
- ✔ Functional Status (FS/SSM) (ICD Column)
- ✔ Signal Name/Parameter Local Name (ICD Column)
- ✔ Application Name (ICD Column) &
- ✔ Label (ICD Column)

Command parameters: By picking this option user can search for various internal parameters of commands, such as their Return status and so on.

Return status of set, check and condition_if actions can be used

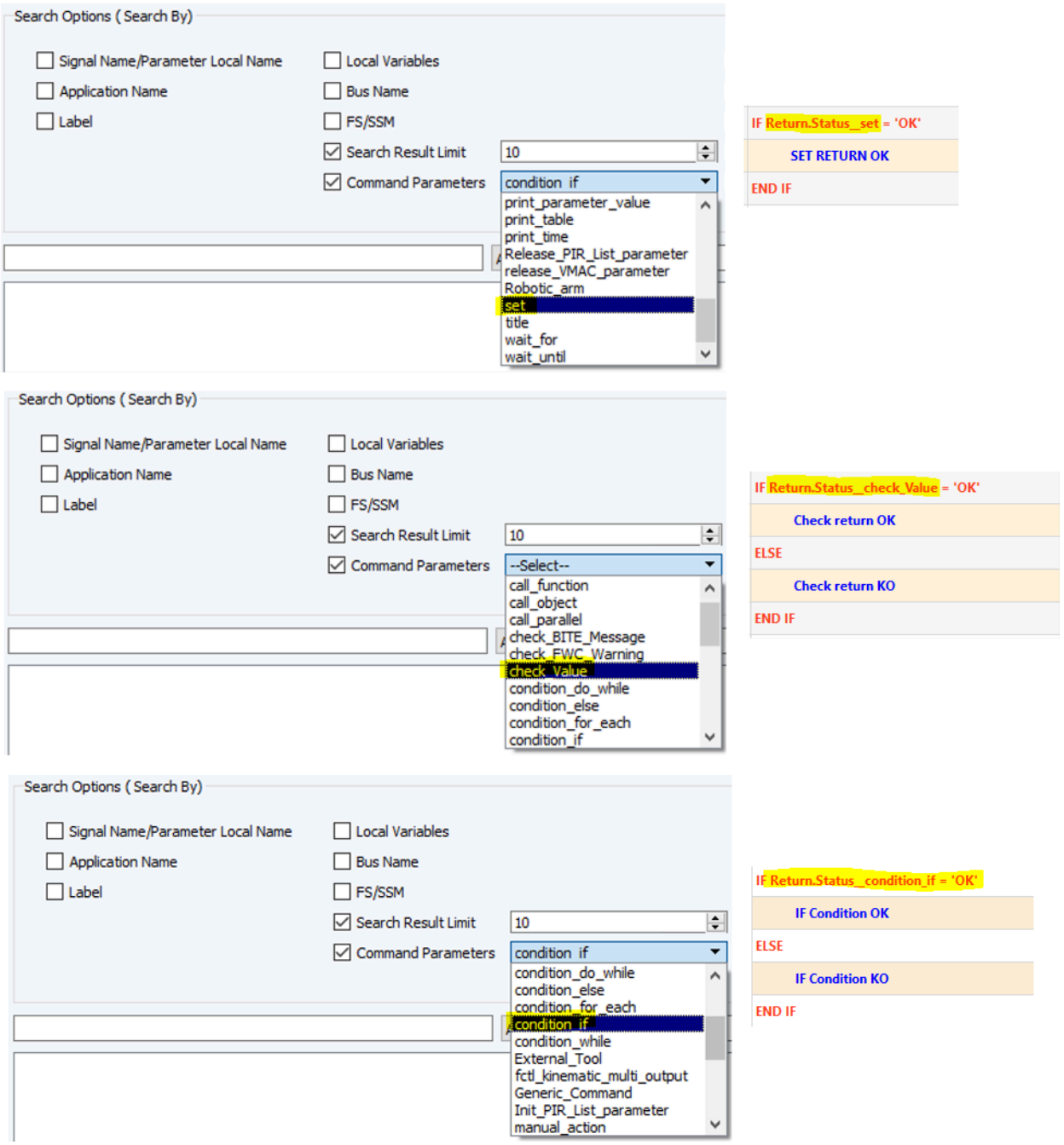


Figure: Parameter selection - Command Parameters - Return status function

Using the Result status for a Check OK or KO in GTA editor, it is possible to:

- use a PRINT message
- use a MAN ACT
- call an external tool like "Snagit", “Sandra”,..

Please, see a ‘Result view’ example in the figure below using these commands linked to a Result status.

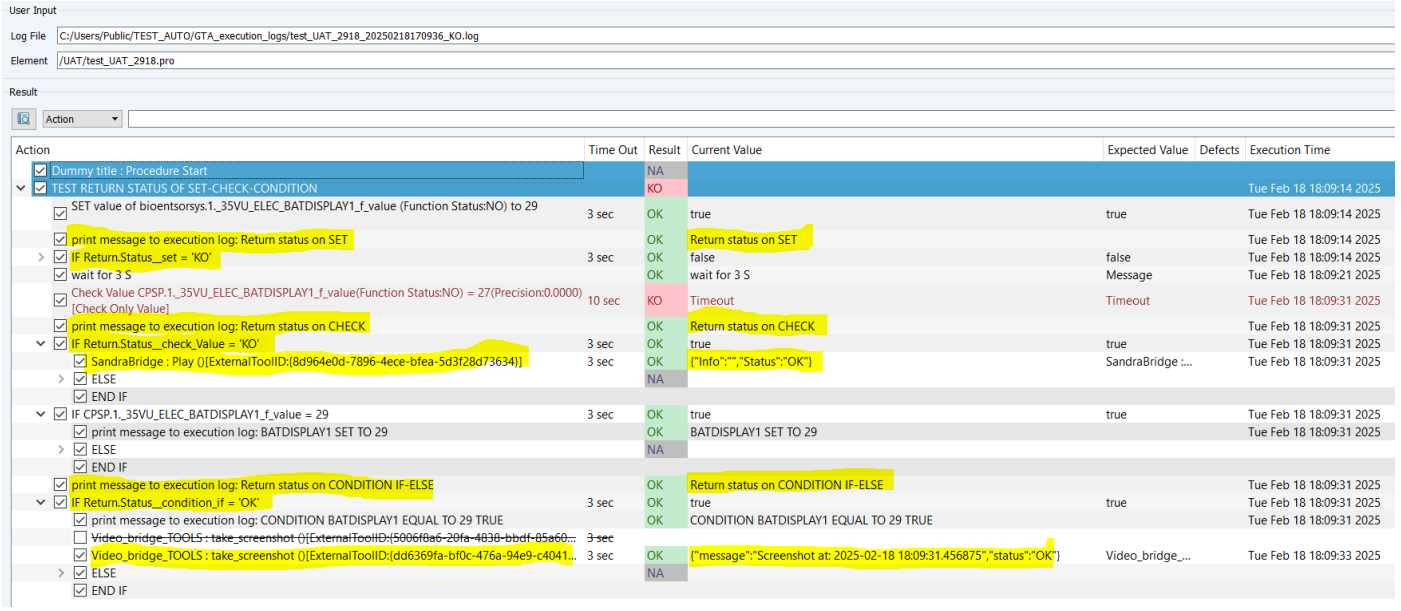


Figure: Parameter selection - Command Parameters - Associated functions/commands with Return status function

On searching a specific text, the search result widget shall be filled in with parameters information based on GTA library or local variable list parsing. (See figure below)

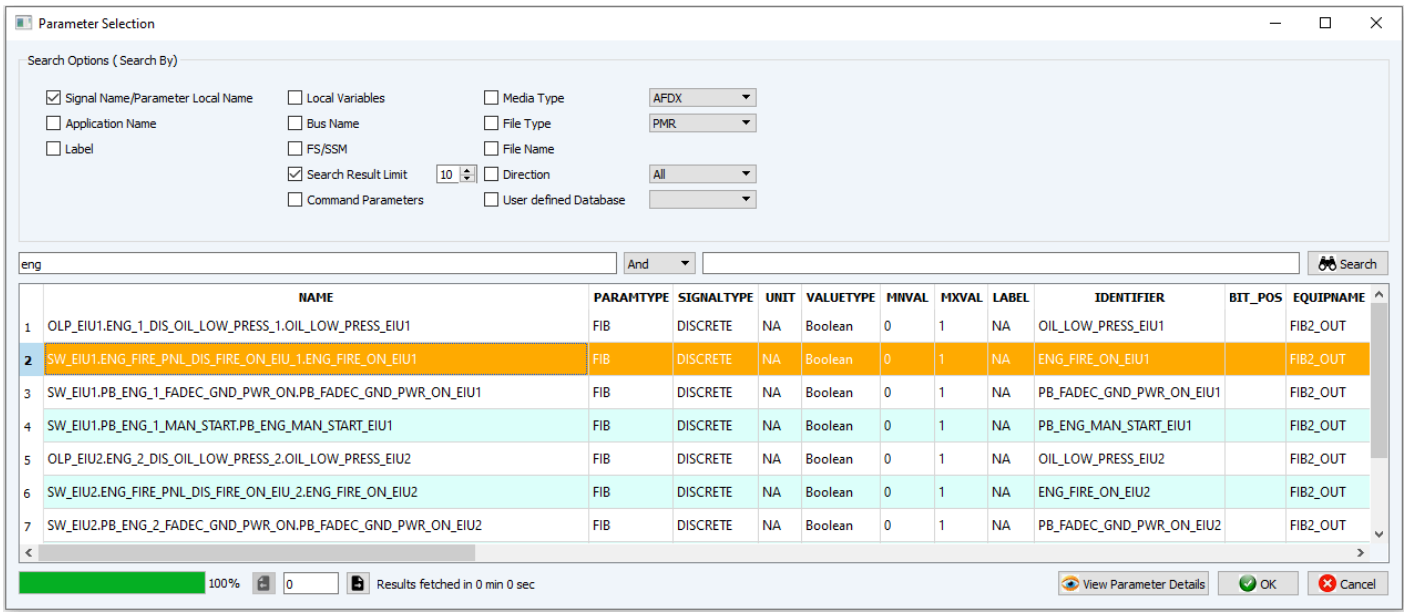


Figure: Parameter Selection - Search results

To view the details about a bench parameter, user can click on the View Parameter Details button. The details are shown as in figure below:

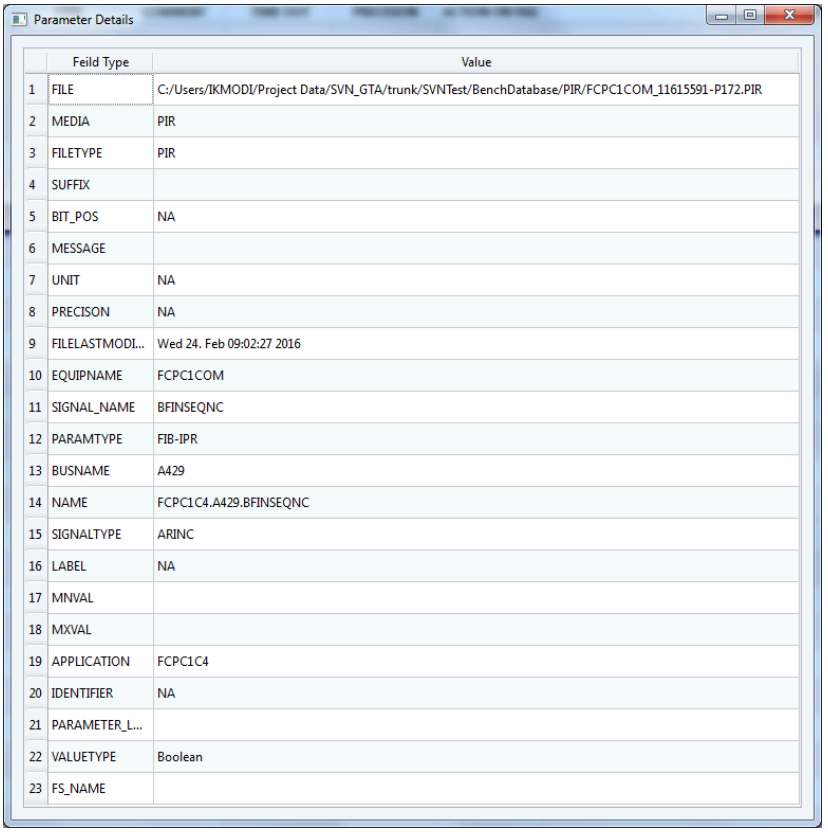


Figure: Parameter Selection - View Parameter Details

3.4. Section - ‘Header’

The Header section allows the user to provide a description for the created Sequence/Procedure/Function and Objects.

3.4.1. Create a New Header

To create a new header the user must click on  in the Header Window. On tool execution start, the user will see a new header by default.
New header by default has the “Header Name” as “Template”.

On saving action, a new header will be created then deleted, just the time to copy all information in the current file. To create a new header from an existing header user must update the “Header Name” and click on “FILE Save Header”.

3.4.2. Customize Header

Users can customize the header as per his requirements by adding, modifying, removing and repositioning Fields as per his needs.

Export to LTRA : User can control if the column needs to be exported as part of LTRA report.

 **Add Field:** Allows users to add a new field to the header

 **Edit Field:** Allows users to update the field properties. (*)

 **Remove Field:** Note that a Mandatory field cannot be removed by the user unless he explicitly changes the mandatory property by editing the field using the “Edit Field” button. (*)

 **Position Field:** Up and Down arrow can be used to position the field in user defined order. (*)

 **Save Updates:** This button only overwrites the changes performed by the user to an existing Header. *(Note: A header must already exist to perform the save updates Else this button will be disabled)*

 **Apply and Close:** This button will apply changes in the current element and close the current Header widget window. **Warning ! To see the update in treeview, click on Re/Create DATA button. Update to be done once all headers are updated, as this may take a while**

***Note:** “Header Name” field and Files information fields cannot be positioned, removed or edited.*

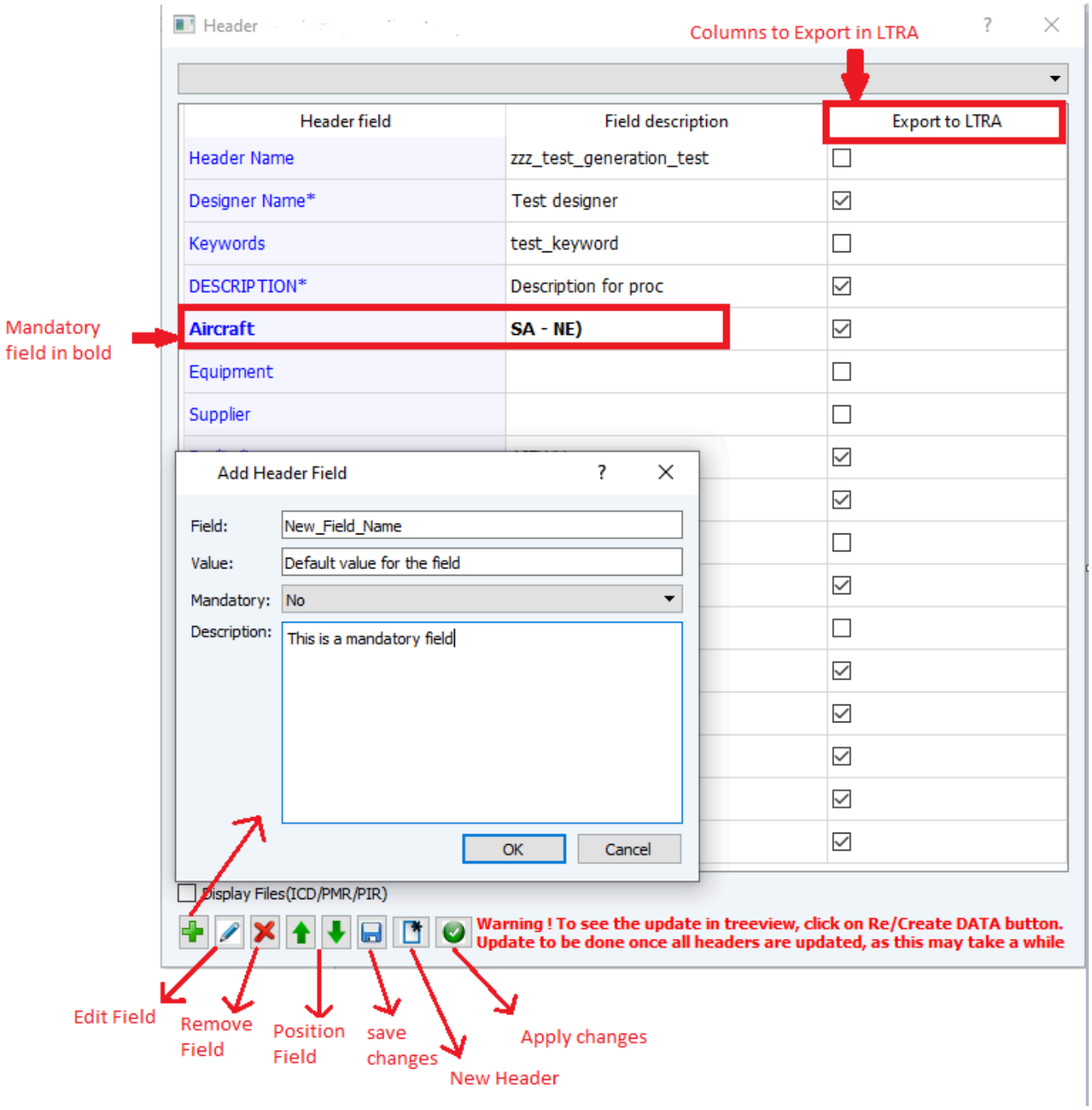
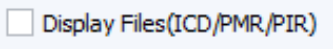


Figure: Manage Header Fields

Above figure displays the buttons that are used to customize and manage a header.

3.4.3. Display File Information in Header

Sometimes user may need an overview of the bench database file names that are used in a specific procedure, to fulfill this feature GTA's header section allows the user to see the list of all the ICD, IPR (Internal Parameters) and PMR files used in a Sequence, Procedure, Function or Object.

The DisplayFiles check box 

allows users to see this list. (See image below)*

**Note: Users must save the updated procedure before clicking on the DisplayFiles option.*

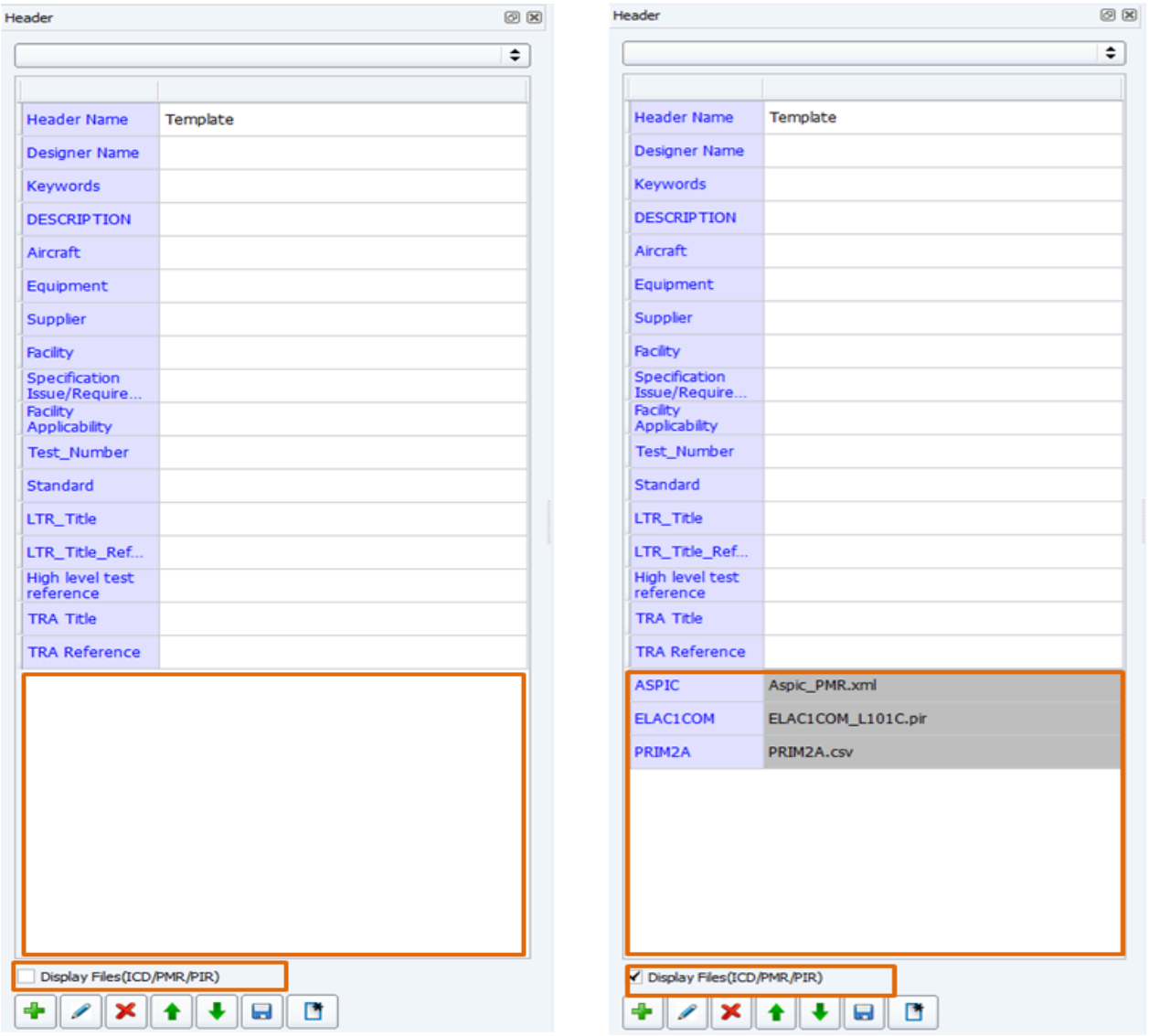


Figure: Display Files in Header

3.4.4. Save Header

3.4.4.1. Save a New Header

To save a header (as .hdr) so as to make it available in the header dropdown (To keep a filled template available for future use), user must update the “Header Name” and click on “FILE Save Header”.
Saved headers are reflected in a drop down box inside the header section.

3.4.4.2. Save Header with Procedure

To save a header along with the procedure, User must open the procedure in the editor and update the header as per requirement and click on save procedure button  on the editor.
This saves the header along with the procedure (i.e. without creating any .hdr or reflecting in the header drop down menu) and this header will be available whenever the user opens the procedure.

Note: This is also applicable for Sequence/Function and Objects. Also all mandatory parameters must be filled for a successful save.

3.4.5. New Header

The user can now create a new header by clicking on the “New Header” button which is placed near the “Save Header” button. The current header is cleared and the user can now write a fresh header.

Note: By default, when the user saves their procedure, a new header file with the same name as the procedure will be created and then deleted just after copying all the information it contains.

3.5. Section - ‘Document Viewer’

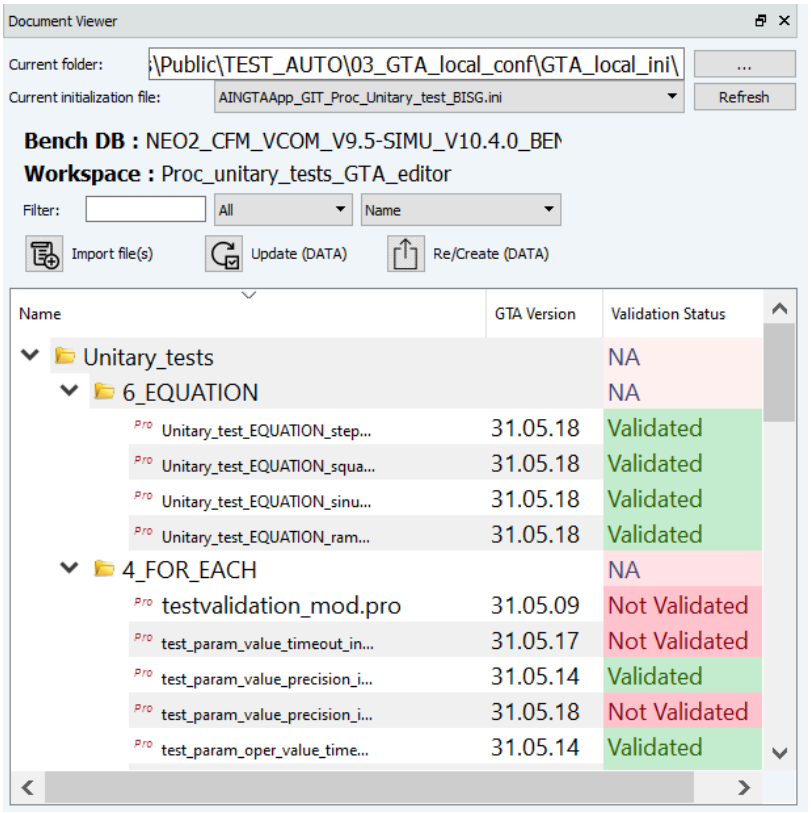


Figure: Document Viewer

The document viewer provides the user with an element browser and a filter to perform a quick search of already created editor items that are stored in the GTA file repository.

Users have the possibility to customize columns with information about their files, and recently added the ability to show the GTA Version with which the file was last modified. The version is updated automatically on file saving. To do that, Right Click on the name of columns is required.

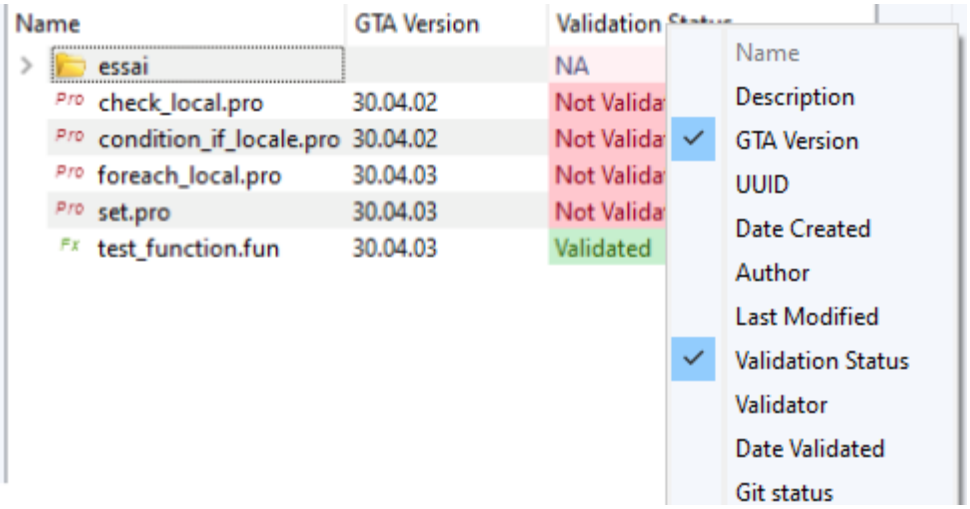


Figure: Document Viewer - Column customization

A right click menu is provided to the user to access services provided in the element browser.

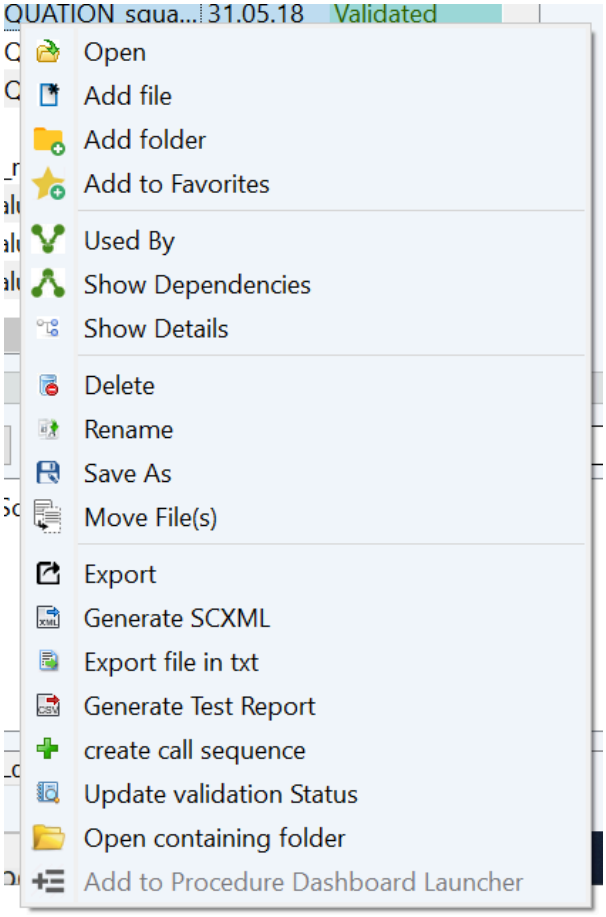


Figure: Element browser Context Menu

Open

It allows the user to open the element (Sequence/Procedure/Function/Object) in the Editor View.
Note: Any editor modification shall overwrite the contents of the element, so the “Name” field on editor view shall be non-modifiable in this case.

Add to Favorites

Users can add any element to the favorite list, this list will be visible in “Fav” element filter dropdown.

Used By

This feature allows the user to check the other existing elements in the repository which are dependent on the current element. This functionality helps users to take appropriate actions on the element (like modification or deletion) based on prior knowledge of usage of the element.

Show Dependencies

This feature allows user to check the other existing elements in the repository including *csv files and external tools which are used by the current element.
Afterwards the user can export the list of dependencies by clicking on the **Export** button of the dedicated widget. A csv file will be generated and it shall have the same name as the procedure followed by “_dependencies”.

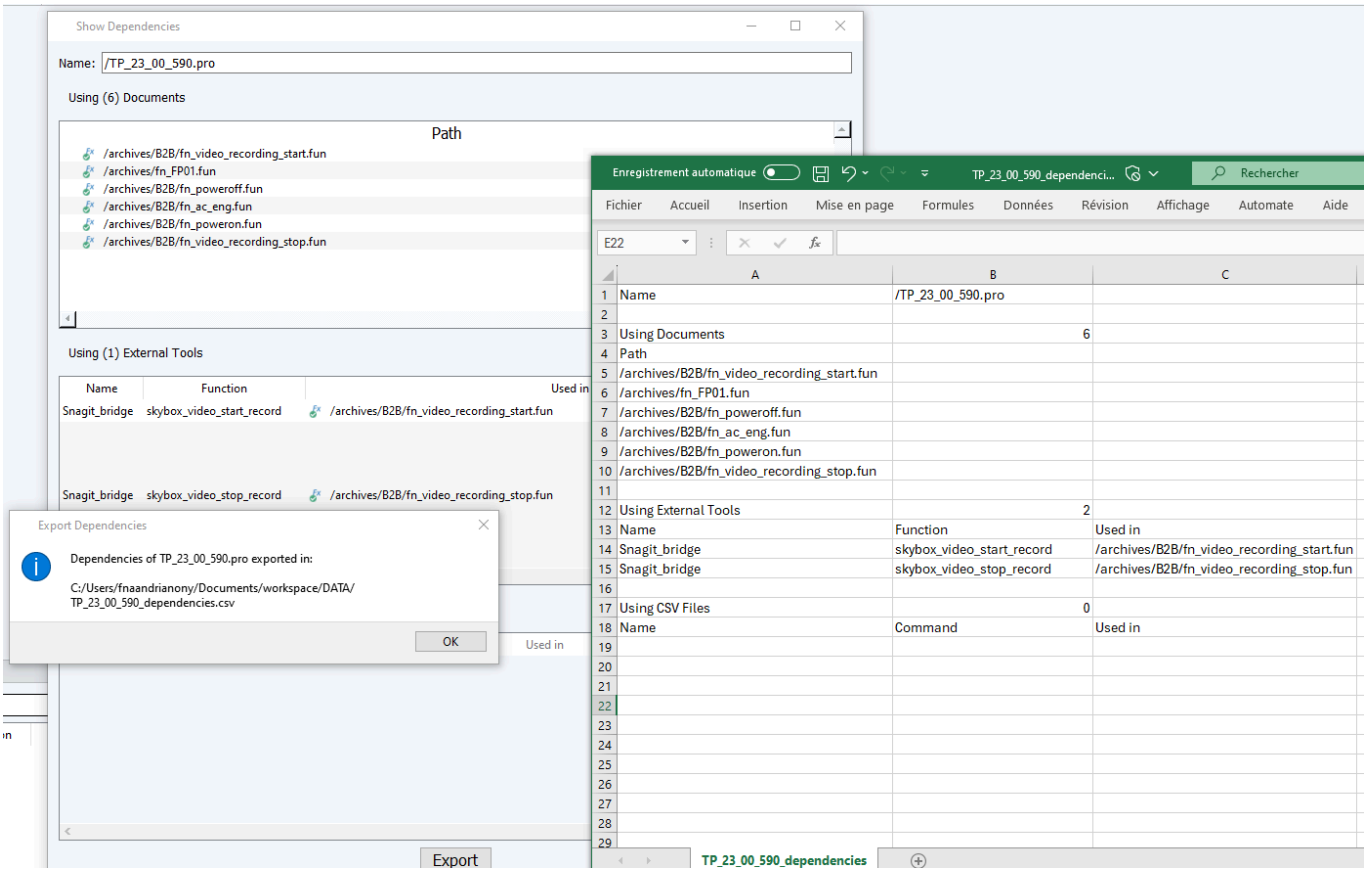


Figure: Show dependencies

Show Details

Show Details will generate a tree view structure where the user can view the element and all called elements in a tree format.

Delete

Deletion of an element from the SFS repository is only possible if no dependency exists for the element. To ensure that an element is deleted, all dependent elements must be deleted first.

Note: User must exercise his sole discretion to modify or deletion an existing element.
The tool does not stop users from deleting a file from the GTA repository. (Since storage medium is SFS repository and repository access is not in tool scope.)

Rename

Users can rename any element from the tool itself.

Save As

If a user wants to create a new element based on an existing element, he can click on the element and use the ‘Save As’ feature. This allows the user to save the modifications to the element by providing a new element name, so the “Name” field on editor view shall be modifiable in this case.

Export

Export of an element from the SFS repository will export all elements and all related elements used in this element like functions, procedures etc. to the user provided directory.

Generate SCXML
SCXML will be generated for all the files selected in the document browser.

Export displayed procedure in '.txt'
Export the content of a procedure as displayed in GTA Editor view into a plain text file.

Generate Test Report
Make a test report in the form of a csv file in the form of a table of checks for each variable in each test case. The prerequisite for this function is a properly generated log file from valid procedure run.

Create call sequence
Users can select multiple elements from the element browser, and create a sequence with call commands.

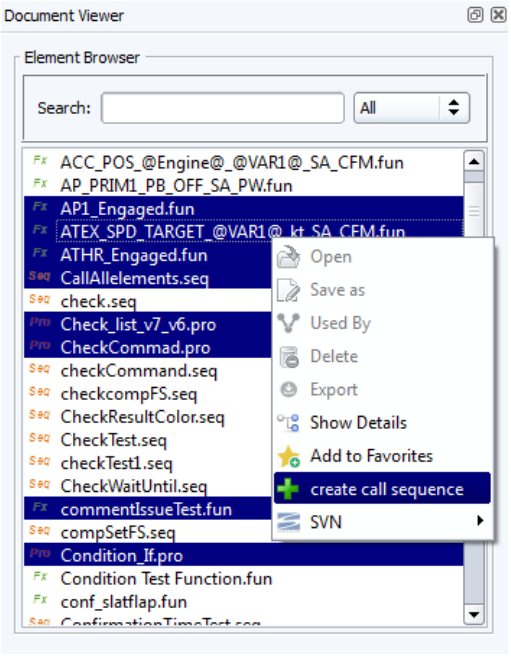


Figure: "Create Call Sequence" Option

	ACTION	TYPE	COMMENT	TIME OUT
0	call function - AP1_Engaged.fun	Action		--
1	call function - ATEX_SPD_TARGET_@VAR1@_kt_SA_CFM.fun	Action		--
2	call function - ATHR_Engaged.fun	Action		--
3	call procedure - CheckCommad.pro	Action		--
4	call procedure - Check_list_v7_v6.pro	Action		--
5	call function - commentIssueTest.fun	Action		--
6	call procedure - Condition_If.pro	Action		--
7	call function - TETS_VAR_FUN.fun	Action		--
8	call object - TEST_VAR_OBJ.obj	Action		--

Figure: Editor Page after selecting the option.

If any invalid call is found a report will be generated as shown below:

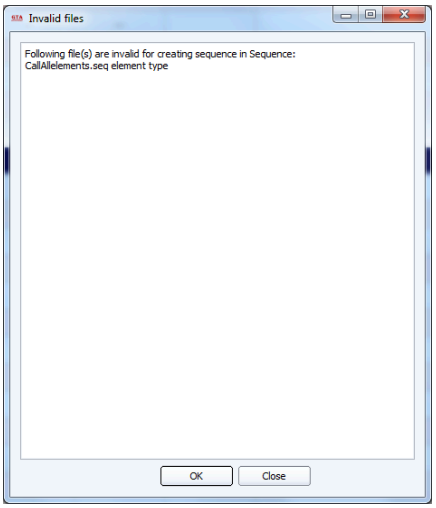


Figure: Error Report window for any invalid Call found.

Update validation status
Allows the user to update validation status for selected element

Open containing folder
Folder which contains the selected element will be opened in file explorer

Add to Procedure Dashboard Launcher
An option to launch the Procedure dashboard dialog on which the execution list of procedures can be maintained and then clicking on Launch button will launch the GTAController Procedure dashboard.
This option is only enabled if the GTAController exe path, AutoLauncher Watch Folder path, AutoLauncher Config File path are set correctly in the Tools -> Configuration settings of GTAEditor.

3.6. Section - ‘Log Window View’

When a procedure is selected and the Compatibility Check is executed, the Log Window opens, displaying any warnings and errors encountered during the Compatibility Check for the chosen procedure. The window provides two key filters: “Log Level” and “Type”.

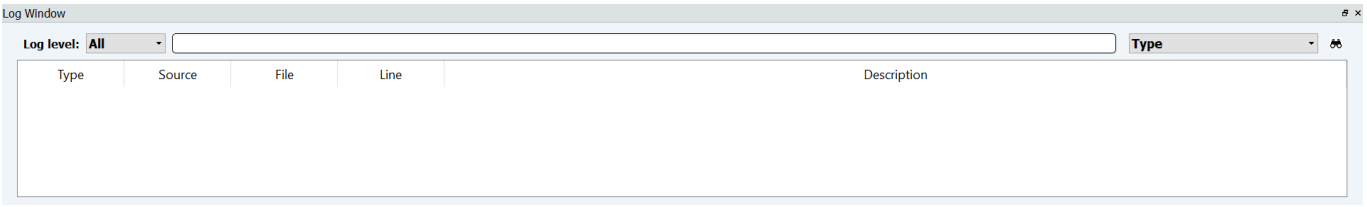


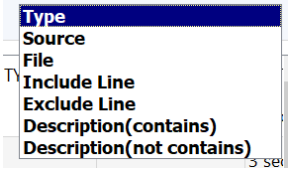
Figure: ‘Log Window” View

- “Log Level” filter combo box:
  - *All*: Displays all log entries, regardless of severity.
  - *Error*: Filters and displays only errors.
  - *Info*: Filters and displays informational messages.
  - *Warning*: Filters and displays only warnings.

- “Type” filter combo box:

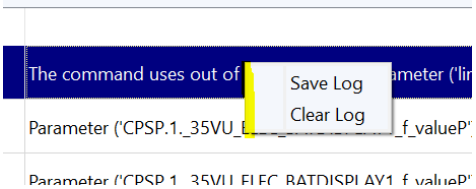
The Type filter provides multiple filtering options based on specific columns in the log:

- *Type*: Filters log entries based on the Type column in the log.
- *Source*: Filters log entries based on the Source column in the log.
- *File*: Filters log entries based on the File column in the log.
- *Column*: Filters log entries based on any specific Column in the log, allowing you to focus on particular fields (e.g., Type, Source, File, or any other column in the log).
- *Include Line*: Displays only log entries corresponding to the line numbers specified in the search window. This filter supports regular expressions to include multiple line numbers (e.g., "1|11|111" or "1").
- *Exclude Line*: Excludes log entries corresponding to the line numbers provided in the search window. Regular expressions are supported to exclude multiple line numbers (e.g., "1|11|111" or "1").
- *Description (Contains)*: Filters log entries to include only those containing specific text in the Description column. This filter supports regular expressions for matching multiple strings (e.g., "editor|database" or "database").
- *Description (Not Contains)*: Filters log entries to exclude those containing specific text in the Description column. It also supports regular expressions to exclude multiple strings (e.g., "editor|database" or "database").



These filters allow users to apply more granular control over which log entries are displayed, enabling filtering by specific columns (Type, Source, File), line numbers, or keywords within the description.

Log displayed in this window should be saved or removed using the right click button. A pop-up will appear with the ‘Save Log’ and ‘Clear Log’ functions:



3.7. Section - ‘Result View’

After execution of a procedure or a sequence on the test bench, the EMOTEST scheduler generates a log for the executed element. This log is post processed by the GTA editor and the results with ‘OK’,’NOK’ for Check Command, and ‘ ✓ ’,’ X ’ for Action commands are provided in the editor’s ‘Result’ view and also the possibility to export the results in an HTML based LTRA is provided.

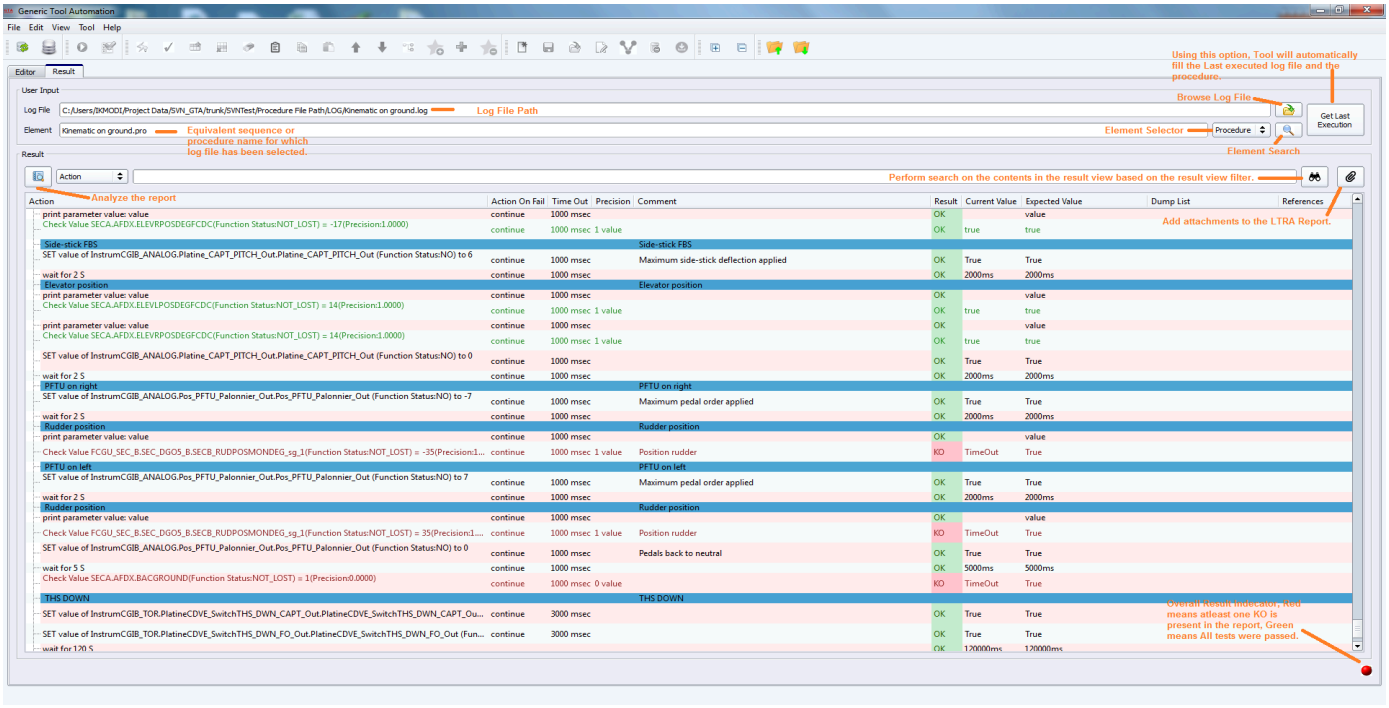
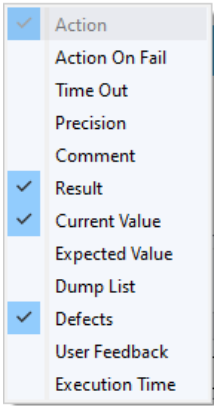


Figure: ‘Result View’ HMI

The ‘Result’ view contains columns with names as shown in the list below. Recent updates introduced the purpose and conclusion items within each procedure. The user can manage them on both Editor and Result View.



- After test execution, in order to view the test results of each procedure/sequence step, the user must switch to Result view
- In User Input section, user must enter the log file path using  or direct text box editing and choose the element for which the log file was generated. Users may use the ‘Element Selector’ and ‘Search element’ buttons for element selection.
  (*Note: GTA allows execution of only 2 elements: a sequence or a procedure*)

- User must click on the **Analyze button**  to :
    - o enable GTA editor to perform a log post processing,
    - o display the results in the ‘Result view’ as shown in figure above
    - o make possible the modification of the associated purpose and conclusion which is located at the bottom part
    - o and update the csv file involved in existing ForEach if the write_back option was selected

  Note: If the input log file is not the correct match for the input element, the analyze button press will lead to a mismatch warning.

- User can also search specific contents in the result view by using the 

filter dropdown and clicking on the search button  .

- After clicking on the analyze button, User may export the results to an HTML LTRA report by clicking on Tool>Export>LTRA option.

- If any Call command is collapsed the content of the call command will not be exported in the LTRA report.

- If any command is unchecked the command will not be exported in the LTRA report.

- User can also add desired attachments to the LTRA report by clicking on  button.

- If any call command is collapsed in “Result View” HMI, then on export the internal details of that call command will not be available in the LTRA/LTR report.

- If the log path is not available in the “Result View” HMI, and user selects the button “Analyze” for any procedure or sequence, then it can be exported as LTR without any results associated to it.

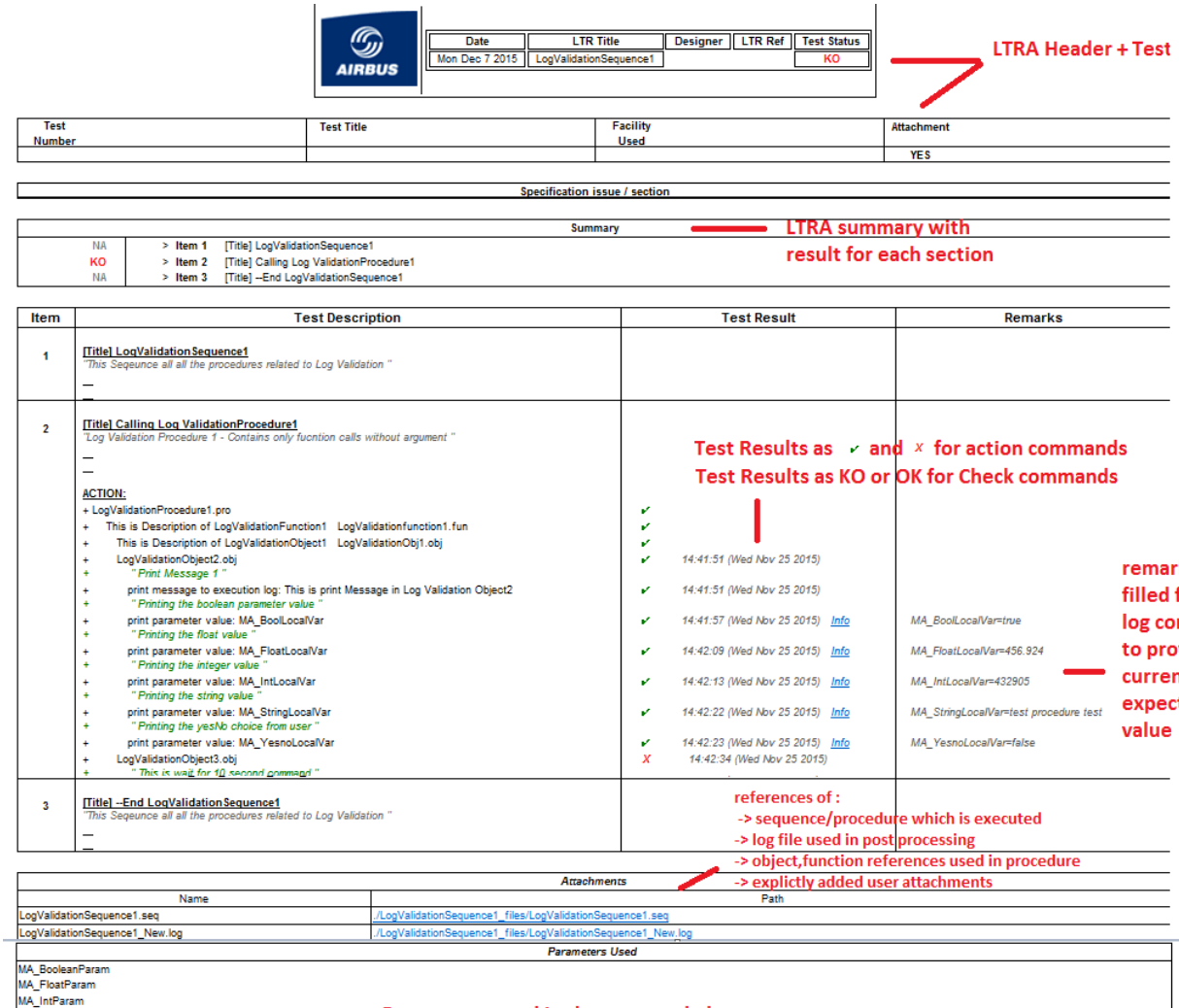


Figure: Exported LTRA format sample with explanation

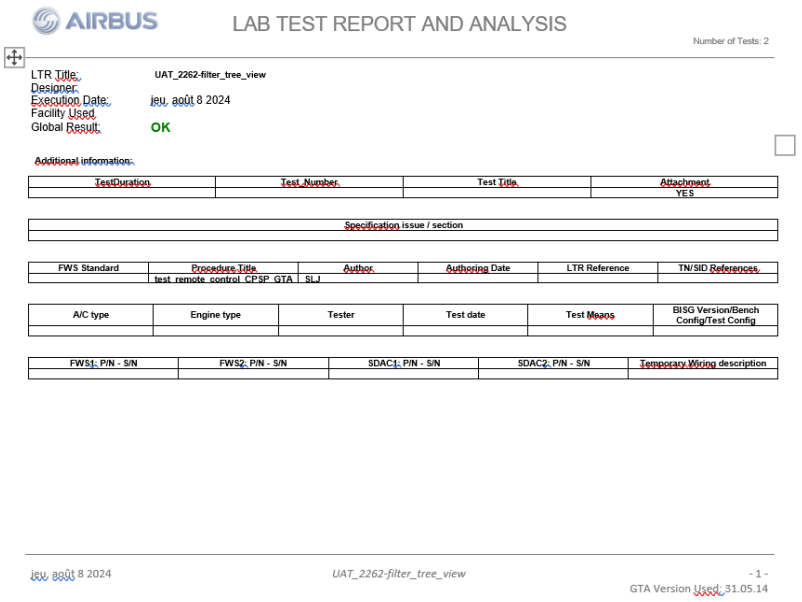


Figure: New LTRA format - Header

AIRBUS LAB TEST REPORT AND ANALYSIS

Number of Tests: 2

Summary		Comments
NA OK	> Item 1 - Dummy title : Procedure Start > Item 2 - PROCEDURE	

Figure: New LTRA format - Summary

AIRBUS LAB TEST REPORT AND ANALYSIS

Number of Tests: 2

Item	Line Number	Test Description	Test Results
1		< DUMMY TITLE : PROCEDURE START	NA
2		< PROCEDURE	OK (15:24:38)
	[1]	TEST PROCEDURE	✓ (15:24:38)
	[2]	> CALL - /UAT/OBJ/test_obj_2262.obj *Created through drag and drop into editor*	OK

Figure: New LTRA format - Test details

3.8. List of commands

Here after, a list of commands to be performed through GTA Editor.

CALL Object	CALL Function	CALL Procedure	CALL Parallel
CHECK BITE Message	CHECK FWC Warning	CHECK Value	CONDITION If
CONDITION Else	CONDITION Do While	CONDITION While	CONDITION For Each
EXTERNAL TOOL	INIT PIR List Parameter	MANUAL ACTION	MATHS
PARALLEL CALL	RELEASE PIR List Parameter	PRINT Message	PRINT Parameter value
PRINT Table	PRINT Time	RELEASE VMAC	SET
TITLE	WAIT For	WAIT Until	

3.8.1. “CALL”

To allow modularity during procedure creation, GTA Editor allows users to create objects and functions that can be called or invoked from a procedure.

- In a “Sequence” file, “CALL” action enables the user to invoke a procedure, a function or an object.
- In a “Procedure” file, “CALL” action enables the user to invoke a function or an object.
- In a “Function” file, “CALL” action enables the user to invoke another function or an object.

The “Function” file is different from other GTA file as it allows users to create functions that have parameters containing placeholders. These placeholders will be replaced during a “Function call” with user-defined value.

Before explaining “CALL” functionalities, it is important to know how to add a placeholder in a Function.

3.8.1.1. Placeholder in a function

To add a placeholder in the parameter name, user must write the following syntax:

@placeholder_name@

If the parameter name is ‘InstrumCGIB_ANALOG.Platine_ENG1.Platine_ENG1’ and the user wants to create a generic function using this parameter, such that the same function can be used for several engines.

In a command, user can import the parameter from search widget and modify it as below:

Original name of the parameter:
InstrumCGIB_ANALOG.Platine_ENG1.Platine_ENG1

Modified name with placeholders:
InstrumCGIB_ANALOG.Platine_ENG@Engine_number@.Platine_ENG@Engine_number@

Note: Placeholders are applicable for following actions: Set, Check, Condition, Wait, Manual Action, Release & Print value

3.8.1.2. CALL Function

The “CALL” Action Selector provides an element selector dropdown and a search item button.

On selecting the search item for a specific element type, the saved element list of the specific chosen type is provided to the user.

Thereafter the user may choose an element and the tool creates a call test step for the called element.

The flow for call function is shown below, same is applicable for Call Procedure and Call Object.
Note: No placeholder’s inputs are required for Call Procedure and Call Object, only for Call Function.

Figure: Call Action – Function Selection

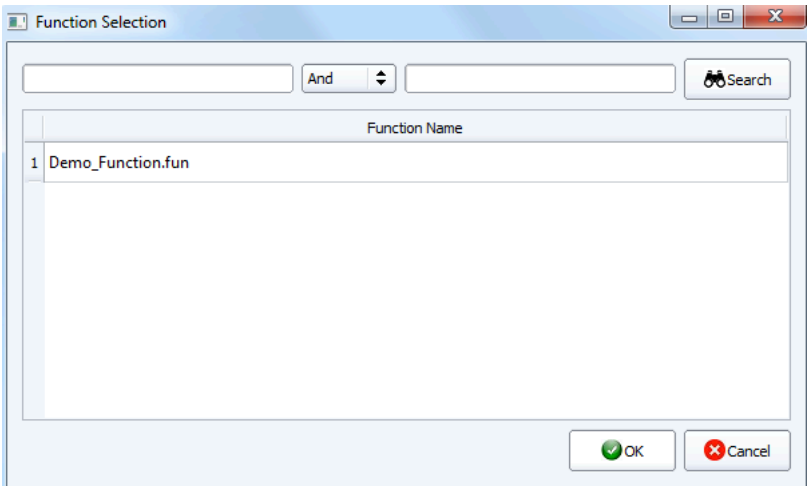


Figure: Saved Function List

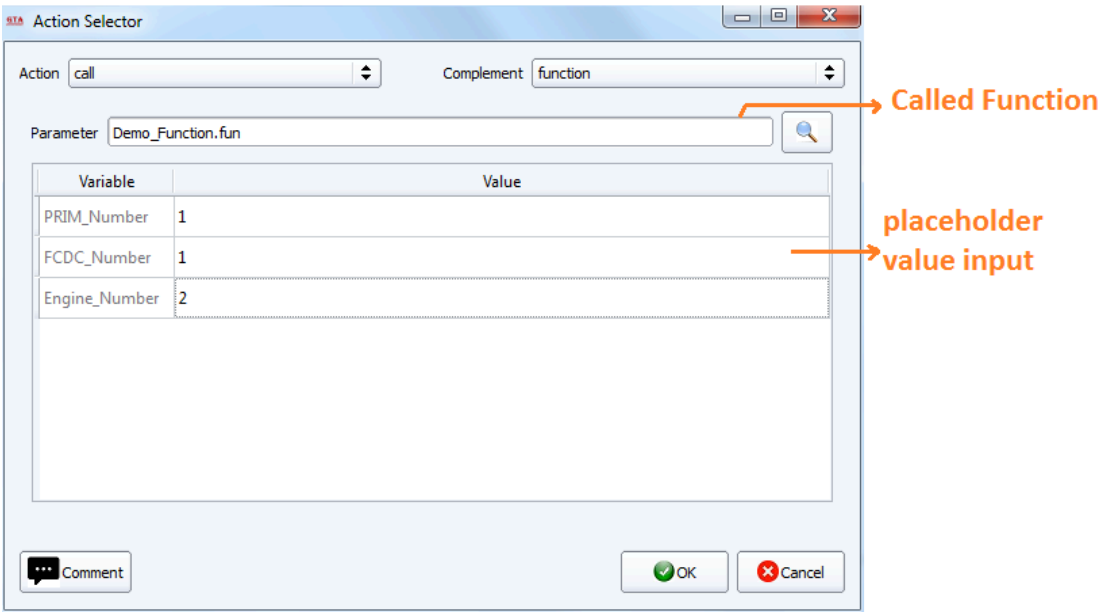


Figure: Call Function Placeholder input

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
call function - Demo_Function.fun	Action	Sample Function with placeholder			

Figure: Call Function Test Step

3.8.2. "CHECK"

The "CHECK" command can be invoked in any of the following ways:

- ✔ Double Click on the 'TYPE' Cell on a row
- ✔ Right Click on the Row > Editor > Check

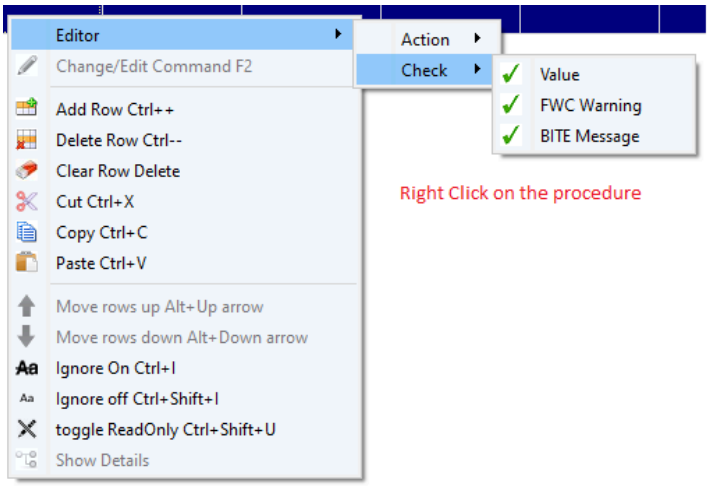
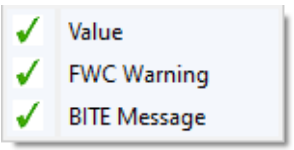


Figure: Ways to access "Check" command

The "Check" command allows user to perform the following check types:



3.8.2.1. Check Value

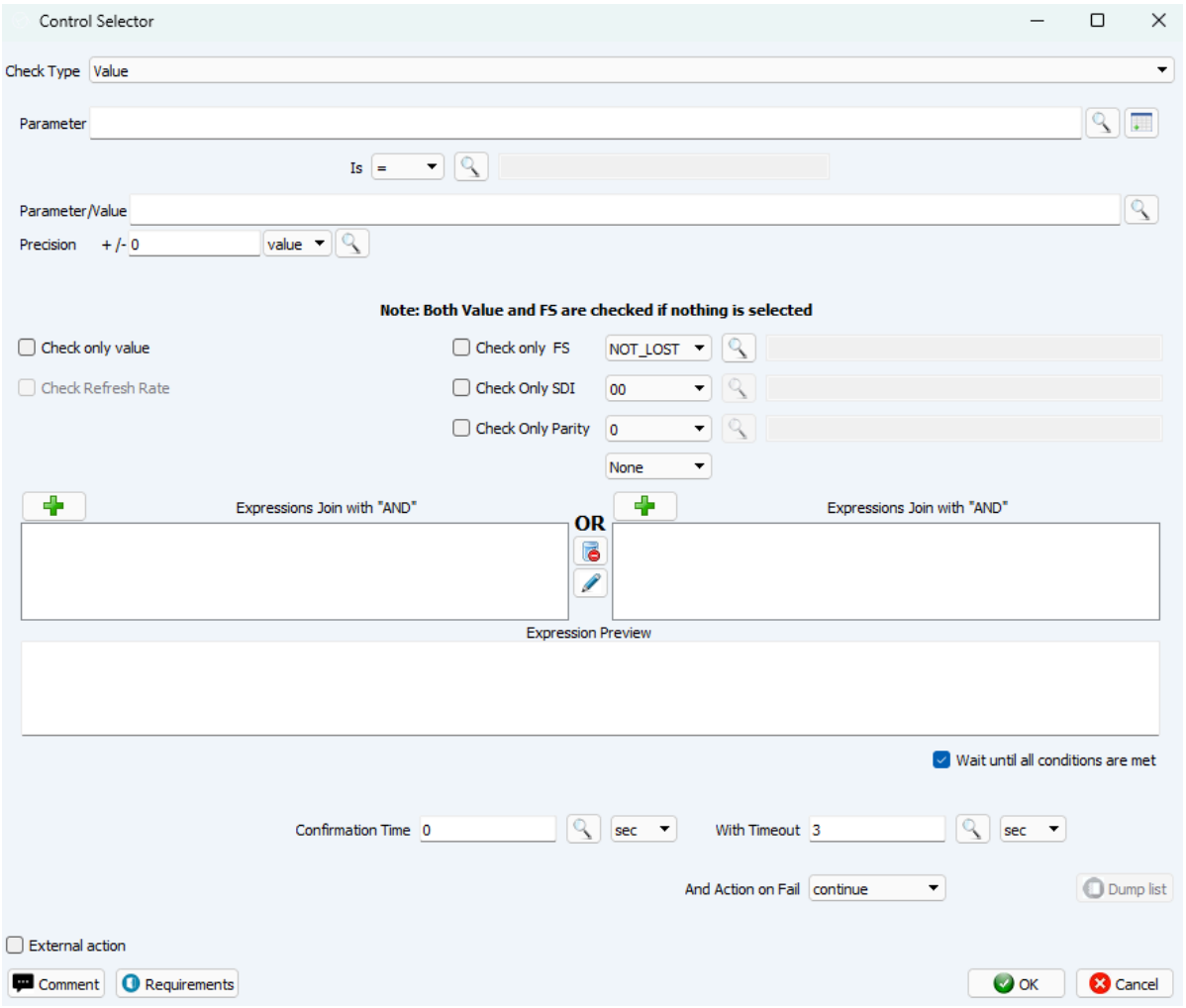
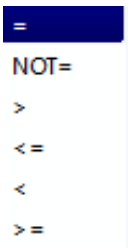


Figure: CHECK Value HMI

The CHECK Value allows user to perform the following cases with different operands:



- Check if the value acquired for a Bench Parameter/Local variable is a ‘constant value’ or ‘value acquired by another Bench Parameter/Local Variable’.
- Only Check the Functional Status of a Bench Parameter.
- Add multiple checks in one control selector HMI.
- Check SDI
- Check PARITY

For an edition, the last updated expression is selected automatically.
If an expression is deleted, the automatic selection is done on the item of the last row.

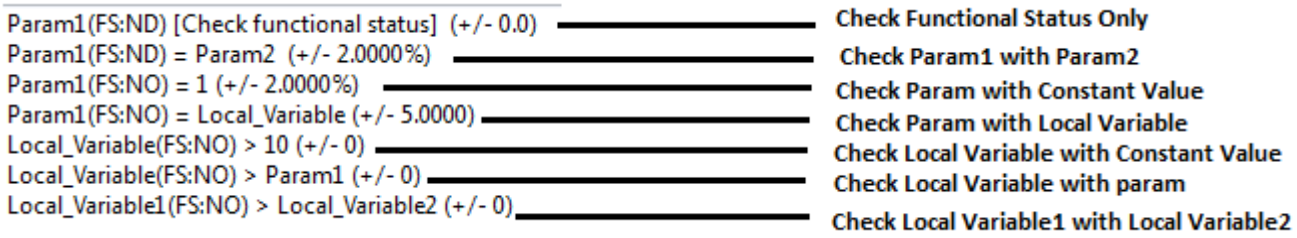


Figure: CHECK Value possibilities sample

3.8.2.1.1. Check Functional Status

The CHECK control selector allows user to check the Functional Status of a bench parameter by selecting the functional status from the following list:

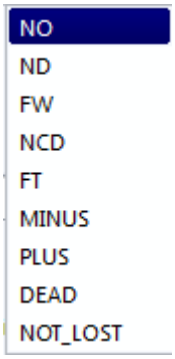


Figure: CHECK Value Functional Status list

User has several options while performing a CHECK Value:

Functional Status (FS) of a parameter can be checked along with value checks by choosing the expected FS from the list and also providing data for 'Parameter/Value'.
Opting to check only the Functional Status of a parameter and not its value by selecting the "CHECK Only value" option and choosing the expected FS from the list.
Only value can be checked, and FS checks can be skipped by selecting "CHECK Only value" option.
CHECK Only Refresh Rate option can be checked.
CHECK Only SDI option can be checked.
CHECK Only Parity option can be checked.

3.8.2.1.2. Check Value HMI Details

1) **User Inputs for each check expression**: Portion of the HMI (as in figure below) allows user to provide specific input in regard to Parameter

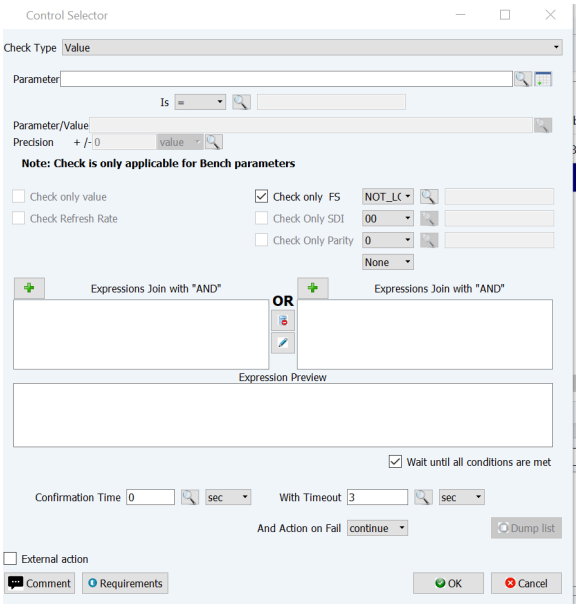


Figure: CHECK Value HMI – User Inputs

2) Check value for a list of parameters (with an input an CSV file)

Check 'Value' HMI allows users to check a list of parameters by using a CSV file as an input.
The mandatory format for input CSV file is as below:

- ✔ CSV File with **AC_Parameter ; Value ; Functional Status ; Precision ; Condition**

Note: User must maintain above order of input fields
Note: Please the recommendations to fill the '.csv' file find in the §3.7.3.5.1

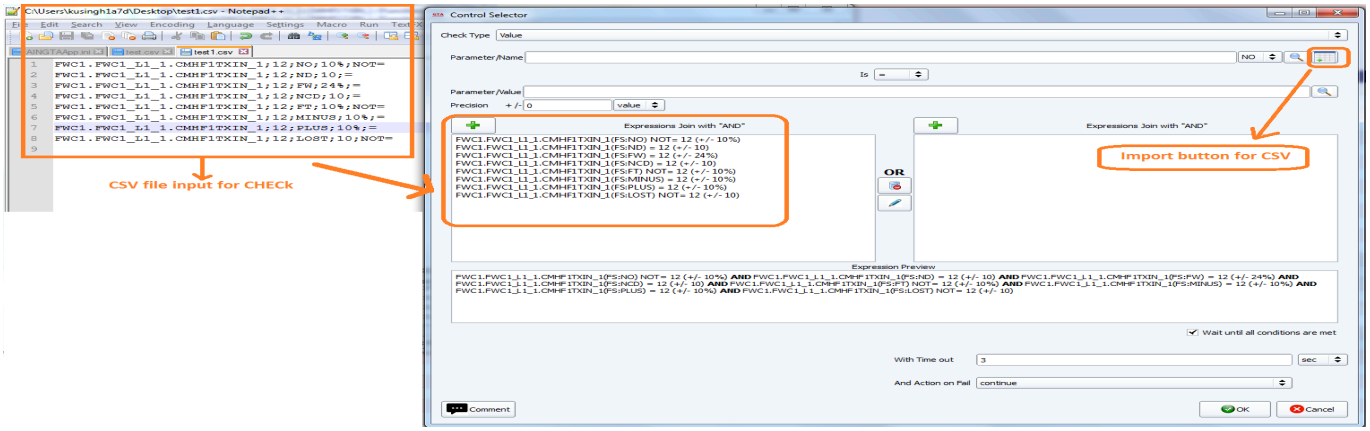


Figure: Check 'Value' HMI – CSV input

Import button  allows the user to browse the CSV input file, If the CSV format is as per required definition, the parameters and its details imported to the AND list as shown in figure above.

3) **Data Presentation Layout**: Portion of the HMI (as in figure below) allows user to add input data in a specific layout to the check conditions based on the expected logic.

- ✔ Layout is divided into 2 blocks.
- ✔ E1 = AND (All Check expressions added to the left block)

- ✔ E2 = AND (All Check expressions added to the right block)
- ✔ Final expression = E1 OR E2

Note: This method covers most of the frequently used logics, however, in the current version, the tool does not allow users to create check expressions not possible using the current method.

Use  button to edit a selected check expression in any block

- ✔ First select an expression in a block
- ✔ On selection the check details will be reflected on the user input's section.
- ✔ Update the inputs as desired and
- ✔ Click on  to ensure that the change is reflected into the selected expression in the block.

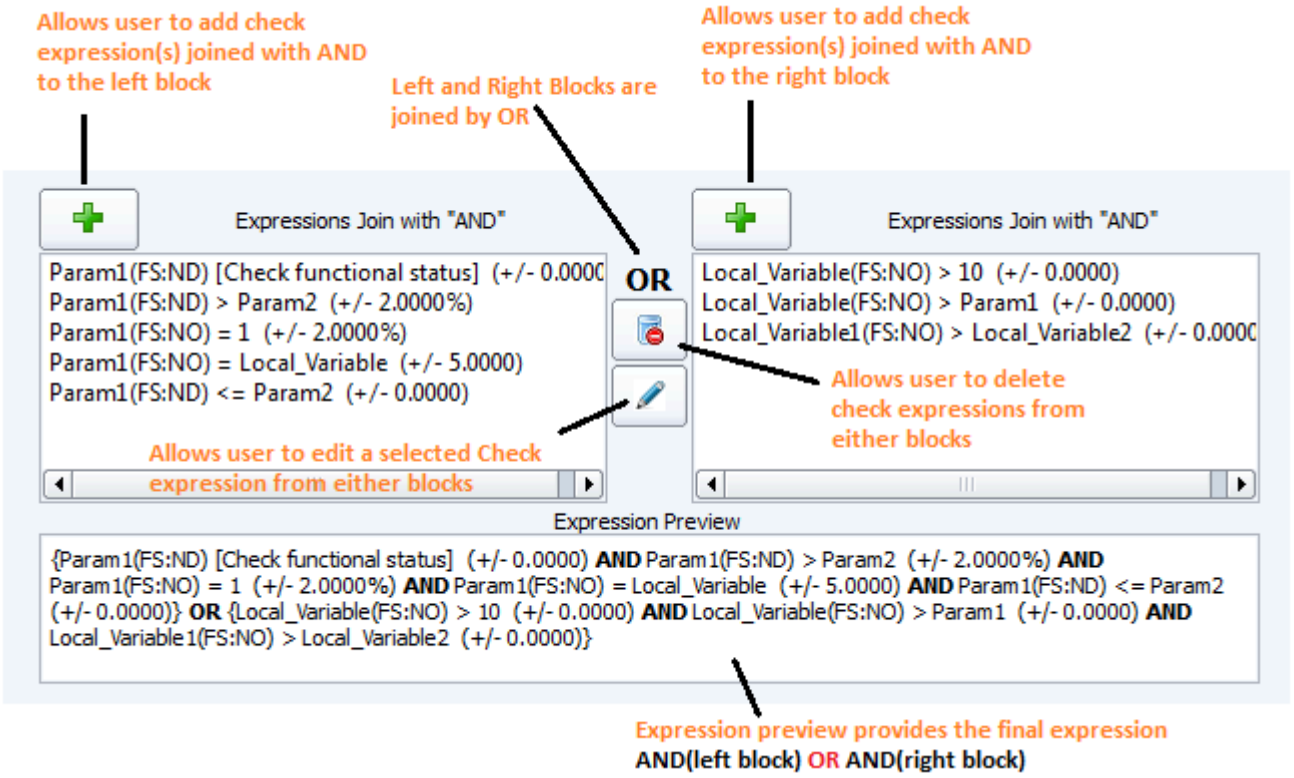


Figure: Check 'Value' HMI – Data Presentation Layout

The figure above provides a quick overview of the data presentation layout.

4) User Inputs common for all check expression(s):

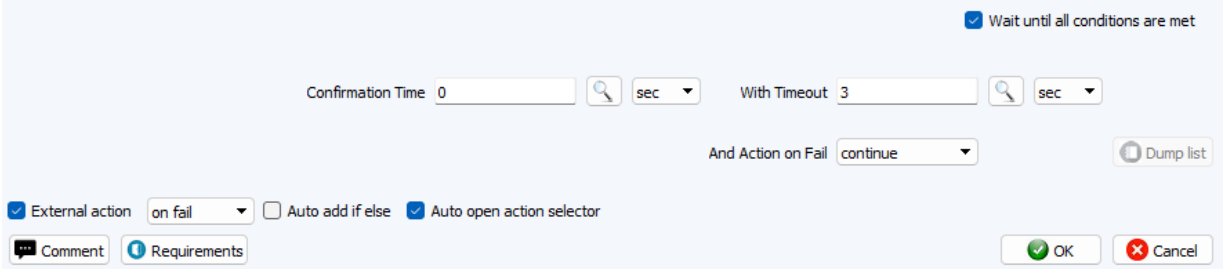


Figure: CHECK Value HMI – Common Inputs

'Wait until all conditions are met': This option is applicable by default and ensures that the execution waits until Timeout expires or until all conditions are met (*Note: Whatever event occurs first*).

"Confirmation Time": The check condition is OK only if it is valid during X seconds of the confirmation time. Condition = TRUE + "Check if condition persists for confirmation time".

"**External action**" This checkbox allows user to automatically input a condition, which depends on the result status of the current check, directly below the check in procedure, allowing for reaction with external action such as external tool.

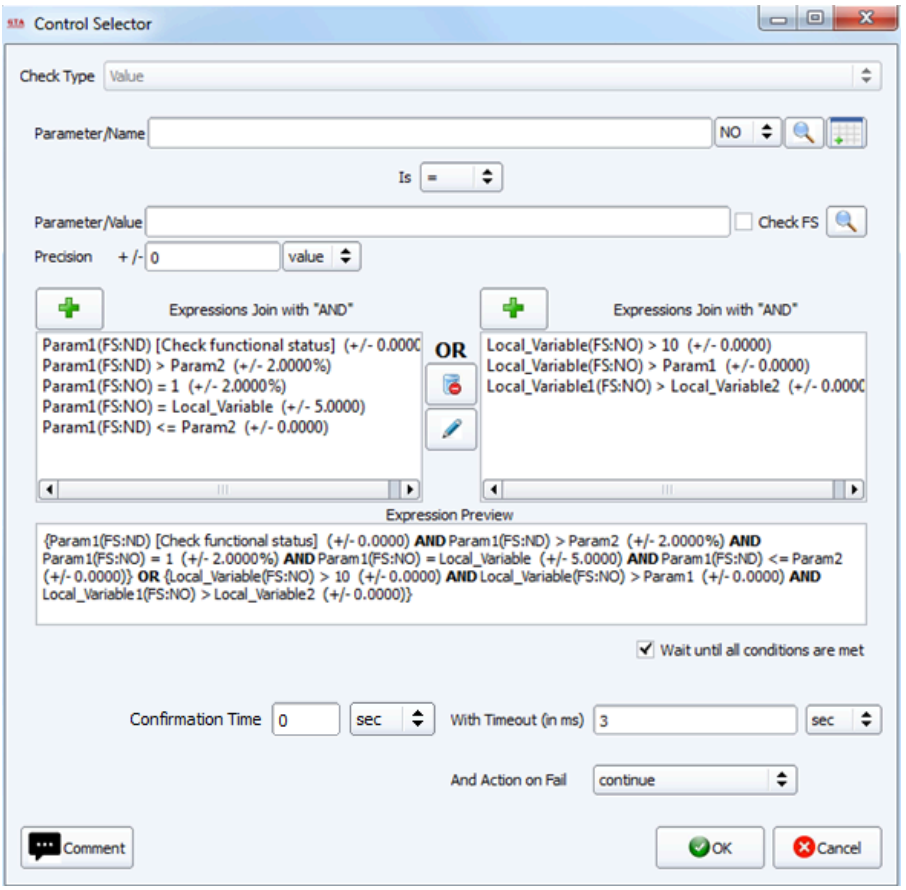


Figure: CHECK Value HMI – Data filled

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
Check Value Param1(Function Status:ND) [Check functional status] (Precision:0.0000) AND Param1(Function Status:ND) > Param2 (Precision:2.0000%) AND Param1(Function Status:NO) = 1 (Precision:2.0000%) AND Param1(Function Status:NO) = Local_Variable (Precision:5.0000) AND Param1(Function Status:ND) <= Param2 (Precision:0.0000) OR Local_Variable(Function Status:NO) > 10 (Precision:0.0000) AND Local_Variable(Function Status:NO) > Param1 (Precision:0.0000) AND Local_Variable1(Function Status:NO) > Local_Variable2 (Precision:0.0000)	Check		3 sec	--	continue

Figure: Sample CHECK Value step on Editor View

3.8.2.2. Check FWC Warning

The CHECK FWC Warning allows users to check generation of FWC Messages by the FWC simulation only.
Note: Real Flight warning display related check could be directly performed using GTA's Image recognition module and 'IRT action'.

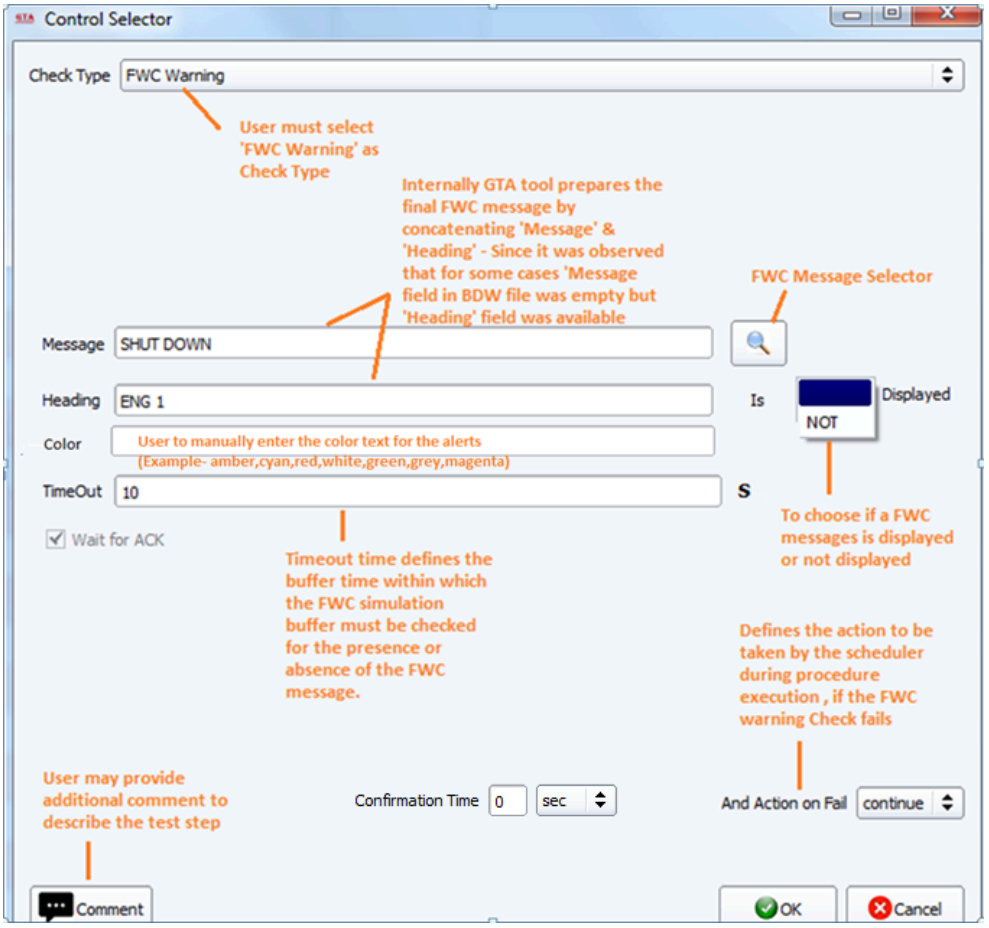


Figure: CHECK FWC Warning HMI

Note: The message selector HMI as shown in figure below provides data based on parsing of flight warning BDW or equivalent file provided by the design office for each program.
If user does not find a specific warning using the message selector widget, he may directly edit the 'Message' or 'Heading' textbox.

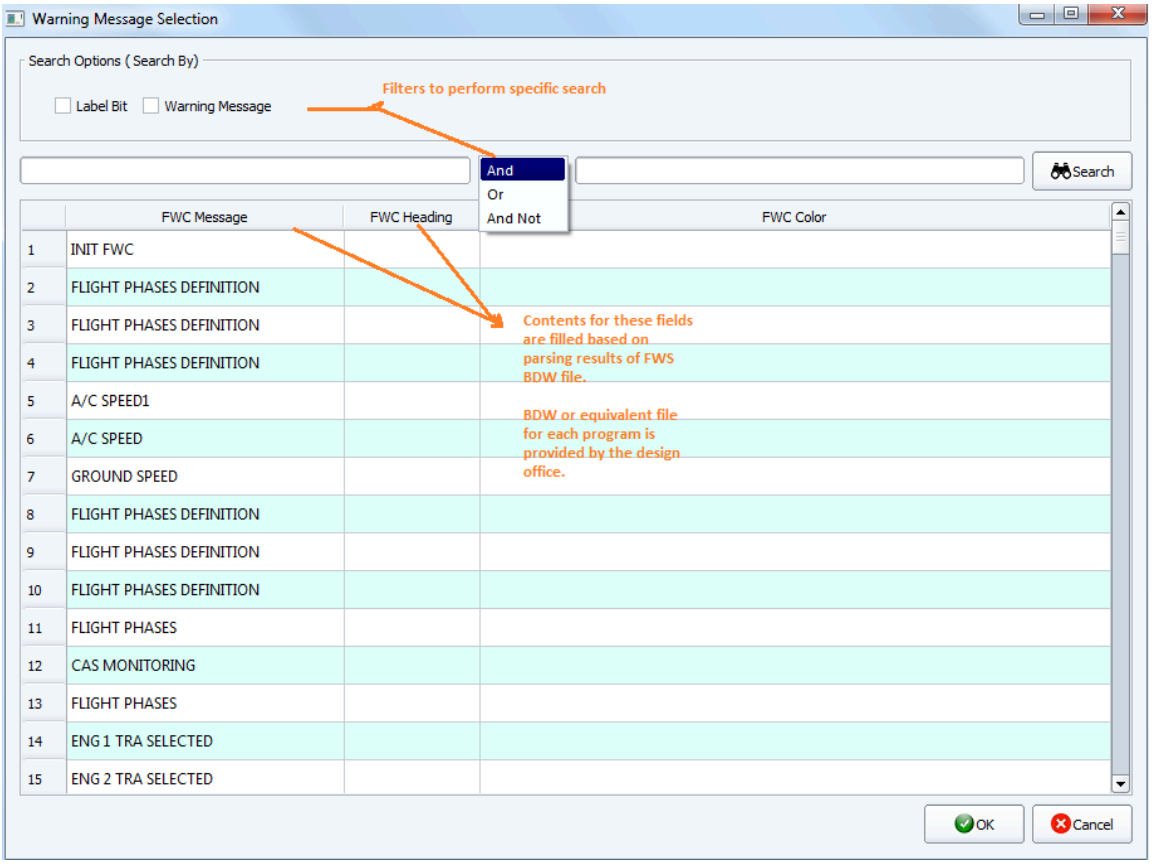


Figure: CHECK FWC Warning –Message Selector

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
check FWC Warning Message="SHUT DOWN" and Heading="ENG 1" is displayed	Check				continue

Figure: CHECK FWC Warning step on Editor View

The above figure shows the display of a CHECK FWC Warning test step on the Editor View.

Note: The FWC_BITE_logger tool must be operational to perform tests related to checking 'FWC message'.
During procedure execution the FWC simulation logged messages are buffered and retrieved by the FWC_BITE_logger .The user expected FWC messages as per inputs in the GTA procedure are then cross checked in the FWC buffer and the results are provided to the EMOTEST log for post processing by GTA editor.

3.8.2.3. Check BITE Message

The CHECK BITE Message allows users to check generation of BITE maintenance messages in the BITE simulation only.
Note: Real display related checks could be directly performed using GTA's Image recognition module and 'IRT action'.

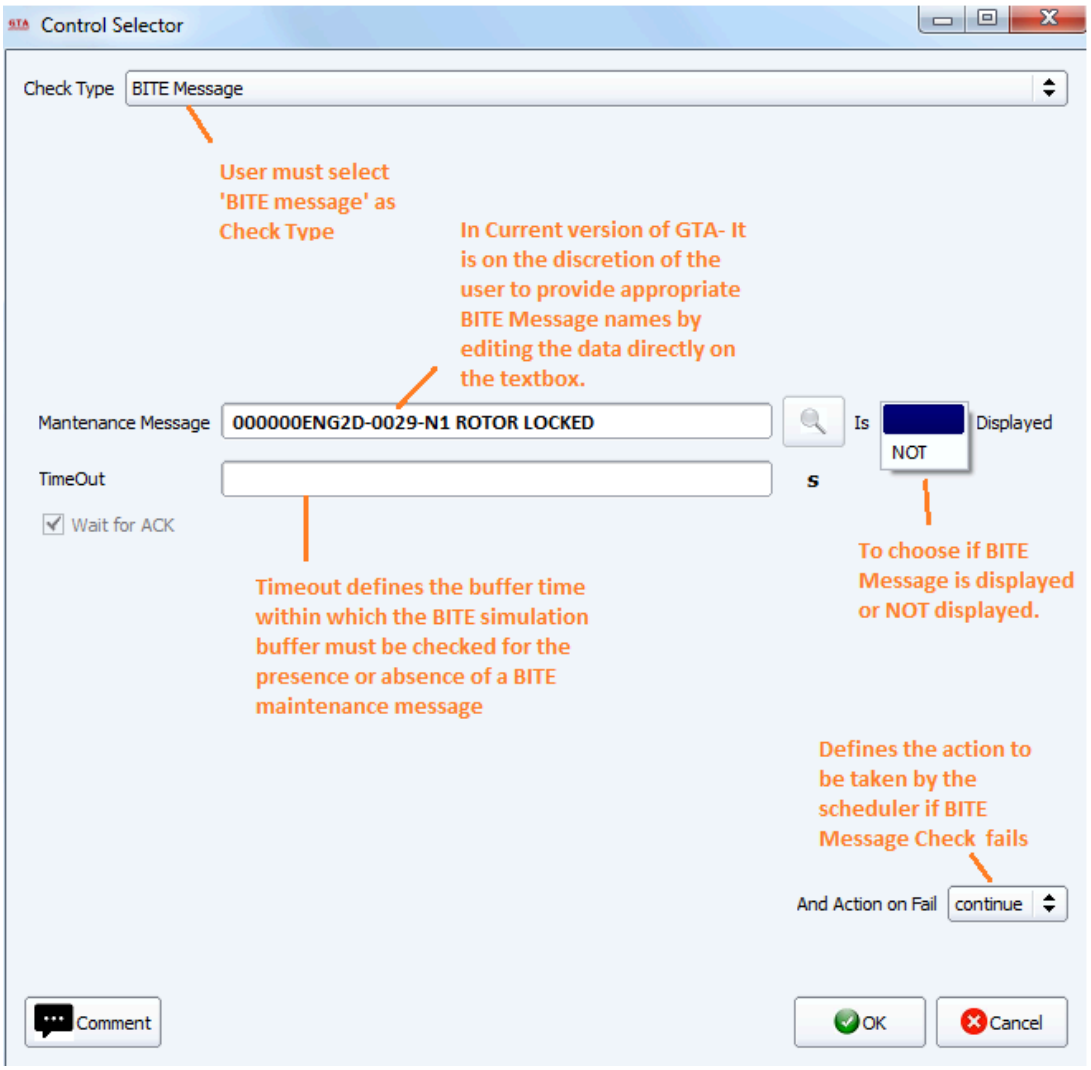


Figure: CHECK BITE Message HMI

Note: The BITE message selector HMI is disabled in the current version of the tool, and users must provide BITE maintenance message by directly editing on the 'Maintenance Message' textbox.

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
check BITE Message "User defined BITE message" is displayed on Maintenance Message	Check		--	--	continue

Figure: CHECK BITE Message step on the Editor View

The above figure shows the display of a check 'BITE Message' test step on the editor.

Note: The FWC_BITE_logger tool must be operational to perform tests related to CHECK BITE Message.
During procedure execution the BITE simulation logged messages are buffered and retrieved by the FWC_BITE_logger .The user expected BITE messages as per inputs in the GTA procedure are then cross checked in the BITE buffer and the results are provided to the EMOTEST log for post processing by GTA editor.

3.8.3. “CONDITION”

To allow logic flexibility during procedure creation, GTA Editor allows users to add conditional statements at test steps involving the use of Bench Parameters as well as Local Variables.

The Conditional Expressions include the following conditions:

- If / Else / EndIf
- While
- Do While
- For Each

Note: GTA Editor does not put any restriction on the levels of conditional nesting that can be present in the procedure.

The Complement dropdown permits users to select If/Else/EndIf or While or DoWhile or ForEach conditions.

3.8.3.1. CONDITION If

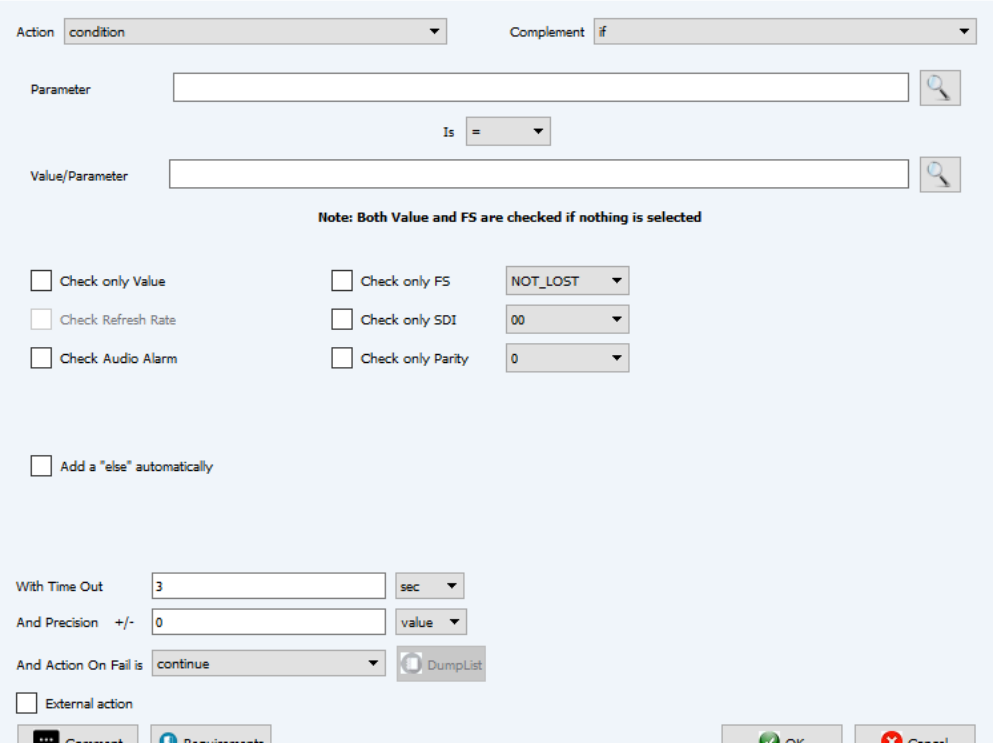


Figure: CONDITION If HMI

Conditional operators are provided under ‘IS’ dropdown.
“Parameter” textbox may include both Bench parameters and Local Variables.
“Value/Parameter” textbox may include Bench parameters, Local Variables & Constants.

User can use the parameter selector button  to perform searches for bench/local parameters.

The “Add a else automatically” checkbox allows the user to insert automatically an ELSE step while adding the IF condition.

The external action checkbox will allow the user to automatically open and insert external action into the IF statement.

“With Timeout” textbox allows the user to provide a time value in seconds or milliseconds.
Note: To allow user defined timeout for the command execution. If an action is not executed within the user defined timeout, the test step is automatically considered as ‘KO’. In such a case based on ‘Action on Fail’ preference, the Procedure execution ends or continues with the next test step.

“And Precision” textbox allows users to provide expected precision in value or percentage.
During execution, user defined values for precision have to be considered as a range within which parameter value check must be performed for the test step to be considered ‘Passed/OK’.

After selecting a specific parameter from the search widget and filling the rest of the fields, the Condition If HMI looks similar to the figure below.

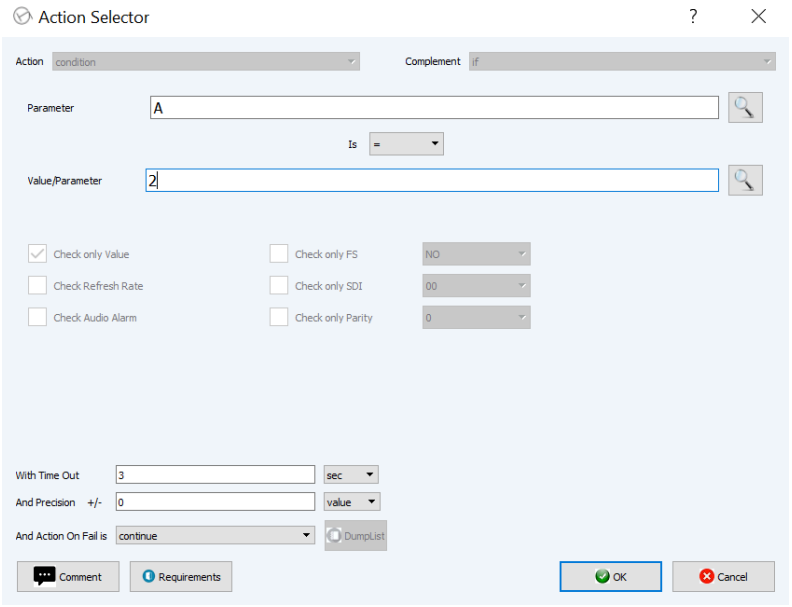


Figure: CONDITION If HMI - Filled

On data submission the “CONDITION If” can be seen in the Editor View.

To add an “Else” into the “If- EndIf” expression user just need to explicitly select “Else” in the complement dropdown at a specific row inside the condition scope (between dedicated rows).

Other actions (including nested conditions) can be added as test steps within condition scope by using add row and add action feature of GTA Editor.

Figure: CONDITION If/Else/EndIf

Note: If an user clicks on any of the Condition syntax rows ('CONDITION','ELSE' or 'END CONDITION' Or 'While', 'End While', or 'Do','Do While End'), the Editor automatically highlights the scope of the condition expression as below:

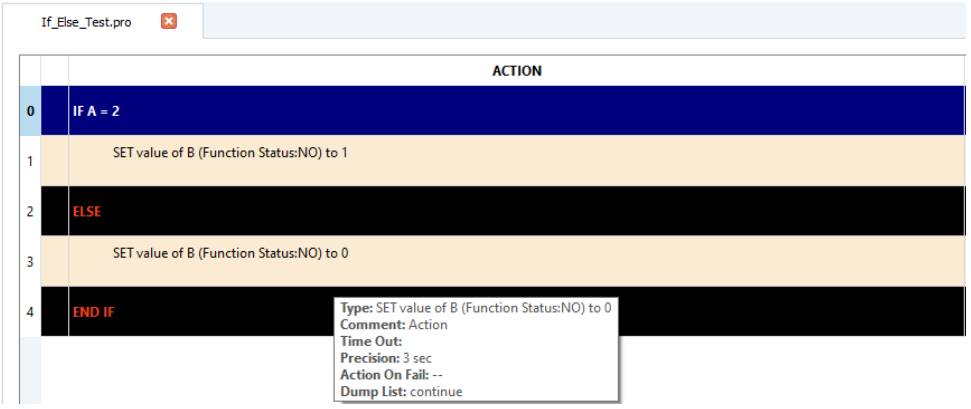


Figure: Condition scope highlighting Feature

The 'While' Condition can be used similar to an 'IF' condition by selecting 'While' command from the complement selector.
'While' Command ensures that the next set of test steps (in while – End while block) would continue execution in a loop until the while condition is satisfied.

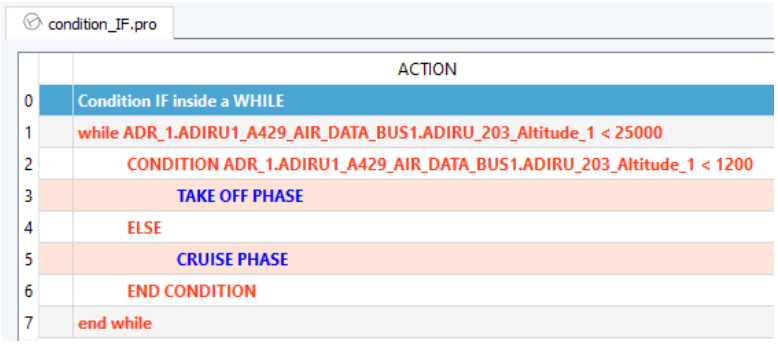


Figure: IF condition within a While Condition

Note:

- *If Copy, Delete Row and Clear Row functionality is used on 'WHILE Condition' expression row, the functionality is applicable to all the contents in the scope of the expression.*
- *Delete Row and Clear Row are not applicable to 'end While' and 'END CONDITION' expression rows.*
- *Copy functionality is not applicable to ELSE Row.*

3.8.3.2. CONDITION While

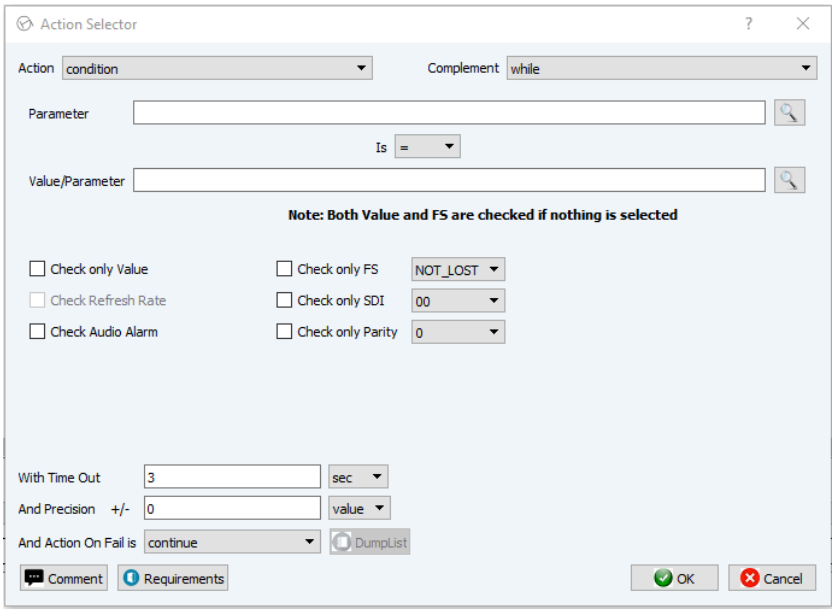


Figure: CONDITION While HMI

3.8.3.3. CONDITION Do While

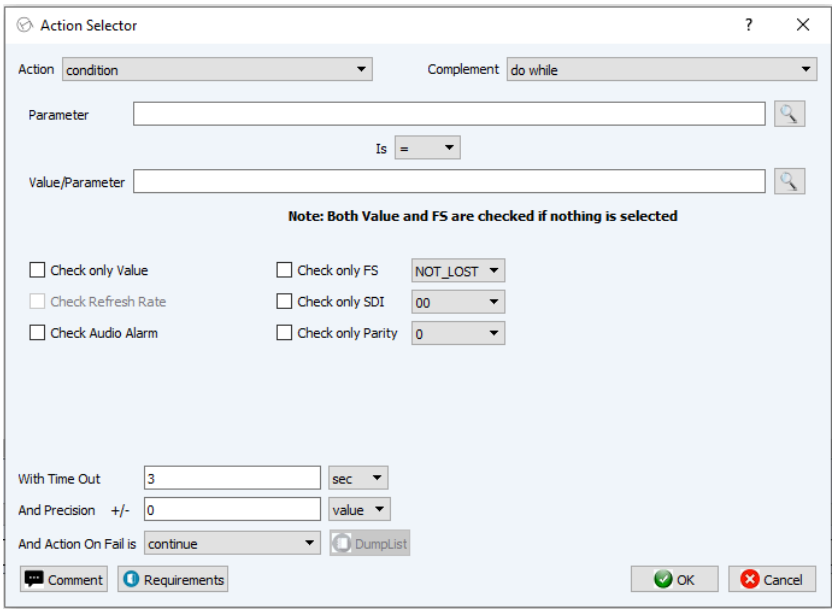


Figure: CONDITION Do While HMI

3.8.3.4. CONDITION Check Audio Alarm (Common)

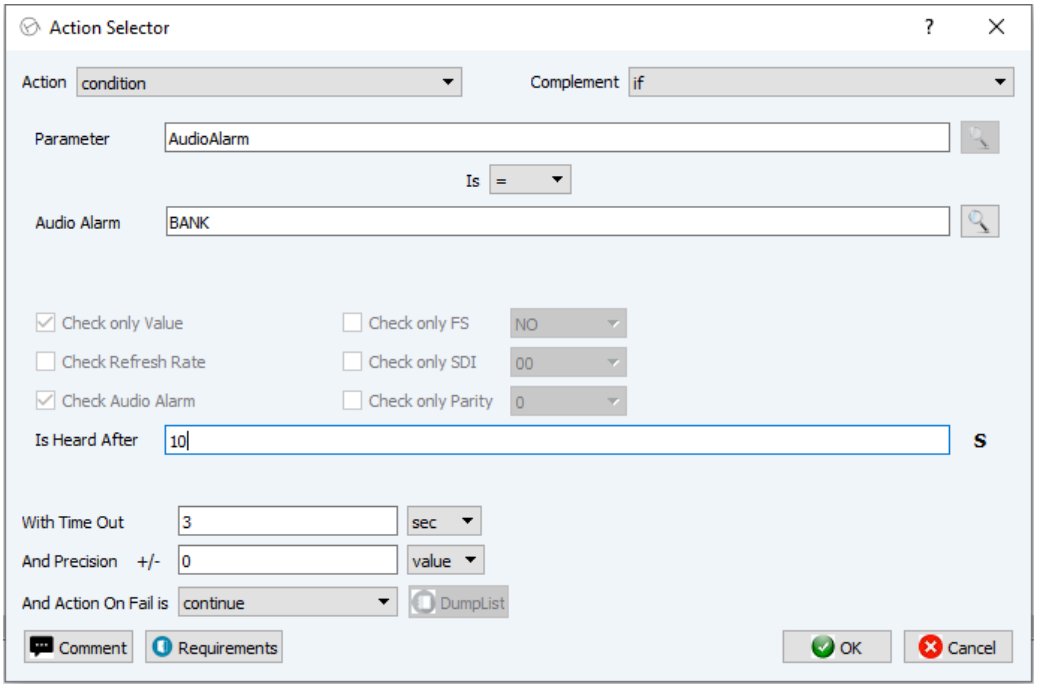


Figure: CONDITION Check Audio Alarm

Note: Conditions (If, While, Do While) HMI are exactly similar with the same features.

3.8.3.5. CONDITION For Each

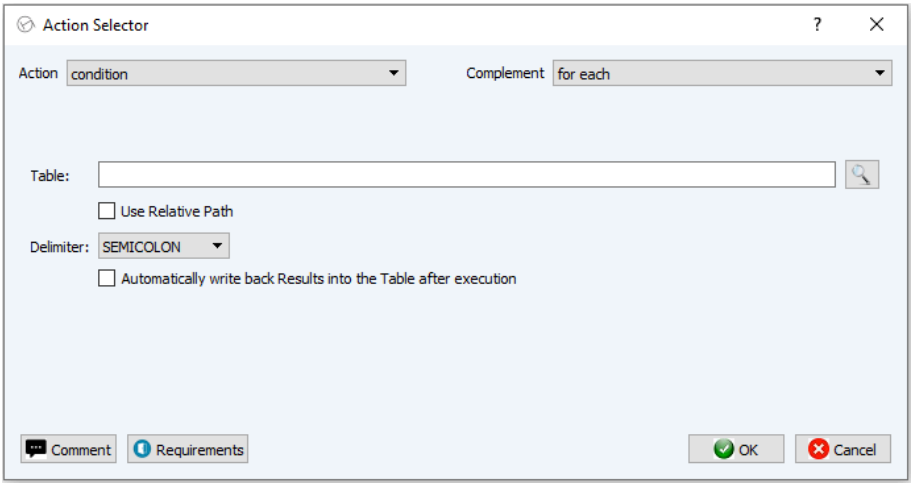


Figure: CONDITION For Each HMI

3.8.3.5.1. Introduction

The FOR EACH Condition is a specific condition action that iterates through a list and allows each variable of the list to be used within its scope.
Whatever the content of the list, it is stored in a file *.CSV extension. Make sure that your files are stored in TABLES/ folder of your Workspace Repository Path. All parameters shall be filled, GTA editor is not able to put default values which might be different from test to test. This may involve csv file update for testers but guarantee that the data sent is the one expected by the user.

Please find hereunder the recommendations to fill a '.csv' file (used for the function 'For each') in order that no issues should be reported during the 'Check Compatibility' function of GTA Editor:

Regarding the column name of the 'csv' file:

- Only '$' and '_' can be used with other characters in the Title
  - '_' cannot be used independently (without other characters)
  - '$' can be used independently (without other characters)
- No other special characters are allowed
- Column name must not have any 'space' character between the words
- Avoid using accents on character (English language)
- 'Parameter' could be replaced by 'Param'
- 'Value' could be replaced by 'Val'
- 'Value' is the only word that is restricted on check compatibility. Even 'value' or 'VALUE' word are allowed

To fullfill the empty cell of the 'csv' file:

- Same recommendations than the ones indicated for the column name
- For column which contain a 'String' type:
  - If you are using the 'AIRBUS' word just to skip the execution of different commands then we can use it in every column (apart from Parameter value column).

    As we don't face any issue in check compatibility so the tester can add conditions correctly and we can add 'AIRBUS'.

  - 'NA' could be used for the "Write back" result column (to distinguish with other columns in which the word 'AIRBUS' is used – but the word 'AIRBUS' can be also used if needed/desired)
- For column which contain a value type ('Integer', Float', Boolean'):
  - Regarding Datatype of the value:

    The datatype is retrieved from the Parameter DataBase. The Check Compatibility expects that the value, entered in the 'csv' file, match with the datatype specified the Parameter DB.

    So, to avoid a warning (reported during the Check compatibility), the value entered always needs to comply with the database type expected.

    But, to facilitate the test procedure using condition to skip the execution of different commands, 'AIRBUS' or 'NA' should be used. Error/Warning will be raised, but they are linked to the line of the 'csv' file.

  - Regarding Min/Max of the value:
    - We do not manage the information provided (if it is indicated or not in the ICD), but :
      - if a min/max values are specified, a value in this range must be used => no error will be reported during the Check Compatibility and the execution of the procedure.

        If the value exceeds this range (min/max), the Check Compatibility will raise an error "Out of range".

      - if no min/max values are specified, the value '999999' could be used => no error will be reported during the Check Compatibility and the execution of the procedure.

- Specific cases:

  For the ‘Timeout’ column, add the unit (eg: ‘30**s**’, ‘5000**ms**’)

The loop does not have a timeout and specific timeouts are handled by actions within its scope. The scope is marked by a ‘FOR EACH END’ statement in the Editor View.

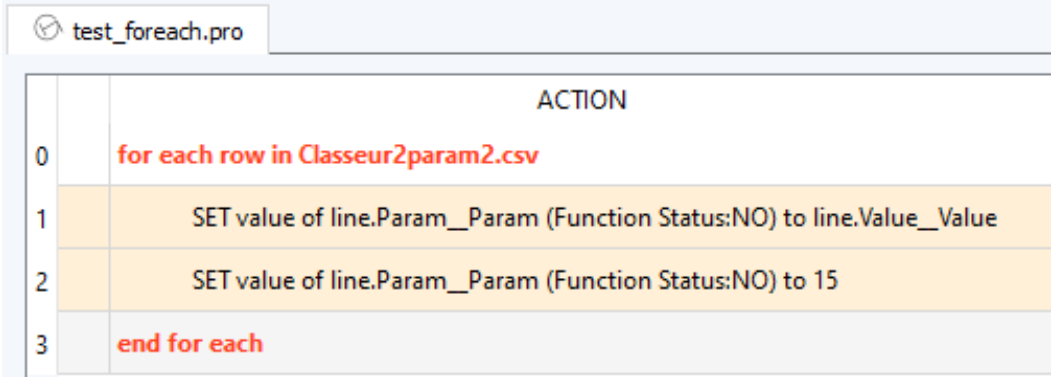


Figure: CONDITION For Each procedure steps

3.8.3.5.2. First use

Setting values from the List can be performed by following the below mentioned steps:

1) Add a “FOR EACH” Condition action

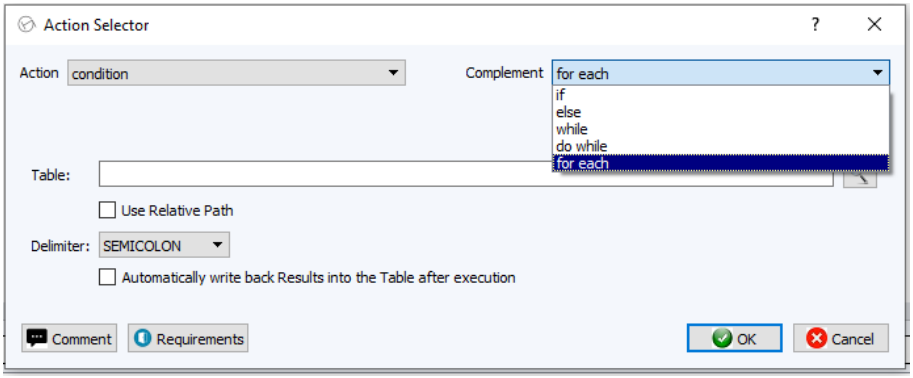


Figure: CONDITION For Each HMI

2) Select a file and the proper delimiter to the file.

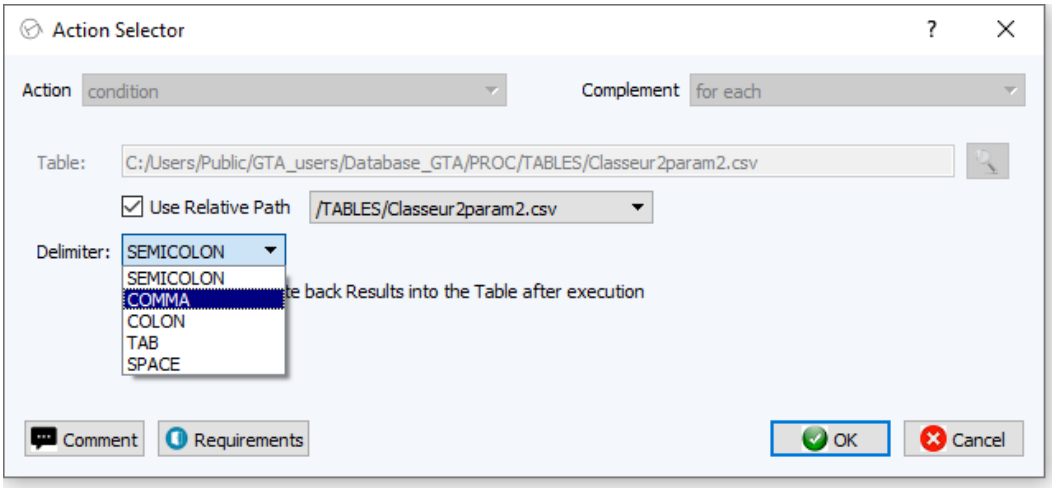


Figure: CONDITION For Each - csv file selection

3) If the user wishes to update some parameters automatically in the input csv table after the execution, the checkbox for “Automatically write back” should be ticked. The file update is performed when clicking on the Analyze button in the Result View.

4) Once the file is selected and validated in the widget, return to the Editor View and add an action within the “FOR EACH” scope. For example, we are going to perform a SET action.

5) In the Parameter/Value tag, open the Parameter Selection by clicking on the Search button.

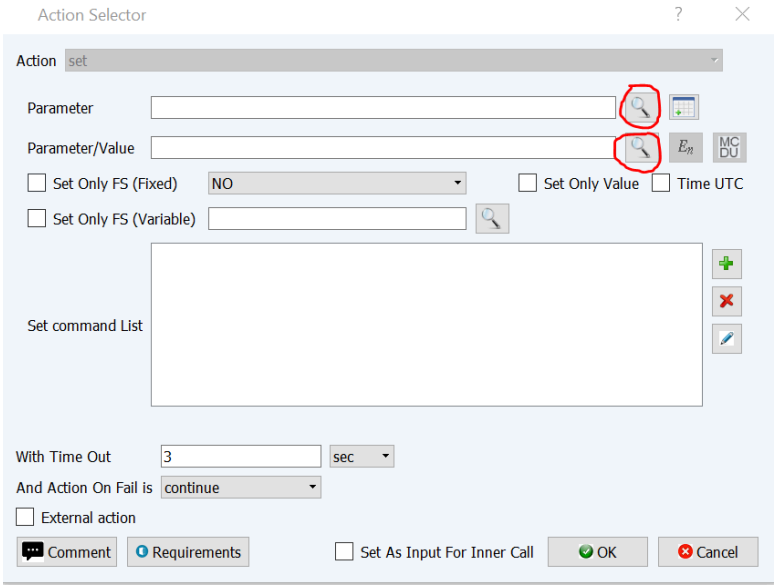


Figure: CONDITION For Each HMI

6) In Parameter Selection, enable Command Parameters and select the “Condition_for_each” option.

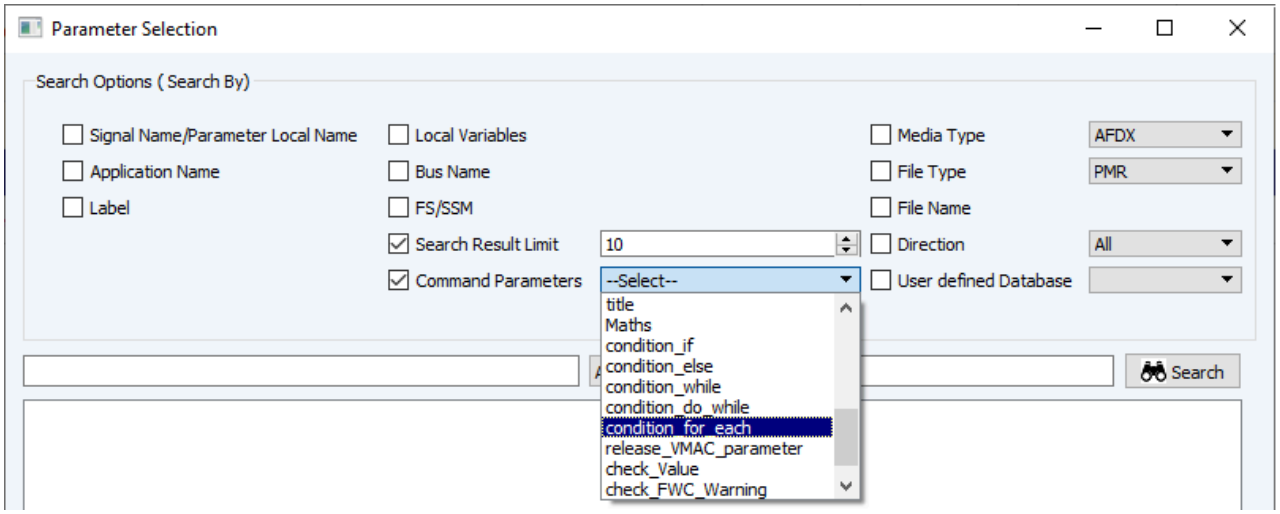


Figure: CONDITION For Each HMI - Parameter selection

7) Select the appropriate column you want to iterate through.

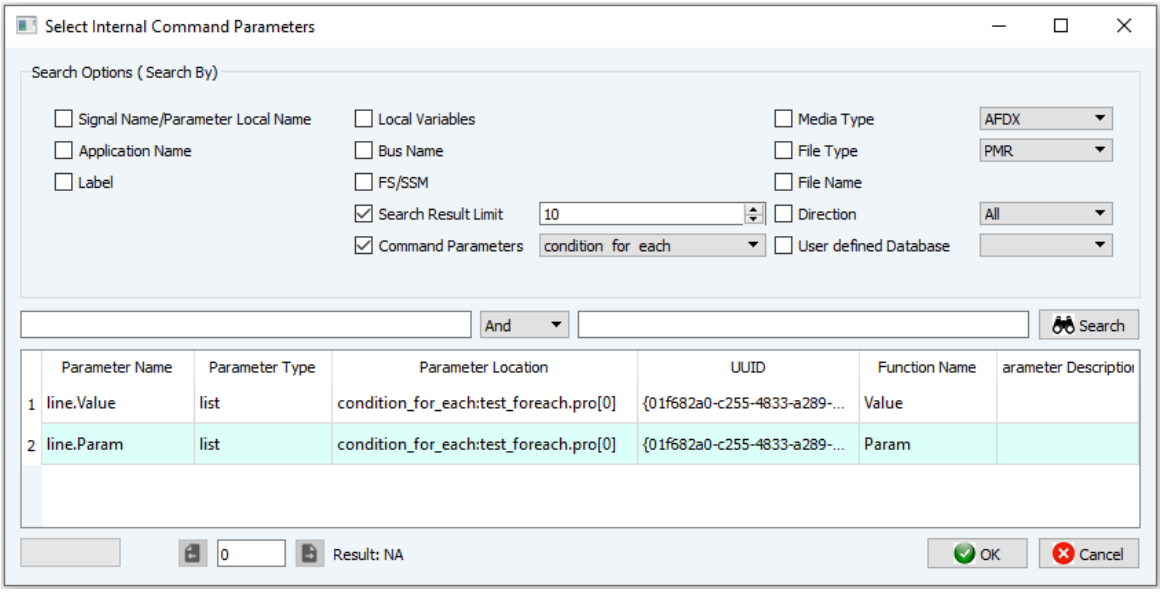


Figure: CONDITION For Each HMI - Parameter selection

8) Once selected, hit OK and the action is successfully added to the Editor View.
9)

3.8.4. "EXTERNAL TOOL"

The "EXTERNAL TOOL" command enables the user to execute a set of external functions that are defined in a specific configuration file. This XML configuration file contains all the functions that any tool needs to execute them.
This way a user can design a number of configuration files based on its requirements and these configuration files will be considered as different external tools.

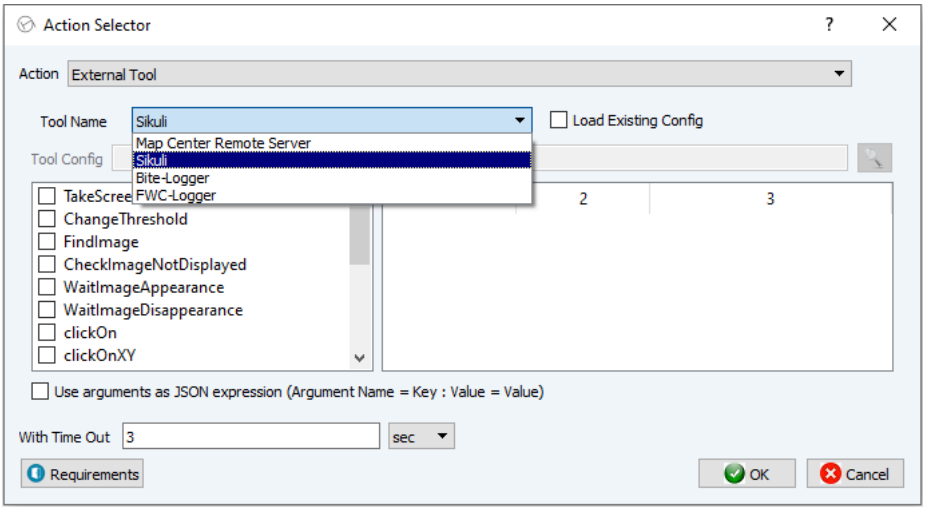


Figure: EXTERNAL TOOL HMI

The user can select the Tool from the Combo box Tool Name. This combo box will list all the tools that are present in the "ToolsConfig" folder. This folder has been created by GTA in the Bench Database path entered in the Path Settings tab.

Note: The user needs to manually copy the configuration files in the ToolsConfig folder.

3.8.4.1. XML Configuration files

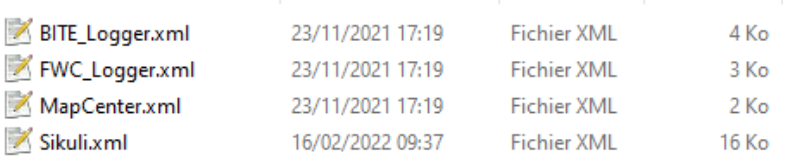


Figure: Configuration Files in the ToolsConfig

On selecting a tool from the combo box HMI displays all the functions of that tool in a list on the left hand side. Then, choosing a function from the list, a table will appear on the right hand side, with all the arguments of that particular function.
The table displays the argument name, type and has a column to enter values for the respective arguments.

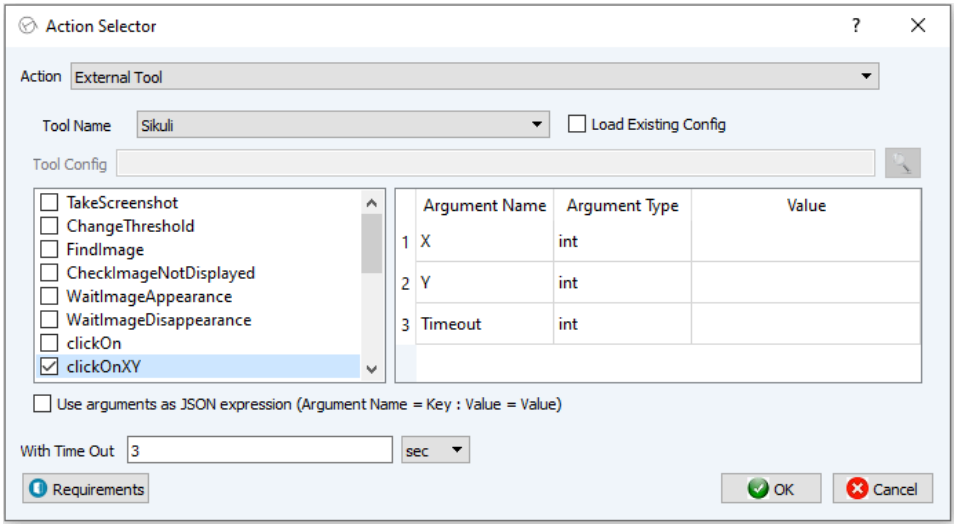


Figure: Arguments of the function

3.8.4.2. Search type option

GTA Editor enables users to search for the generic parameter present in the specific database. Using the search type option (Double click on value cell), GTA identifies and displays all generic and available parameters present in the database.

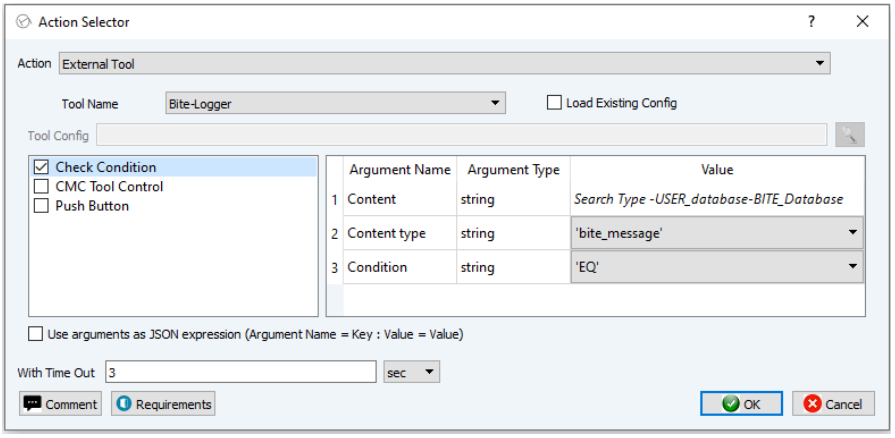


Figure: Search type option

Once double-clicking, we will have the possibility to make a search through the specific database.

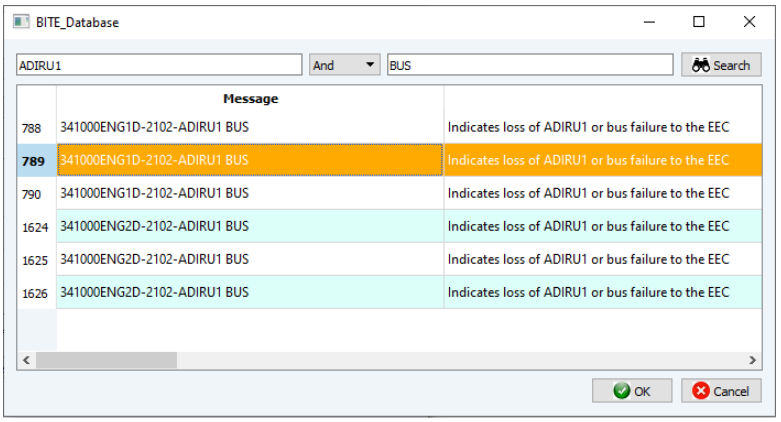


Figure: Search type option

3.8.4.3. External Tool expression

Once parameters are selected, the External Tool command is submitted. The external tool expression is visible in the Editor View.

		ACTION	TYPE
0			
1		Bite-Logger : Check Condition (Content : '341000ENG1D-2102-ADIRU1 BUS', Content type : 'bite_message', Condition : 'EQ')[ExternalToolID:{9386180d-3525-46d3-83ec-a7f4f4f098b3}]	Action
2			

Figure: External tool expression in the Editor View

3.8.4.4. Configuration file structure

Just for information, here is a part of the XML configuration file (Tool: SIKULI).

```
<GENERIC_TOOL xmlns="https://www.w3schools.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ToolsConfig.xsd" name="Sikuli" toolDisplayName="Sikuli">
    <HMI hasActionOnFail="0" hasTimeout="1" hasComment="0" hasDumpList="0"></HMI>
    <functions type="STATIC">
        <function toolId="Sikuli" argList="" functionDisplayName="TakeScreenshot" name="TakeScreenshot" toolType="Sikuli">
            <arguments>
                <argument HMILabel="Target" mandatory="yes" name="target" searchType="USER_database" userDbName="Sikuli_Database_target" defaultValue="" cond="" values="" type="string" constValue="" />
                <argument HMILabel="Name" mandatory="yes" name="name" defaultValue="" cond="" values="" type="string" constValue="" />
            </arguments>
            <returns>
                <return type="struct" name="Result" />
            </returns>
        </function>
        <function toolId="Sikuli" argList="" functionDisplayName="ChangeThreshold" name="changeThreshold" toolType="Sikuli">
            <arguments>
                <argument HMILabel="Threshold" mandatory="yes" name="threshold" defaultValue="0.95" cond="" values="" type="float" constValue="" />
            </arguments>
            <returns>
                <return type="struct" name="Result" />
            </returns>
        </function>
        <function toolId="Sikuli" argList="" functionDisplayName="FindImage" name="FindImage" toolType="Sikuli">
            <arguments>
                <argument HMILabel="Target" mandatory="yes" name="target" searchType="USER_database" userDbName="Sikuli_Database_target" defaultValue="" cond="" values="" type="string" constValue="" />
                <argument HMILabel="image" mandatory="yes" name="image_path" searchType="USER_database" userDbName="Sikuli_Database_image" defaultValue="" cond="" values="" type="string" constValue="" />
                <argument HMILabel="Timeout" mandatory="yes" name="timeout" searchType="" defaultValue="" cond="" values="" type="int" constValue="" />
            </arguments>
            <returns>
                <return type="struct" name="Result" />
            </returns>
        </function>
```

Figure: XML tool configuration file

3.8.4.5. Generic database

The Generic Database feature allows users to describe their own database in a configuration file so that that configuration file could be used as an External Tool command to pick up values from databases at runtime.

Following is the general format of the Generic DB.xml

```
<GENERIC_DB xmlns="https://www.w3schools.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <DB name="BITE_Database" relativePath = "\GenericDatabase\BITE_database.csv" delimiter=";">
        <attributes columnToReadFromSearchResult = "0" concatString = "">
            <attribute id = "0" name = "Message" column ="0"/>
            <attribute id = "1" name = "Description" column ="1"/>
            <attribute id = "2" name = "Type INT/EXT" column ="2"/>
            <attribute id = "3" name = "Conf Time" column ="3"/>
            <attribute id = "4" name = "FAULT COMBINATIONS" column ="4"/>
        </attributes>
    </DB>
</GENERIC_DB>
```

Figure: Generic DB xml file

The Generic DB.xml file will have a node DB name which will have the name of the database, path of the database.
The DB node will also have children nodes, namely attributes.
The attributes node will have the name of the attribute and the column number of that attribute in the *.CSV that needs to be read.

Similarly, several databases can be listed and used in Generic tool command via the Search type option. The search type in the Generic tool command configuration file will be required to be set as “USER_database”.

3.8.5. “PIR Parameters”

These commands enable users to subscribe and unsubscribe PIR parameters at any point in the procedure. It overcomes the limitation of being able to subscribe to only a limited number of PIR parameters on a bench.
Note: Each user must know the limitation of subscription.

The HMI of subscribe and unsubscribe is similar to that of print parameter value. Using the search option the user can select the parameter he/she wants to subscribe or unsubscribe. Following is the image for subscribe parameter, unsubscribe too has the same HMI.

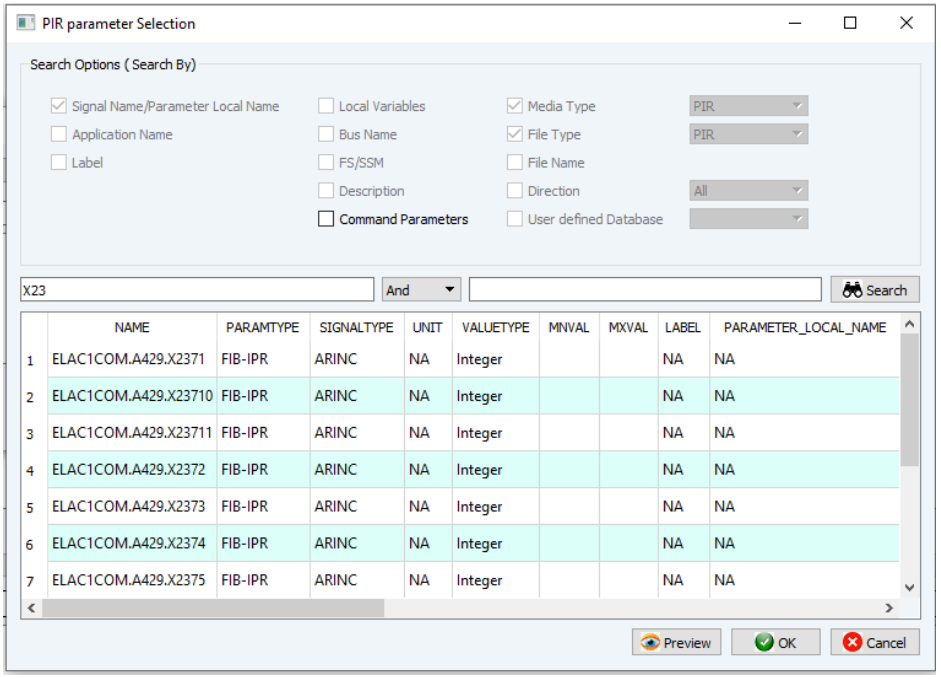


Figure: Search PIR parameters

3.8.5.1. Init PIR Parameters

Once a user has selected a PIR Parameter, the Init PIR Parameter HMI has changed.

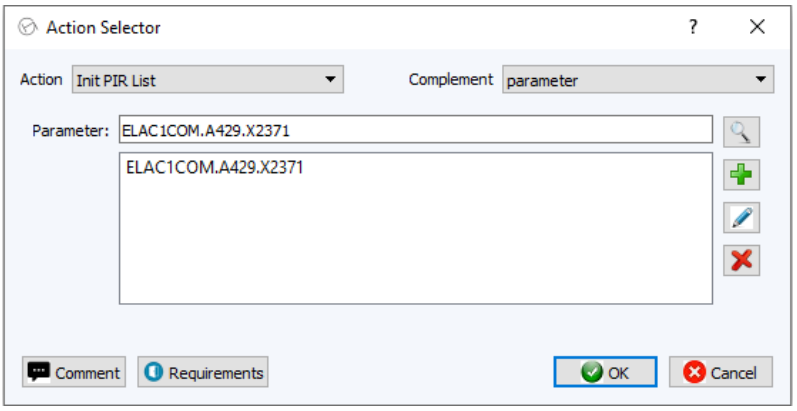


Figure: Init PIR parameters

3.8.5.2. Release PIR Parameters

The user can release all the PIR parameters in a procedure by ticking the checkbox “Release All”. All the PIR parameters initialized in the procedure are then released.

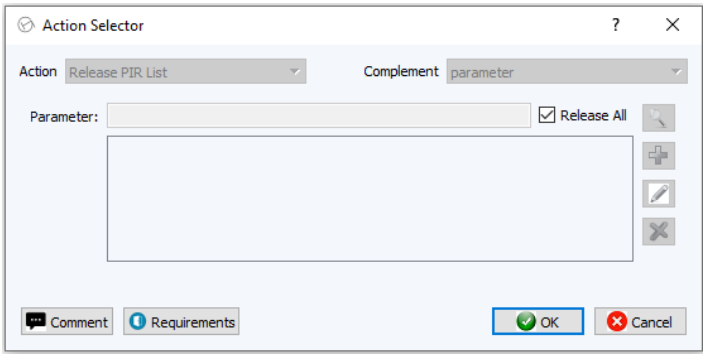


Figure: Release PIR parameters

3.8.6. “MANUAL ACTION”

A manual action describes a task that the tester must perform at any stage as defined in the test procedure, which allows the procedure execution to continue further.
In order to execute the test procedure in a ‘Semi-Automatic’ mode, GTA Editor provides an interactive controller module that handles any manual action that the user wants to perform during test execution.
The tester’s response to a manual action may or may not affect the next step under execution.

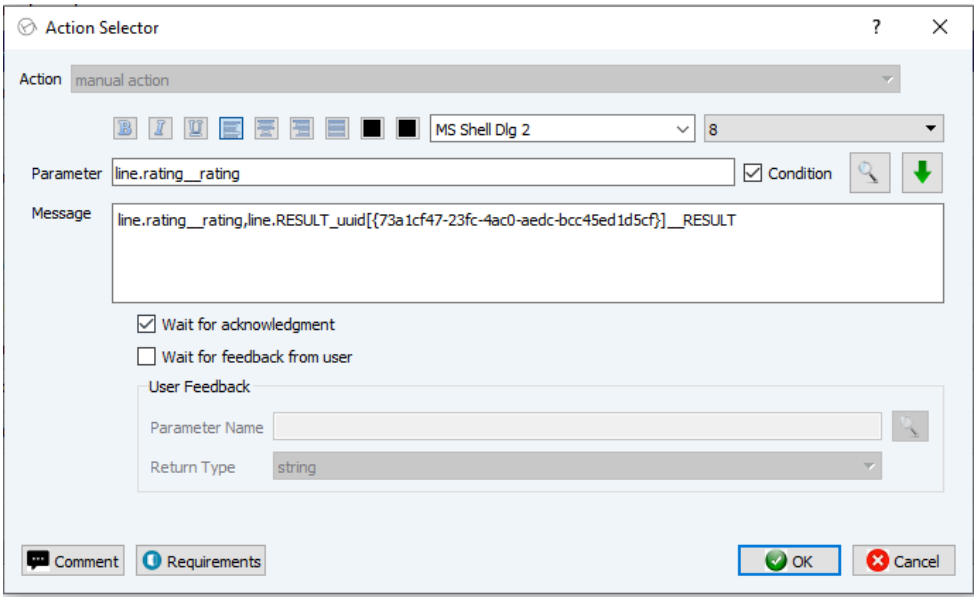


Figure: MANUAL ACTION HMI

Parameter: If the manual action row was added inside some condition which is contained in a ForEachLoop (i.e. involving a CSV file), then the user can insert the parameter of the current condition by clicking on the Checkbox. That will display the parameter of the condition in the line edit. To add into the message, the user has to click on the green down-arrow. The user can also search for other variables with the use of the search button.

Below is an example of the Manual Action popup displaying the value of some parameters. Notice that both the value and its parameter name are shown in the widget and separated by a “=”.

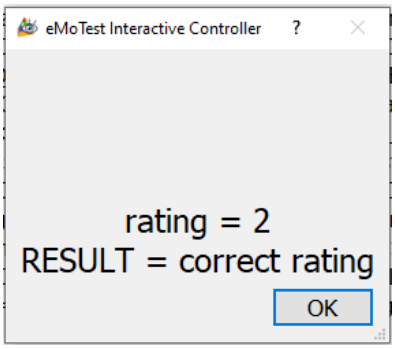


Figure: Manual action pop up

Message: The user can define the action that is expected to be performed manually by the tester at the current test step to continue with procedure execution.

Note: the + character is not allowed in a Manual Action message command

Wait for acknowledgement: If chosen, then during execution the test step shall continue to wait for user response to the requested manual action.

Wait for feedback from user: GTA Editor allows users to add a “User Feedback” to the manual action command.

- ✔ **Parameter Name**: Local variable or Bench Parameter in which the response shall be stored internally by GTA.
  Note: User feedback value for the parameter can be used later in the procedure.
- ✔ **Return Type**: Expected return type for the User Feedback (Boolean, Float, Integer or String)

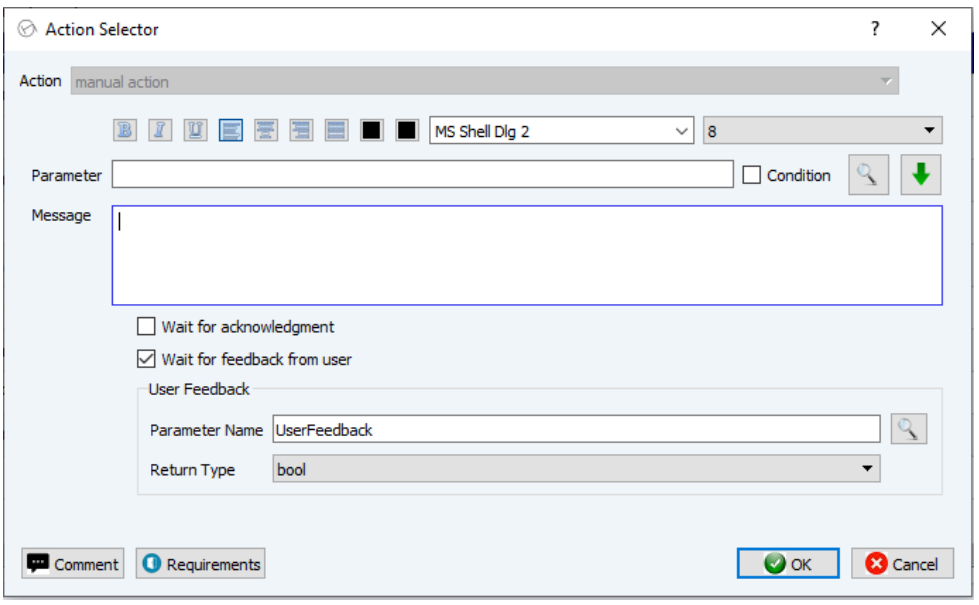


Figure: MANUAL ACTION – User Feedback

On pressing the OK button, the Manual Action test step can be seen in the Editor View as shown in figure below.

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
ManAct - Remove the jumper AJ1 and Check that Master Caution Light is not lit.	Action			--	

Figure: MANUAL ACTION – Test Step

Note: During execution of the 'Manual Action' Test Step, the Interactive controller (IC) module shall allow users to provide acknowledgement or Feedback by means of appropriate HMI's based on requests in the manual action command.

3.8.7. "MATHS"

This command is used to set a mathematical expression to a signal/variable. Mathematical expression generated has to be compatible with ECMA.

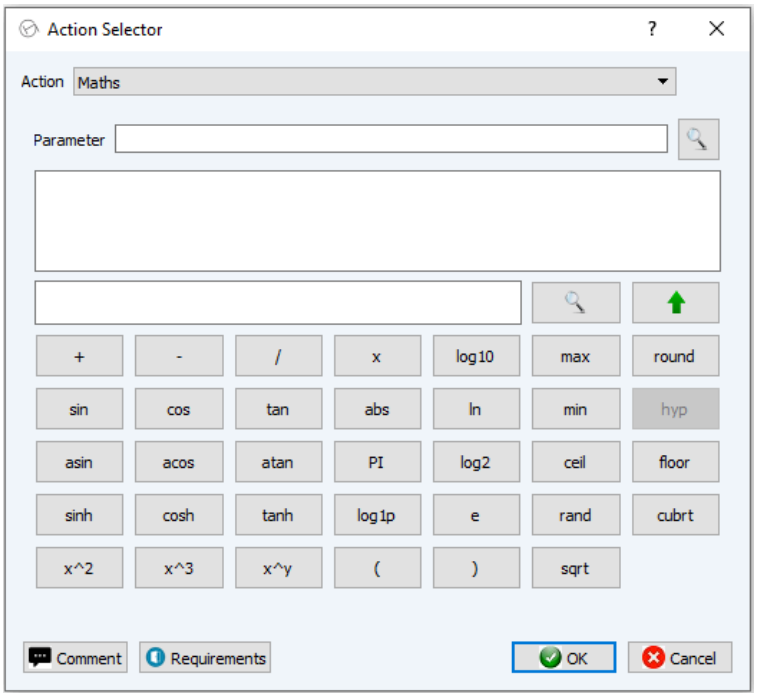


Figure: Maths HMI

For a valid mathematical command both parameter line edit and expression should not be empty.

Parameter text can be a local variable or DB parameter, use is similar to that of set command.
Expression building is similar to that of a scientific calculator. Line edit provided above operator buttons are used to enter constants, variables or DB parameters (using the search button provided next to it).

Mathematical symbols are not allowed in expression line edit.

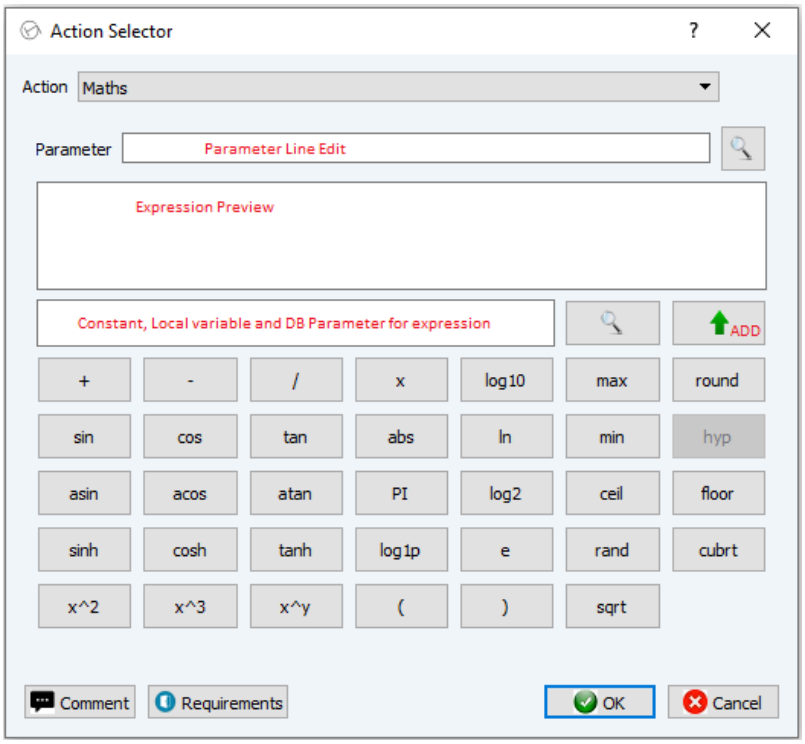


Figure: MATHS – Labels

Functionalities of Maths Command are briefly labeled in the figure above.

3.8.7.1. Creating Equation

Procedure to create an equation:

Example: $\sqrt{e^{var1} - 2 * var2 * A400MFCDC1.AFDX.BABERROR1}$

$\sqrt{e^{var1} - 2 * var2 * A400MFCDC1.AFDX.BABERROR1}$

Note: 3rd variable in above equation is an ICD parameter (A400MFCDC1.AFDX.BABERROR1)

Enter any constant, local variable or DB parameter in expression line edit.

1) Click on **sqrt** button followed by open braces button and **e** (exponential) function button.

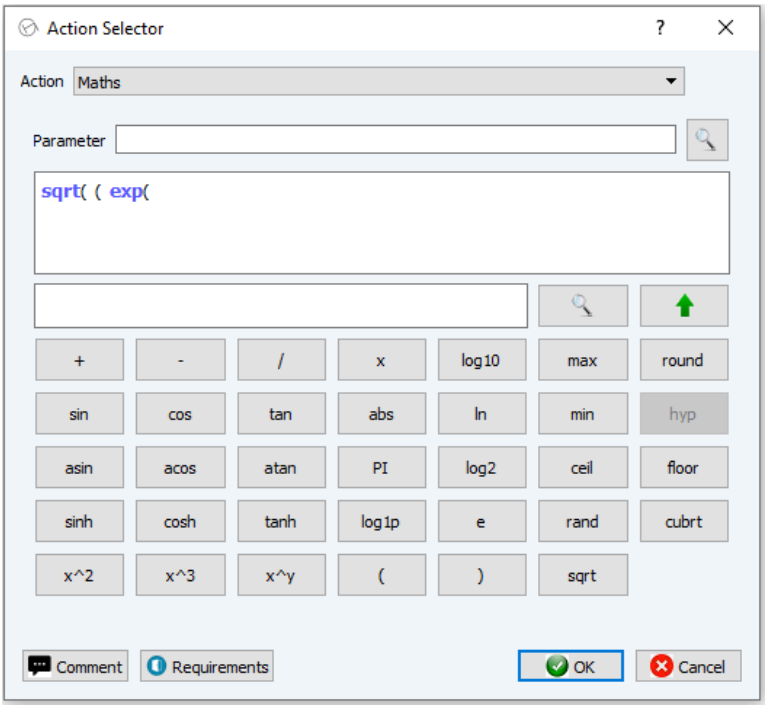


Figure: 'Building expression step 1'

2) Enter "var1" in the expression *line edit* and click on *add to expression* button. Close braces using the close bracket button.

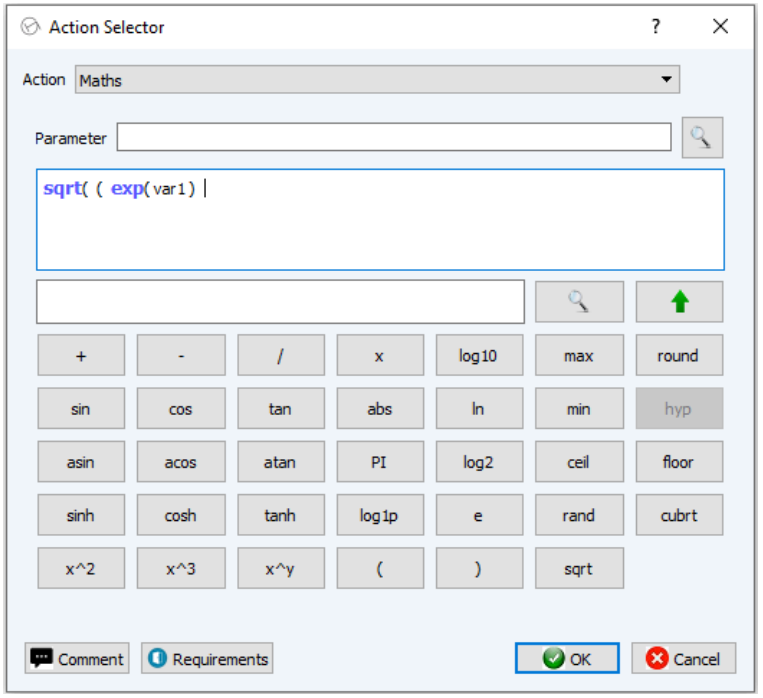


Figure: 'Building expression step 2'

3) In similar fashion insert operator – followed.
4) Constants are entered in "*expression line edit*" and pushed into expression using "add to expression" button.

5) Use operator and expression line edit and add to expression to the equation as shown in next figures.

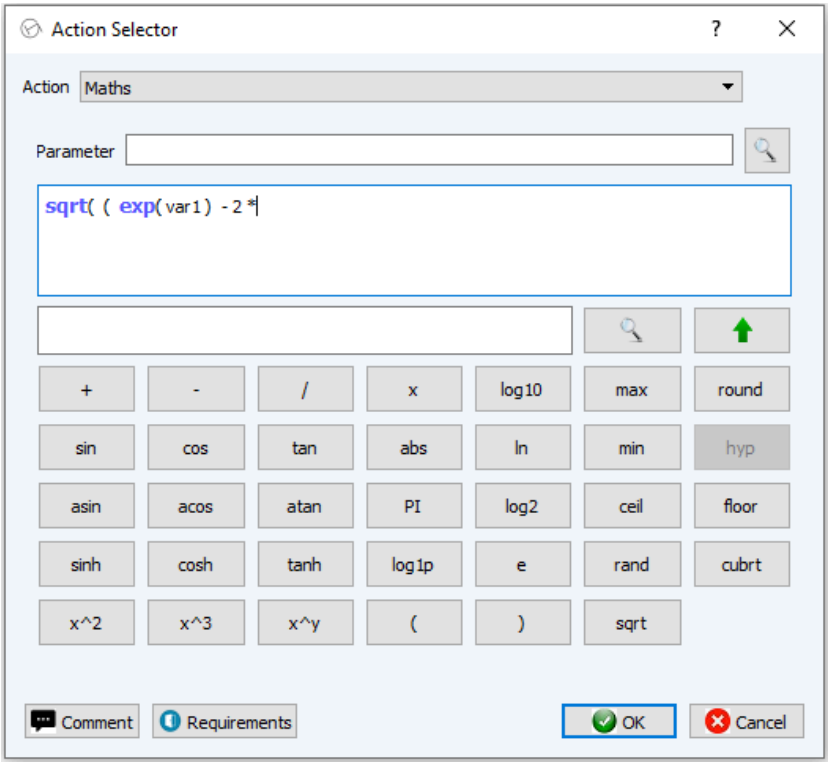


Figure: 'Building expression step 4'

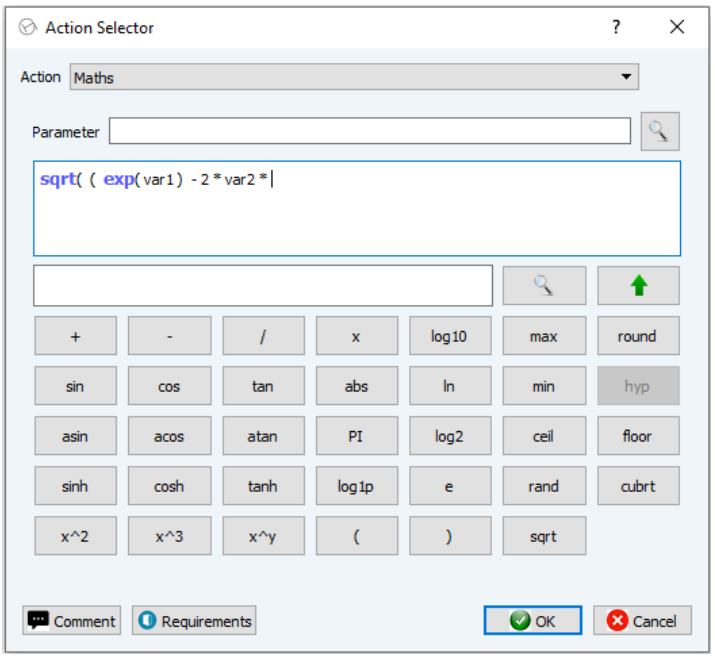


Figure: 'Building expression step 5'

6) Next variable to be entered is ICD parameter, use search button adjacent to "*expression line editor*" which will pop the parameter search window, operation is similar to that of searching parameter for set/check command.
7) On double clicking the parameter it would appear in Line edit add this to expression followed by close braces button.

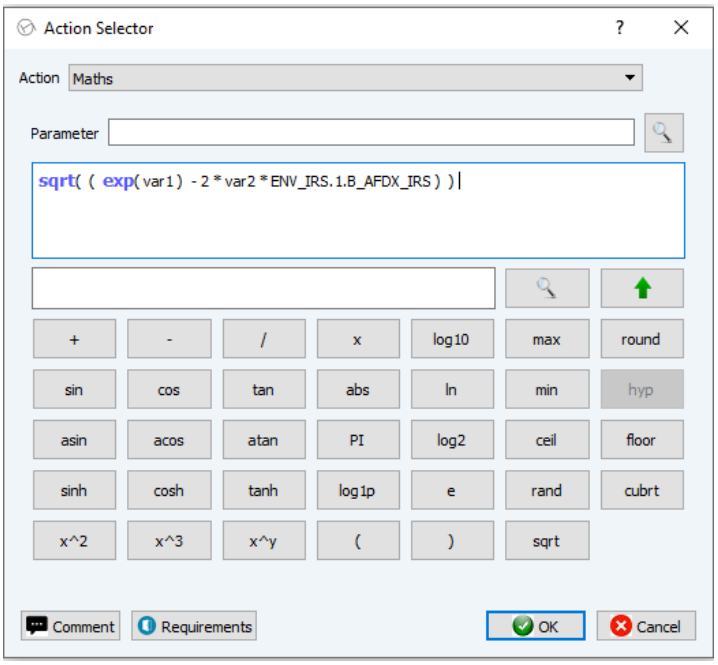


Figure: 'Building expression – Final step'

The expression is now complete.

3.8.7.2. Warning/error messages

Invalid input while forming expressions will be displayed. For instance expression like $cos(0))cos(0))$ is invalid.

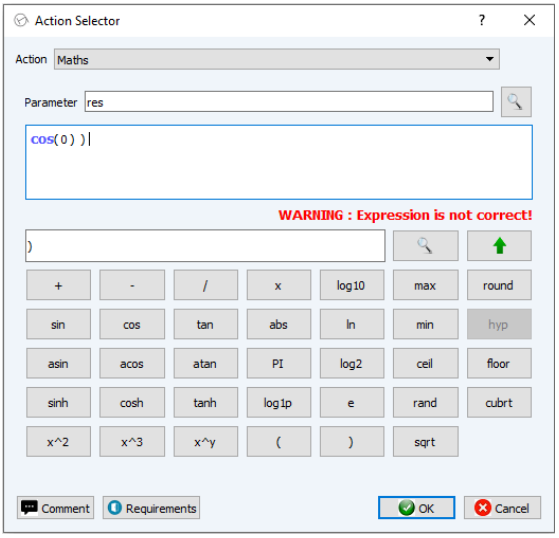


Figure: Expression is not correct

3.8.8. "PRINT"

"PRINT" action allows the user to print details on the execution log file for later reference.

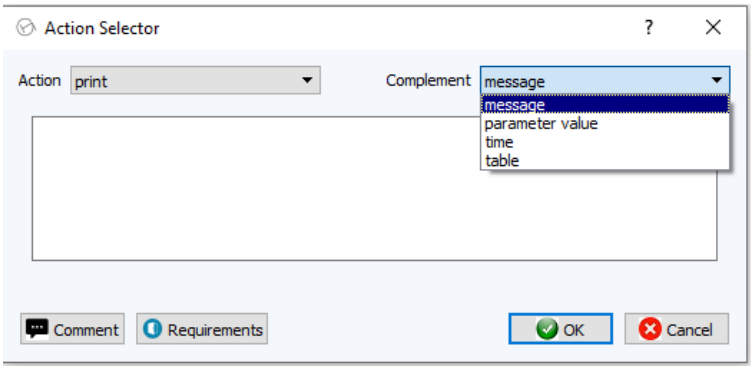


Figure: Print Value HMI

3.8.8.1. PRINT Message

The Print Message allows user defined messages to be printed on execution log file.

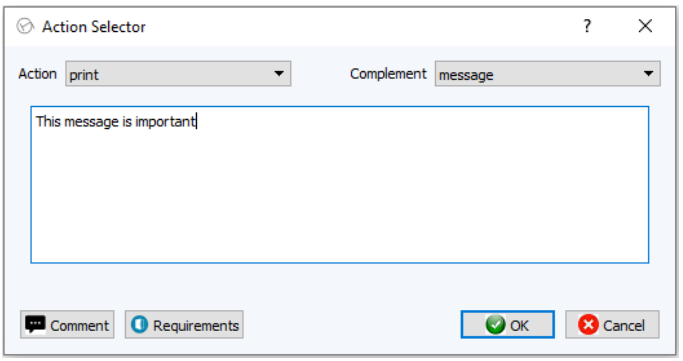


Figure: Print message HMI

Note: the following characters are not allowed in a Print message command
& é è à ù µ ç " ~ {} ' () [] | `/ \ ^ £ ¤ * % § ! ; ° , +

3.8.8.2. PRINT Parameter Value

The Print Parameter Value allows the value of a Bench Parameter or a Local variable to be printed on an execution log file.

The following steps show the different options available in Print “Parameter Value” HMI.

1) Search for Local variable or Bench Parameter using  button. Otherwise the user may just edit the “Parameter” text box directly.

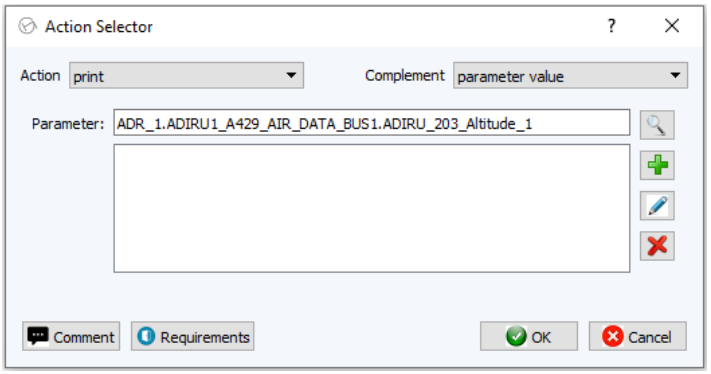


Figure: Print ‘parameter value’ HMI

2) Use  button to add the selected parameter to the list to be printed (This feature allows value of multiple parameters to be printed to the log using same HMI).

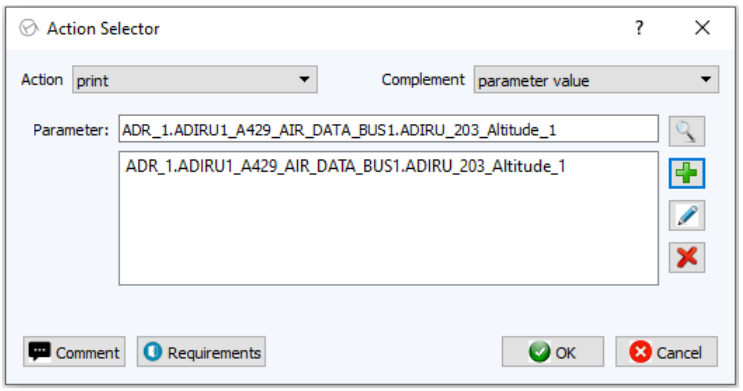


Figure: Multi Print HMI

3) Use  button to edit a selected parameter in the list
   - ✔ First select a parameter from the list to be edited
   - ✔ On selection the parameter will be reflected on the ‘parameter’ text box.
   - ✔ Edit the parameter as desired and
   - ✔ Click on  to ensure that the change is reflected into the print list.

4) Use  to delete a selected parameter from the print list

3.8.8.3. PRINT Time

The Print Time allows to log the current time (Local/UTC format) to be printed on the log file.

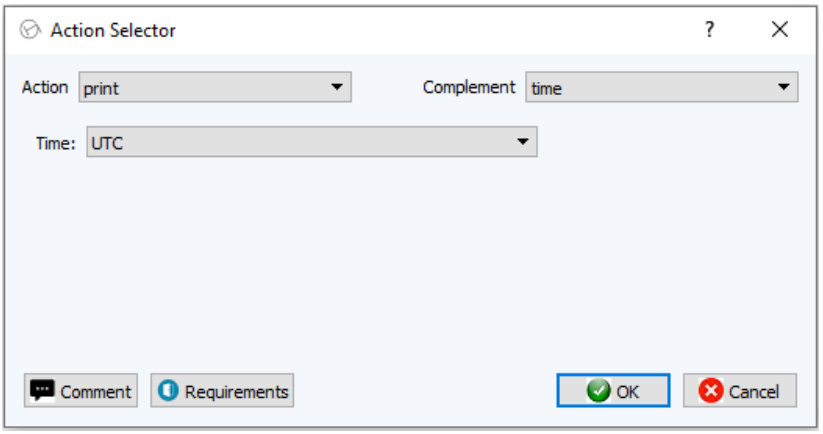


Figure: PRINT Time HMI

Time has two selections (now/NA). Now uses UTC format

3.8.8.4. PRINT Table

The Print Table command is used to print multiple parameters, which can be visualized in a CSV table format after execution during report generation.

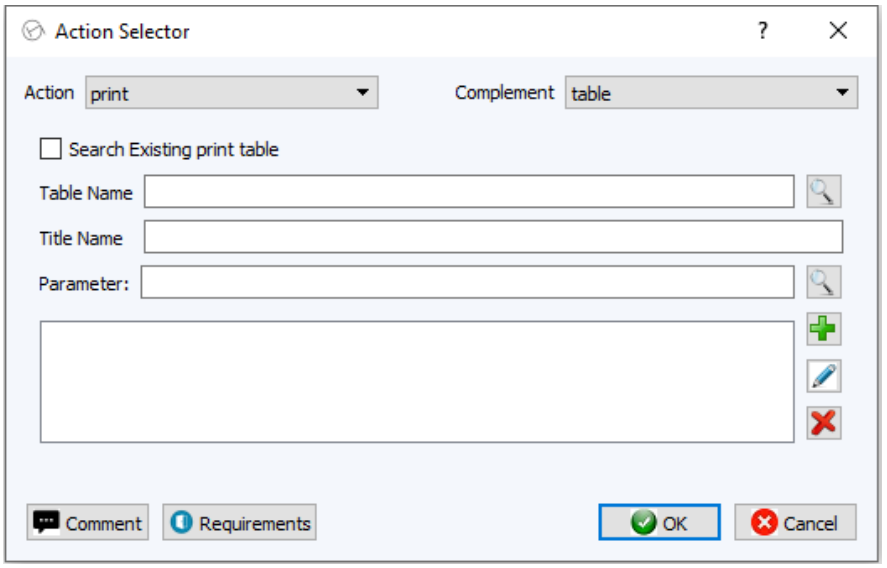


Figure: 'Print Table' HMI

This command is almost the same as the print command. The difference being users can store a set of parameters that needs to be printed as a table and reuse the same table at various points.

- Table Name:

Table name is mandatory. In case the provided name exists in current procedure, the user will be warned and will be overwritten if he chooses to do so.

- Title Name:

This is a mandatory field. The title will be used to generate the ***CSV*** file on LTRA export.

- On selecting “Search existing print table” would pop a search dialog containing an already existing print table for current procedure. Double click on table names to select the existing table. Clicking on add button () it adds all the parameters from the existing print table to the list in case the table does not exist nothing will be added.

-  Edit button:

Selection mode will change to “Search existing parameter” on selecting any parameter from the list. After search/keying in the required parameter, press on the edit button to edit the selected parameter.

-  Delete button:

Select the parameter in list to be deleted, and press delete button the item will be removed from list.

Editing existing table if is altered by this dialog then all existing print table commands in procedure with the same table name, will also be altered.

The below figure displays the test step on GTA editor for ‘print value’ actions.

ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
print message to execution log: This is a test message	Action		--	--	
print parameter value: MULTICAST.VL_ACSVCS_OPS_P9_1_GENER...	Action		--	--	
print time: Current Time in UTC	Action		--	--	
Print Table[TestTable] : parameters:	Action		--	--	

Figure: ‘Print’ test steps

3.8.9. “SET”

The “SET” command can be invoked in any of the following ways:

- ✔ Double Click on the ‘ACTION’ Cell on a row
- ✔ Right Click on the Row > Editor > Action > Set
- ✔ Toolbar Menu “Edit” option > Action

3.8.9.1. SET Value

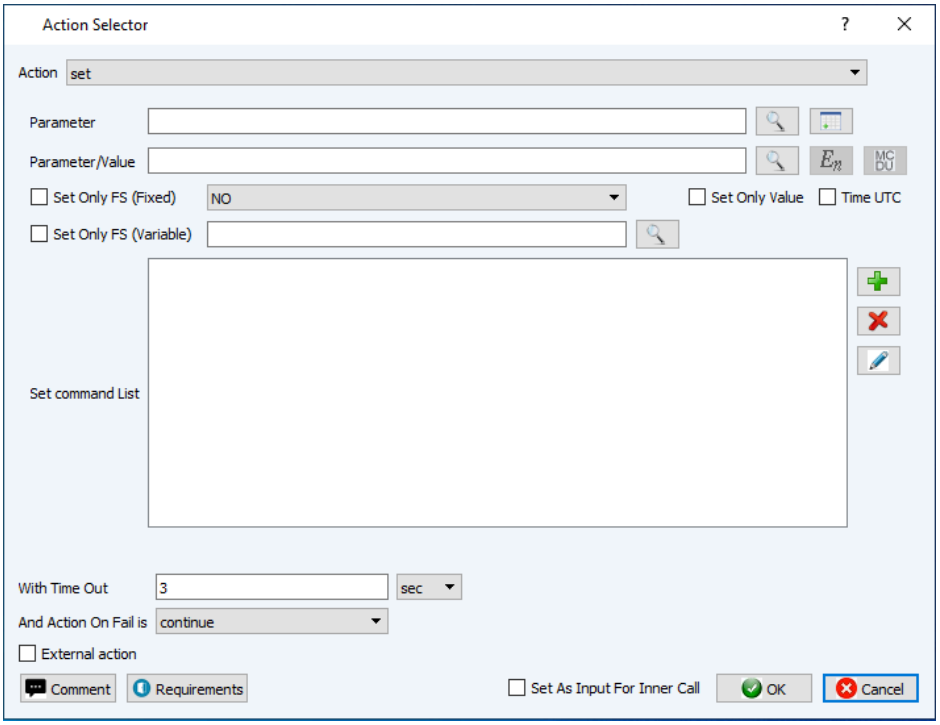


Figure: SET ‘Value’ HMI

The SET Value allows user to perform the following cases:

- ✔ SET a Bench Parameter/Local variable to a ‘Constant value’ or ‘an equation’ or ‘Value acquired by another Bench Parameter/Local variable’ or ‘MCDU request function”
- ✔ Only SET the Functional Status of a Bench Parameter.
- ✔ Add multiple SET in one control selector HMI.
- ✔ SET TimeUTC
- ✔ Set As Input For Inner Call: the value of a parameter instead of its name is applied within a called function if this checkbox is ticked.

For an edition, the last updated item in the command list is selected automatically.
If an item is deleted, the automatic selection is done on the item of the last row.

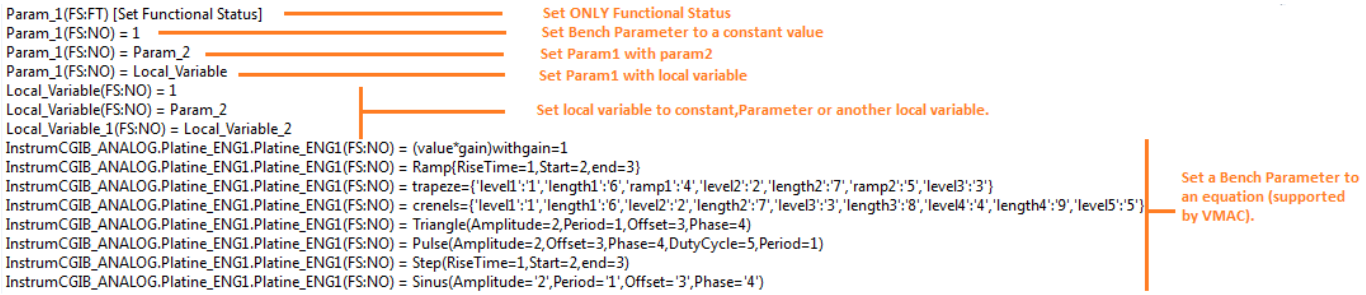


Figure: SET 'Value' possibilities Sample

3.8.9.2. SET Functional Status

The SET action selector allows user to set the functional status of a bench parameter by selecting the functional status from the following list:

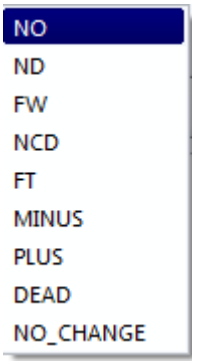


Figure: SET 'Value' Functional Status list

User has several options while performing a SET Value:

Functional Status (FS) of a parameter can be SET along with value by choosing the expected FS from the list and also providing data for 'Parameter/Value'.
Opting to SET only the Functional Status of a parameter and not its value by selecting the feature and choosing the expected FS from the list. Set Only FS

3.8.9.3. SET 'Value' HMI Details

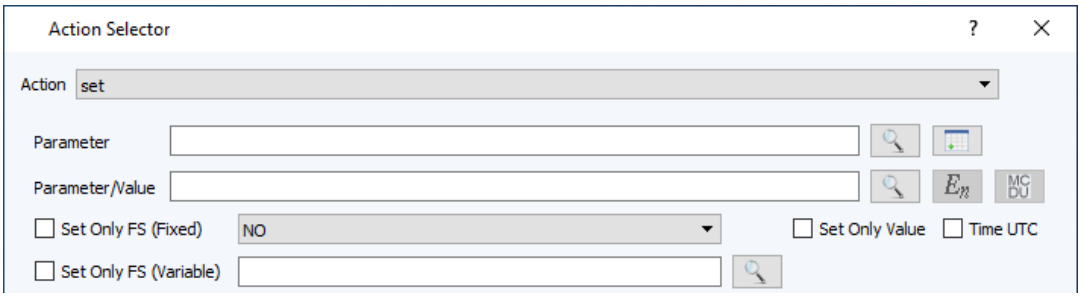


Figure: SET 'Value' HMI – User Inputs

We are going to explain all the different and available methods to SET a value.

3.8.9.3.1. SET command with Local variable

This command is the easiest way. To do this, it is necessary to enter a Parameter name in the specific field and associate it with a value in the Parameter/value field.

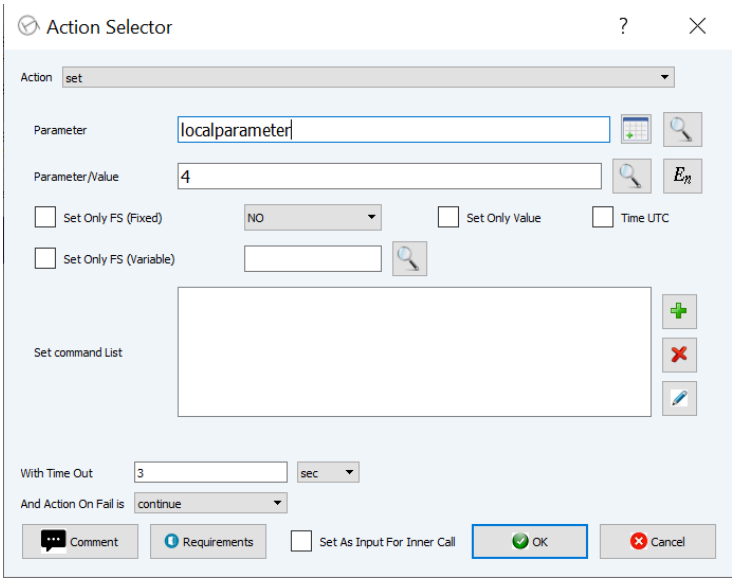


Figure: SET 'Value' HMI – Local variable

Note: A local variable has no Functional Status, so its FS is NO_CHANGE and SetOnlyValue option is checked.

3.8.9.3.2. SET command with AC_Parameter

This command allows the user to search AC_Parameter in the Database by clicking on the loop icon of the Parameter field.

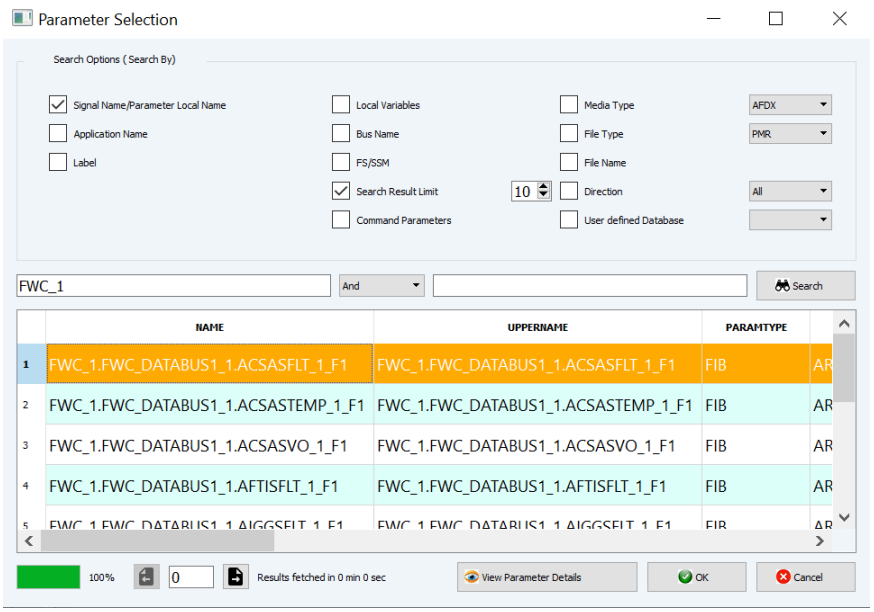
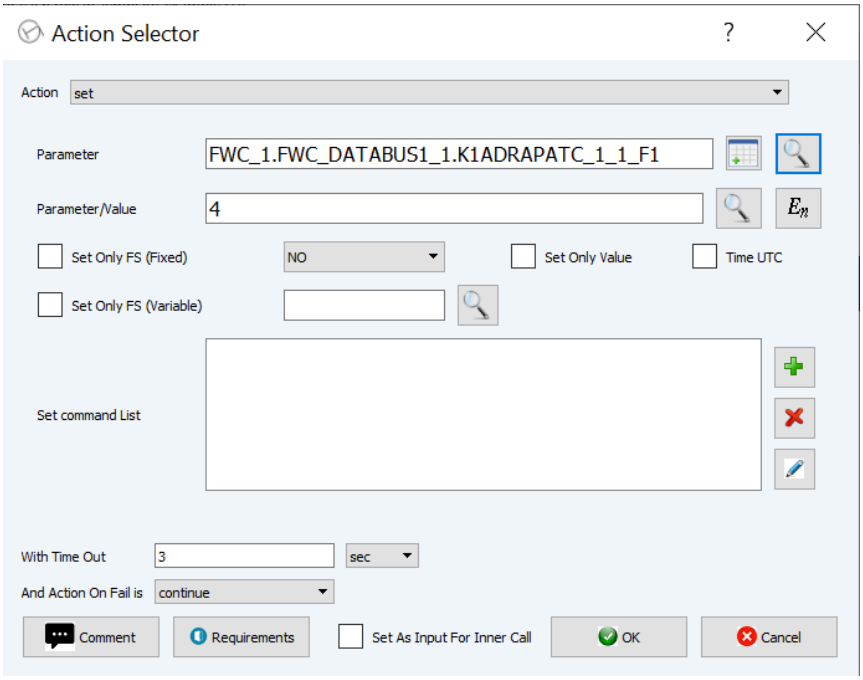


Figure: SET 'Value' HMI – AC parameters

Once the search is performed, the AC_Parameter selected and validated, the SET command will be displayed.



Note: This AC_Parameter is non-VMAC type, so the user will have the choice of its Functional status and the possibility to check SetOnlyValue option.

3.8.9.3.3. SET command with imported values

This command allows the user to set a list of parameters by using a CSV file as an input. The mandatory format for input CSV file is described below.

- **AC_Parameter (triplet) ; Value ; Functional Status**

Note: Users must maintain the above order of input fields.

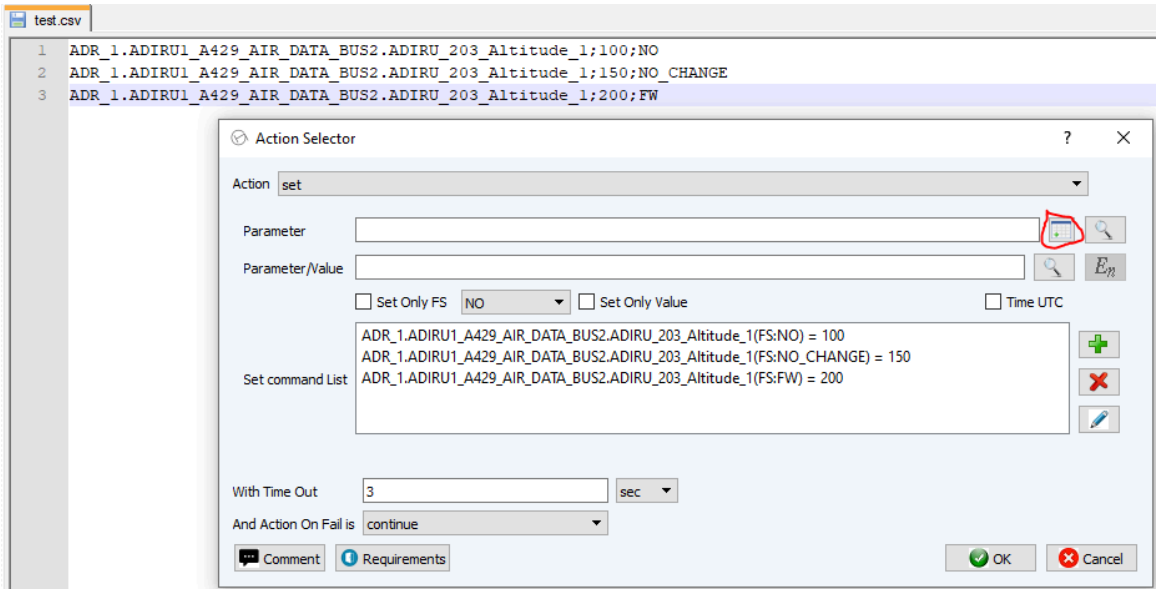


Figure: SET 'Value' HMI – Imported values

3.8.9.3.4. SET command with an Equation

This command allows the user to set any type of parameter associated with an equation. To do this, it is necessary to fill in the Parameter field and click on the EQ icon.

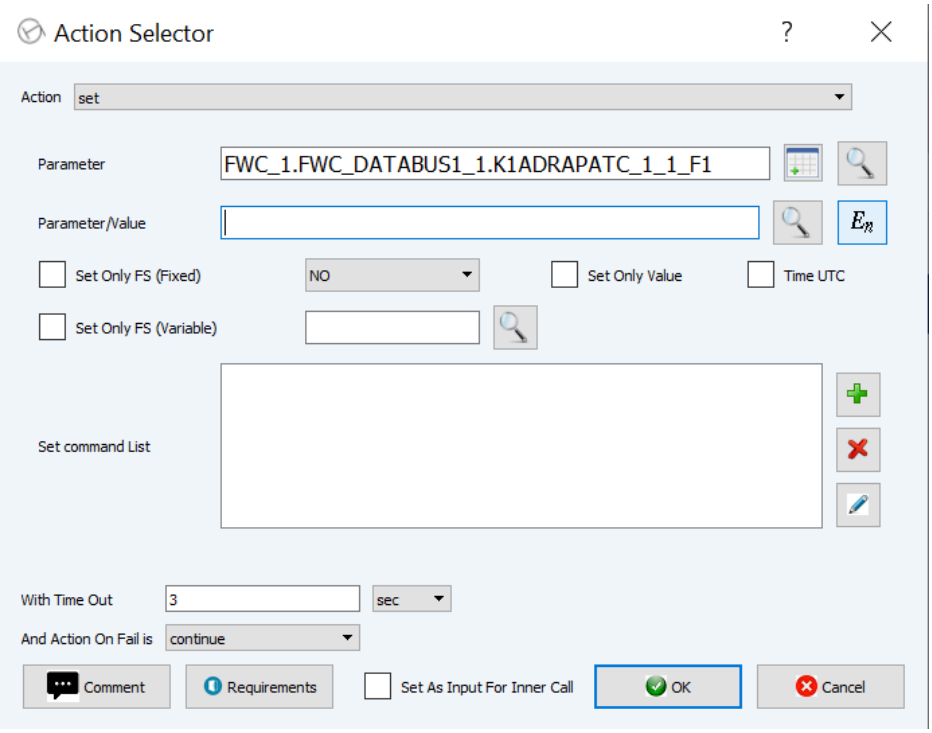


Figure: SET 'Value' HMI – Equation

According to the parameter type (VMAC or not) several equations could be used.

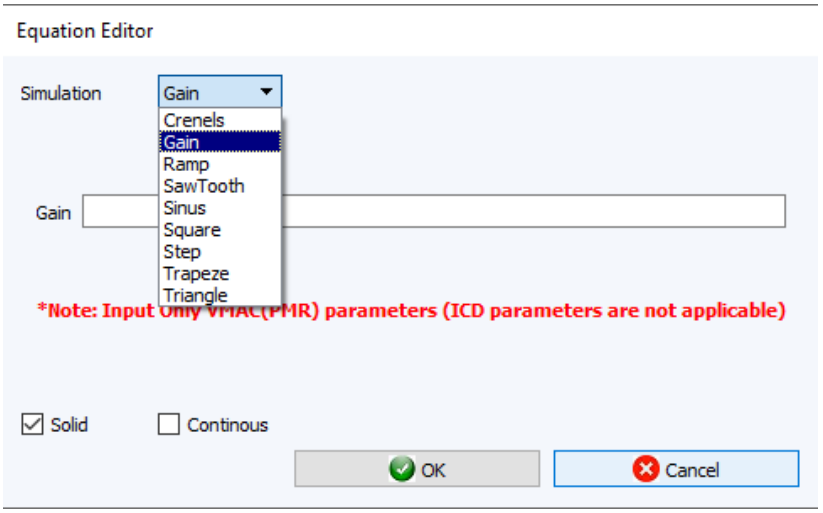


Figure: SET 'Value' HMI – Equation

For each kind of equation, Parameter/value fields can be fulfilled with values (Integer, Float) and/or local variables.

NB: Please make sure for Triangle Wave, Sinus and Sawtooth, that the 'Start value' is between 'Max value' and 'Min value'. A warning is added in the HMI for these equations:

Figure: SET 'Value' HMI – Warning message

3.8.9.3.5. SET command with TimeUTC

This command allows the user to set a parameter to Time UTC value. To do this, it is necessary to check the Time UTC option; the Parameter/value field will turn into the CurrentTimeUTC.

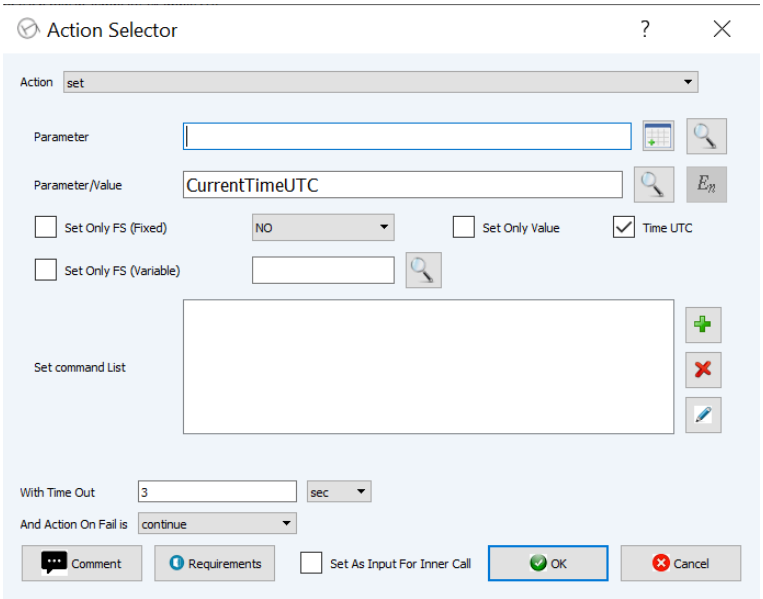


Figure: SET 'Value' HMI – Time UTC

This parameter/value will allow the user to get the current time UTC once the communication with eMoTest is performed.

3.8.9.3.6. SET command with MCDU request

This command allows the user to set a MCDU request with one of the available functions. These functions are defined in the **MCDU_GTA.xml** file which should be delivered within the GTA-Editor package. Similarly to ExternalTool, this file needs to be copied into the ToolsConfig folder to be visible by the application.

The MCDU button is disabled until the user enters a valid Parameter name in the Parameter field : it should contain the keyword **MCDU_REQUEST**. If the button becomes enabled, the user is no more allowed to update the Parameter/Value field manually. By clicking on this button, a MCDU function widget is displayed inviting the user to select a function and to set its arguments.

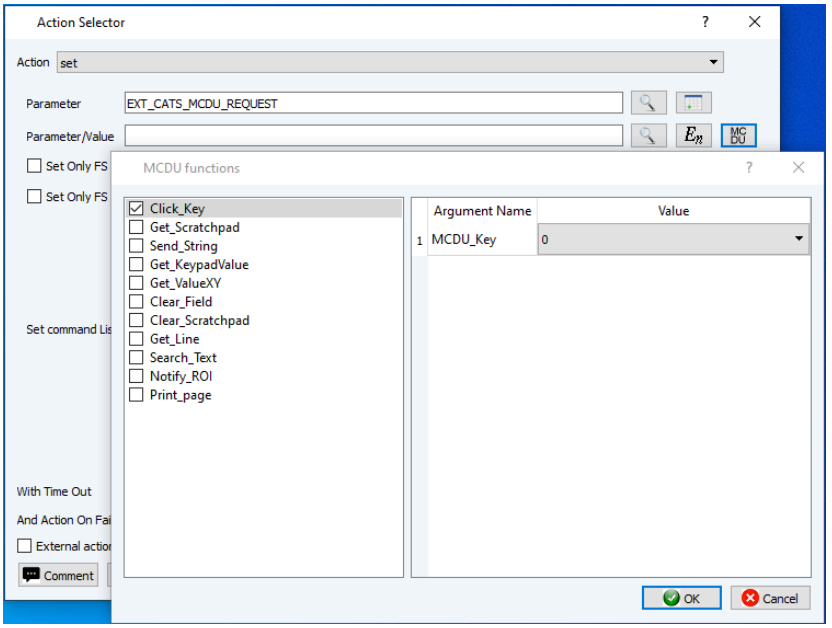


Figure: SET 'Value' HMI – MCDU commands

When inserting a new SET command, all the functions are visible whereas when editing, only the current function is accessible and the last user selection is automatically set.

Conversion to MCDU codes is performed from the widget to the Parameter/Value field and reverse conversion is done on edition.

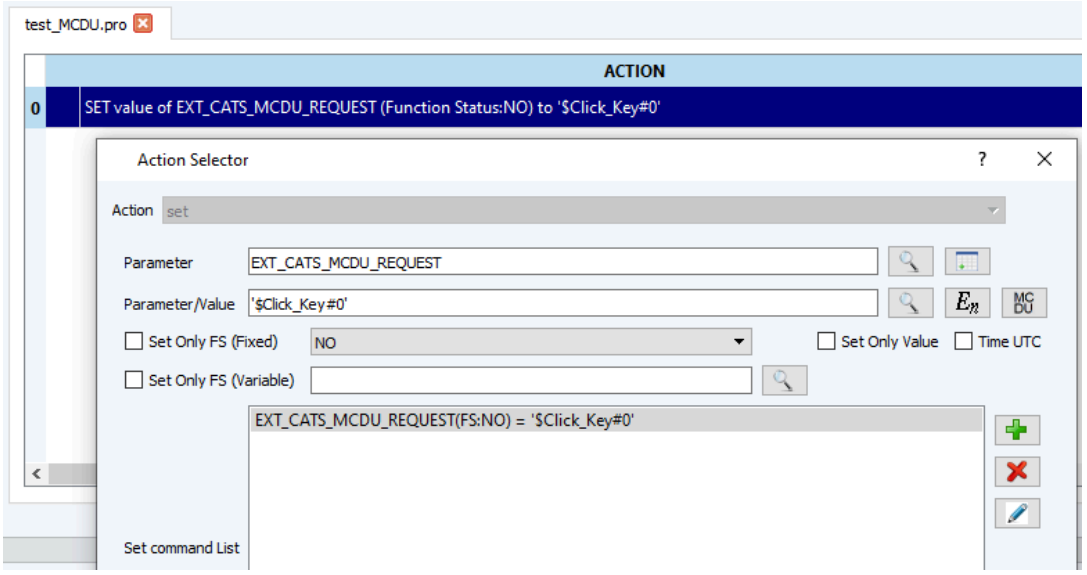


Figure: SET 'Value' HMI – MCDU commands

Notice that since the complete MCDU request format is :

Timestamp$MCDU_Function#Param_1,Param_2,…Param_I

the Timestamp is added by the MCDU bridge at the time it receives the variable before sending the full request to the MCDU Api.

3.8.9.3.7. External action

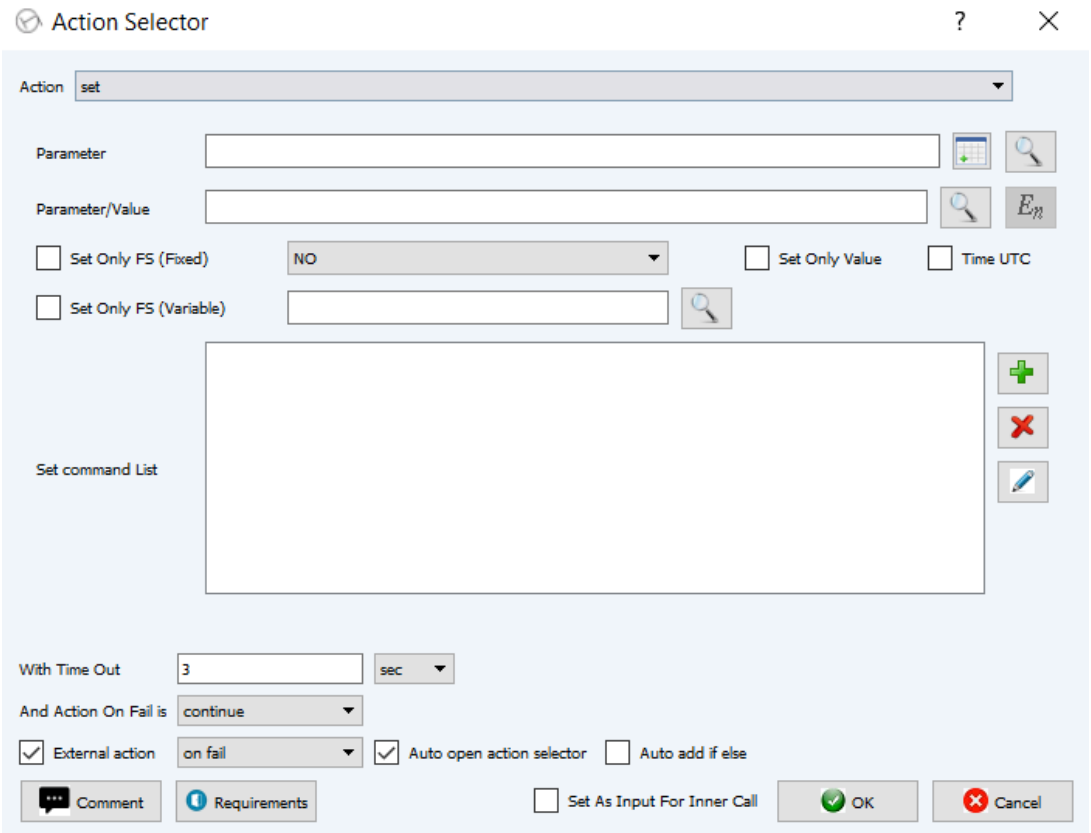


Figure: SET 'Value' HMI – External actions

Option for automatic external action on set success/failure is available.
This option will automatically insert an IF condition below created SET statement. The IF condition will check for return status of the created SET command.

Checking the checkbox will expand a new sub-menu of choices.
First combobox allows the user to specify if the action should be performed on set's success or failure.
Auto open action selector will automatically open an external tool window selection, upon clicking OK on this submenu, the selected action will be automatically inserted into the IF condition.
Auto add if else checkbox will automatically add an ELSE below IF condition.

3.8.10. "TITLE"

The "TITLE" command allows the user to provide a heading for a set of steps that helps other users to better understand the content of the procedure. This behavior is a little bit like a section.
Titles are often used to summarize the results in the LTRA for the tester to quickly have the results overview.

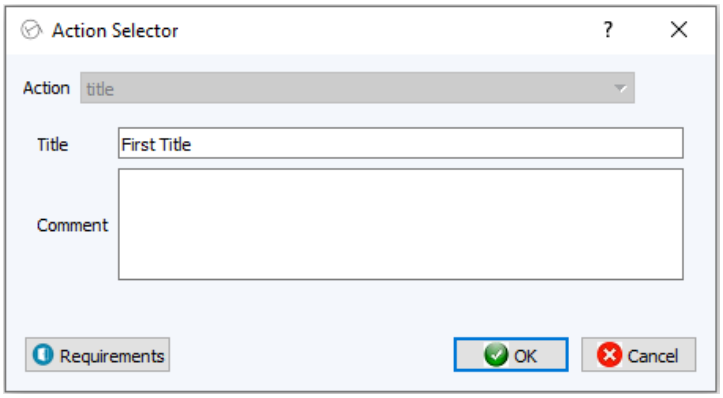


Figure: TITLE Action

Note: the following characters are not allowed in a Title command
& é è à ù µ ç " ~ {} ' () [] | ` / \ ^ £ ¤ * % § ! ; ° , +

Note: The feature is currently disabled under all conditions.

Title row on the editor acts as a separator to distinguish a set of steps from others and hence these rows are highlighted (blue) on the editor as shown in figure below.

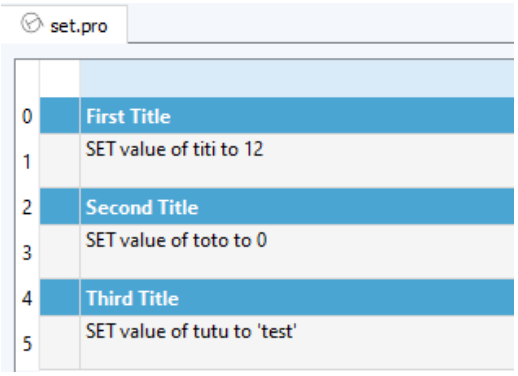


Figure: Title Action Steps on Editor

Collapse to Title – This feature allows user to view a summary of the complete procedure (or any other GTA element) by minimizing contents based on title rows.

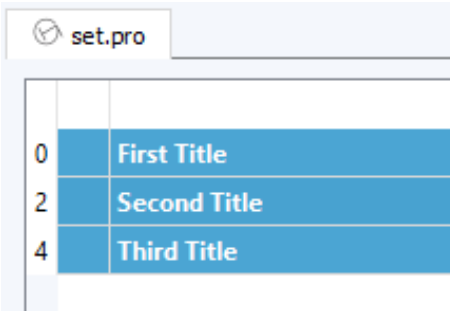


Figure: Collapse to title

User may select a specific title row in the above view and click again on button, this will expand the complete procedure and the scrollbar will move to the location of the procedure with the selected title row.

Note: This feature is useful in quickly switching to the desired section of the procedure in case of larger procedures. Also no modification can be performed to the procedure when in collapsed view.

3.8.11. ”WAIT”

The “WAIT” command has 2 options. Either “WAIT FOR” or “WAIT UNTIL”.

3.8.11.1. WAIT For

The “WAIT For” command allows the user to wait for a defined time.

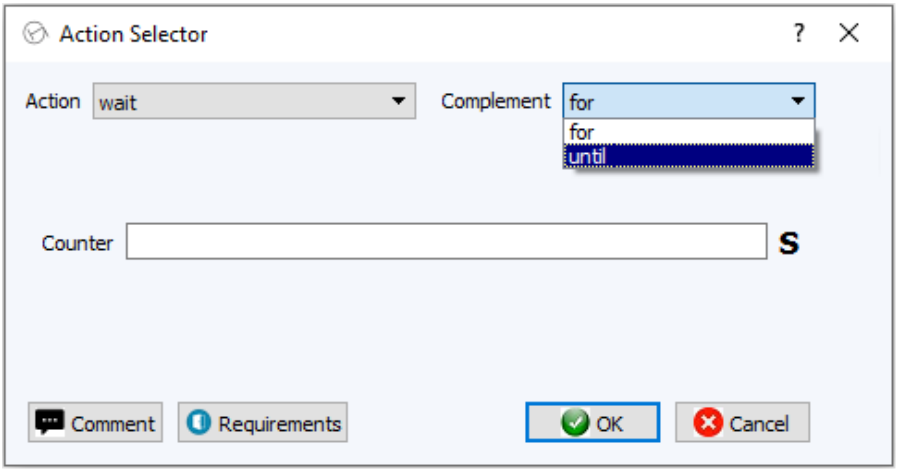


Figure: WAIT FOR action selector

“WAIT For” action selector allows the user to provide a time for the procedure execution to wait before execution of the next test step.

It is possible to enter an integer or float value, but also an argument or a local variable. To do so, users either must use @x@ where x represents the name of the argument or directly enter the name of the local variable.

Make sure the local variable is being already defined.

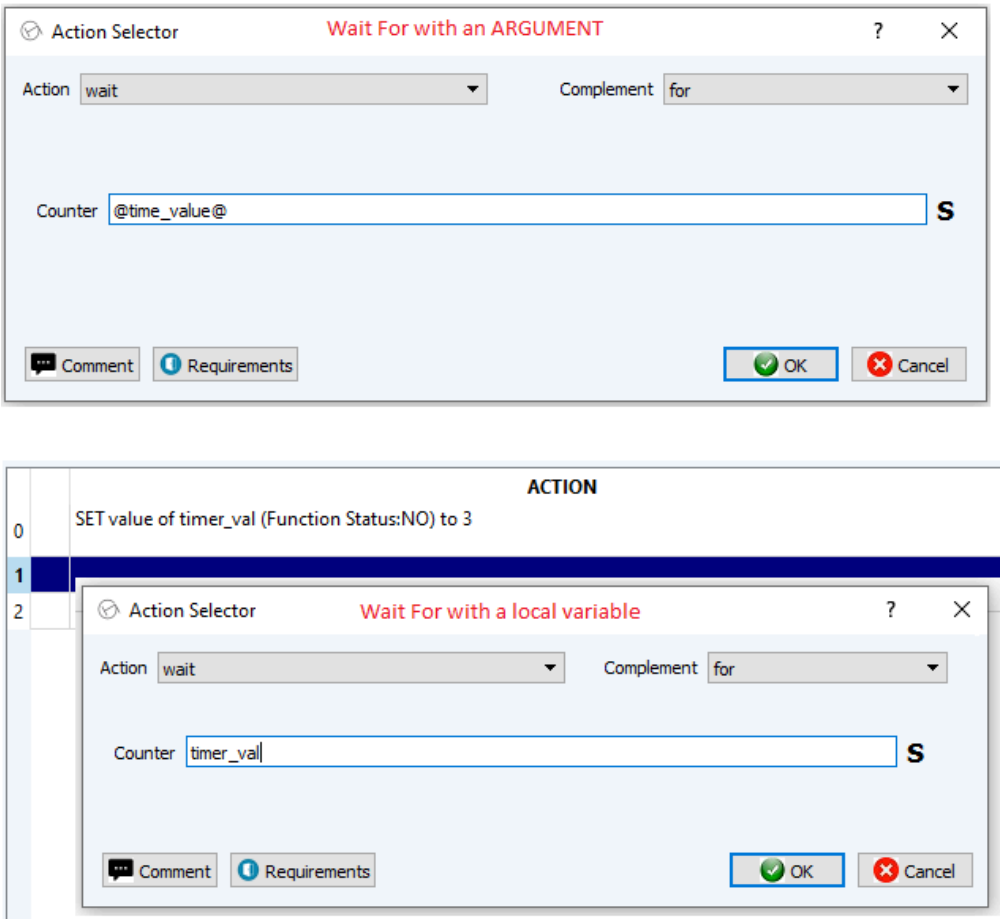


Figure: WAIT FOR action - Argument & Local variable

3.8.11.2. **WAIT Until**

The “WAIT Until” action allows the user to define a condition on which the procedure execution will wait before execution of the next test step.

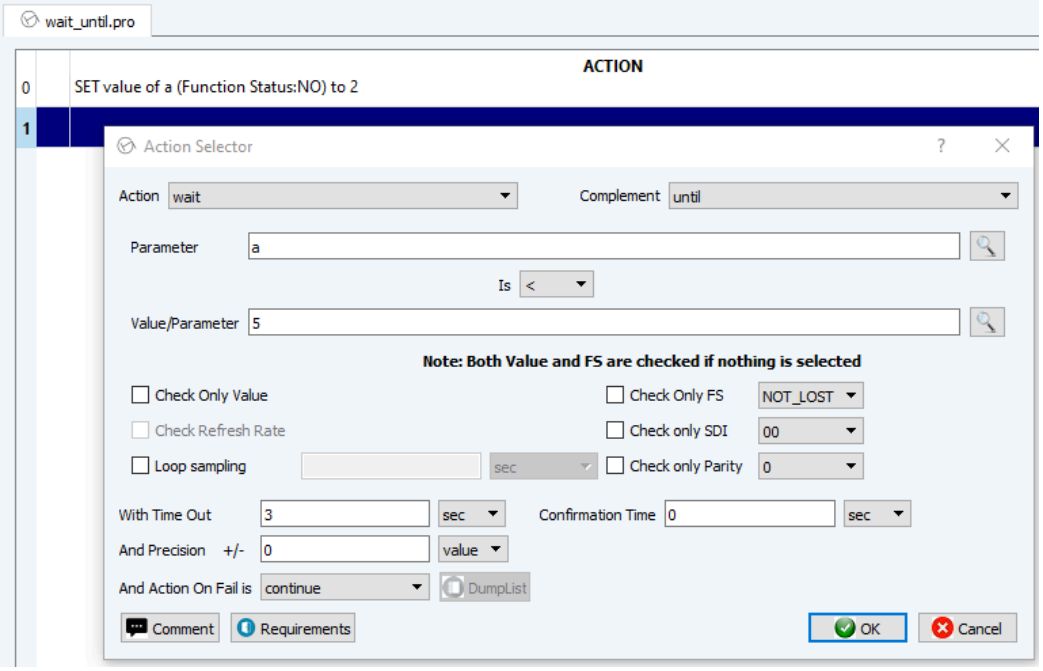


Figure: ‘wait until’ action selector

On the HMI, the user can define the timeout (in seconds or milliseconds) for the “WAIT Until” action and the precision (in % or value) with which the condition check has to be performed.

“Action on fail” definition allows the user to define the course of execution (abort or continue) based on the result of the condition check & timeout inputs.

The user can use the Loop Sampling option by ticking the checkbox, then enter a value and choose an unit (sec/ms). This option allows the user to iterate through the log messages until the condition or the timeout is reached. Once this option is enabled, this information is stored in the file.

```
</ACTION>
<ACTION PRECISION="0" NAME="wait" COMPLEMENT="until" CONFIRMATIONTIMEUNIT="sec" IGNORE_IN_SCXML="FALSE" ONFAIL="continue" (
 <PARAMETER CHK_PARITY_ONLY="FALSE" NAME="a" LOOP_SAMPLING_STATUS="TRUE" LOOP_SAMPLING_VALUE="4" LOOP_SAMPLING_UNIT="msec"
 <PARAMETER NAME="LOOP_COUNTER" VALUE=""/>
 <COMMENT>""</COMMENT>
 <REQUIREMENTS/>
</ACTION>
```

The figure below shows the editor steps for “WAIT For” and “WAIT Until” actions.

wait_cmd.pro

	ACTION	TYPE	COMMENT	TIME OUT	PRECISION	ACTION ON FAIL
0	wait for 3 S	Action		--	--	
1	SET value of param (Function Status:NO) to 2	Action		3 sec	--	continue
2	wait until param > 4	Action		3 sec	10 %	continue
3						

Figure: Editor steps for “WAIT” actions

3.9. Additional Features

3.9.1. Find & Replace

Find dialog enables user to perform the following searches:

- Text Search – Users may search for any text in the procedure. Searched rows are highlighted in green as shown in image below.
- Parameter search – User may restrict his search to specific parameters used for a command.
- To move to the next searched row, the user can click the F3 button.

The ‘Find and Replace’ dialog must be present to view the results, upon closing this dialog the editor highlights are reset.

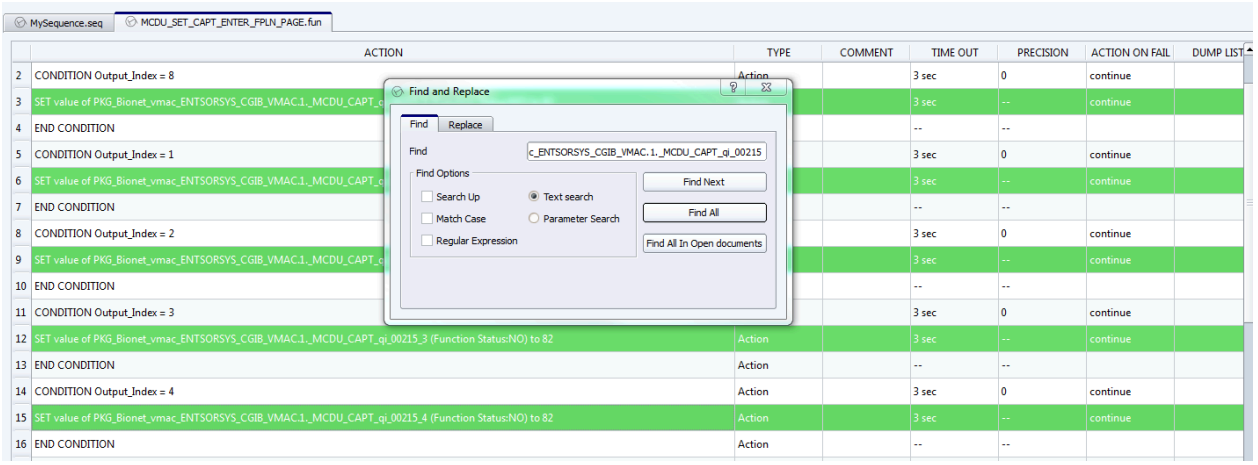


Figure: Sample ‘Find All’ operation

The user can also search for any text/parameter in all the procedures that are open in the editor.

The search results from different files are listed in the Search Result window. In the search result window, the file name, number of occurrences of the search string and rows number against the content at that row is displayed. Clicking on any row number will take the user to that specific document at that specific line and the line will be highlighted in green.

The screenshot below shows the “Find All In Open Documents” functionality.

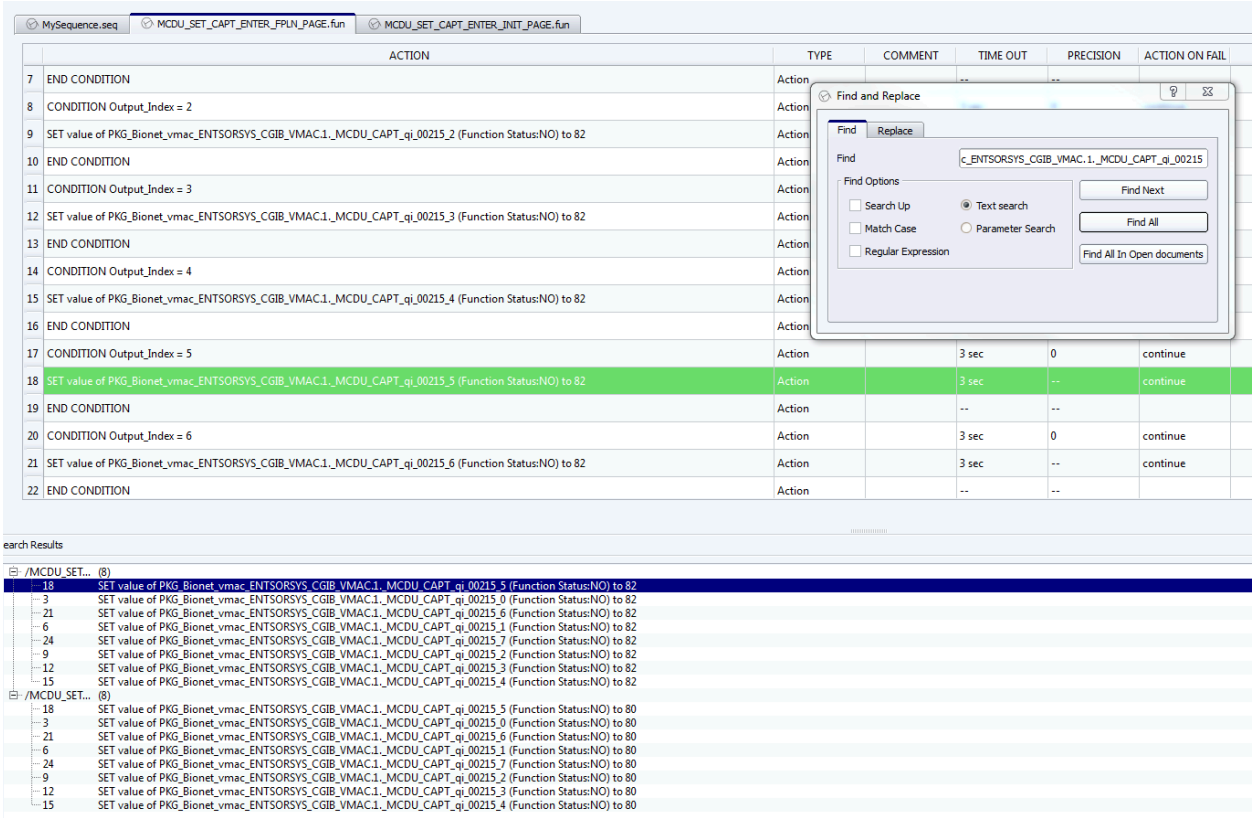


Figure: Sample 'Find All In Open Document operation

Other Find Options include:

- Searching Up or Down (default – unless checkbox for Up is checked)
- Match Case upon search operation.
- Search items making use of Regular Expressions.
- Find Next – Searches the next occurrence of an item.
- Find All – Searches all occurrence present on the editor

Note: All find operations are based on option selection for Text/Parameter search.

Replace dialog enables users to perform 'Find and Replace' on bench "Parameters" only.
Users can replace one or multiple occurrences of a searched item at the same time.

Figure below demonstrates 'Replace All' functionality – where all occurrences of a GADIRU parameter were replaced.

Note: Find & Replace feature is only applicable to loaded contents on the editor
(Example- No searches or replacements will be performed inside a call command)

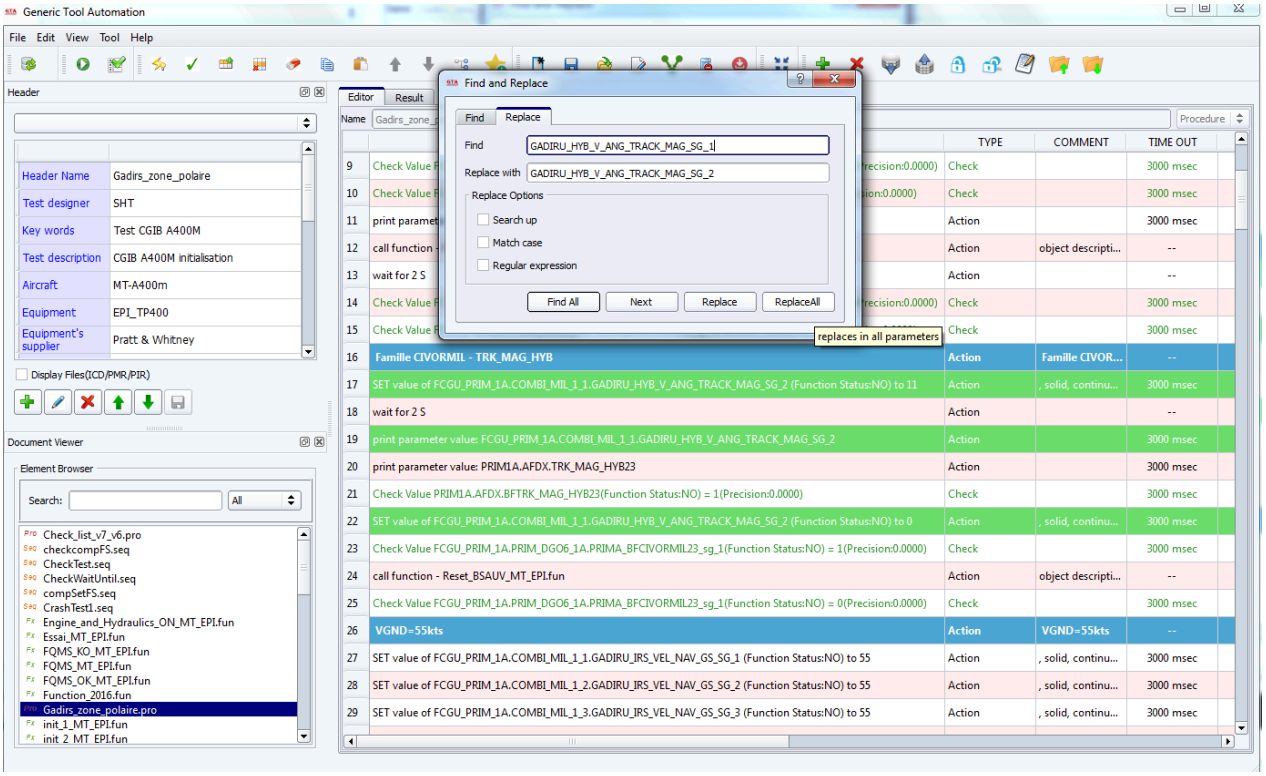


Figure: Sample 'Replace' operation

Also, "Replace All in Open Documents" will replace the content in all the files that are open in the editor.

Equipment dialog : enables the user to replace all equipment (first component from the parameter's triplet) in the file opened in the editor.

Users can give the full equipment name or part of the name in the "Equipment" text field. The matching findings will be replaced with the text provided in the "Replace with" field.

No searches or replacements will be performed inside a call command.

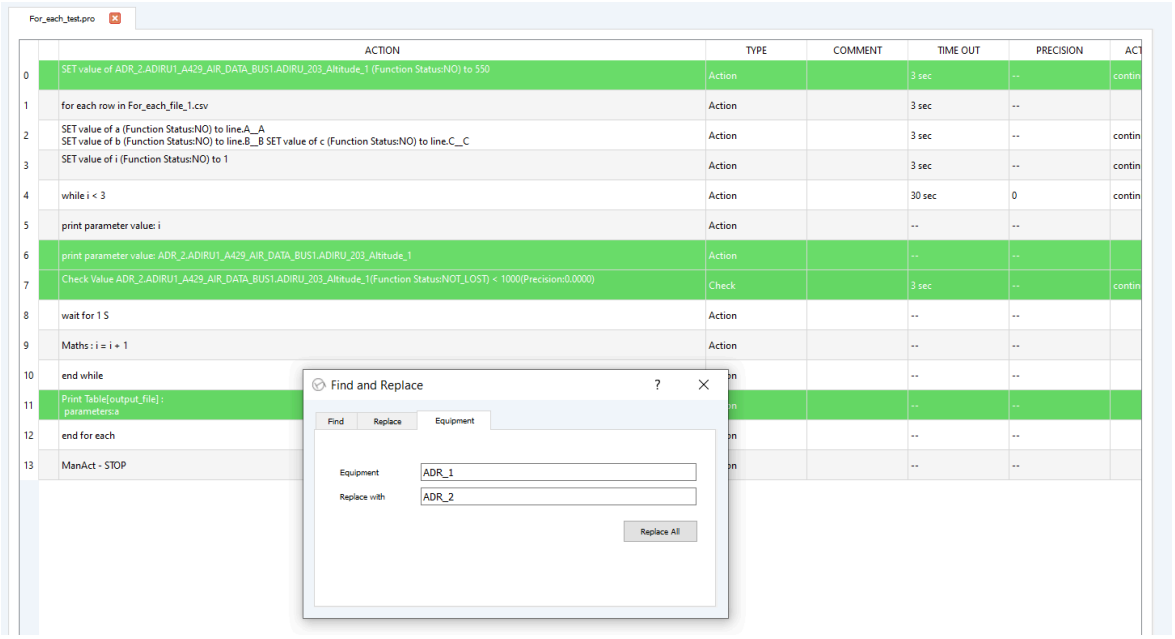


Figure: Equipment 'Replace' operation

Smart Research Dialog : This dialog enables the user to search with two options in all the documents of the database:

- Find a Requirement : Find all the elements of the database where the lines contains the given characters to find
- Find a Tag : Find all the elements of the database where the header contains the given characters to find
    - By pressing the Find Tag button, all the Headers containing the desired text will be displayed on the log section
    - Double clicking on a file opens it
    - Double clicking a second time opens the Header with the Text Highlighted

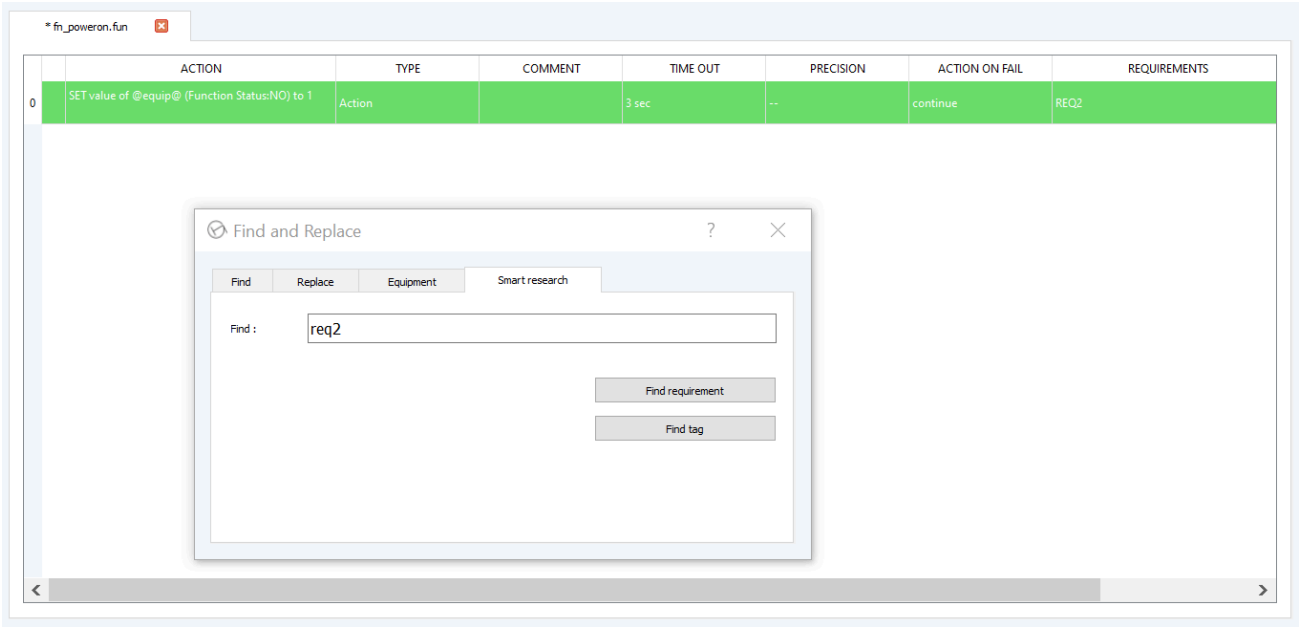


Figure: Smart research operation

3.9.2. "Export" Option

'Export' option relates to exporting a selected element and all its dependent files (Ex. Called Object,Function,Procedure) at a default folder provided by GTA or at a user defined folder.
This functionality enables users to use the exported files for verification at a different test setup, share the files for reference, use it in reports etc.

The 'Export' operation can be performed only on elements in the 'Document Viewer' section using the right click dialog menu or using the option provided at GTA toolbar.

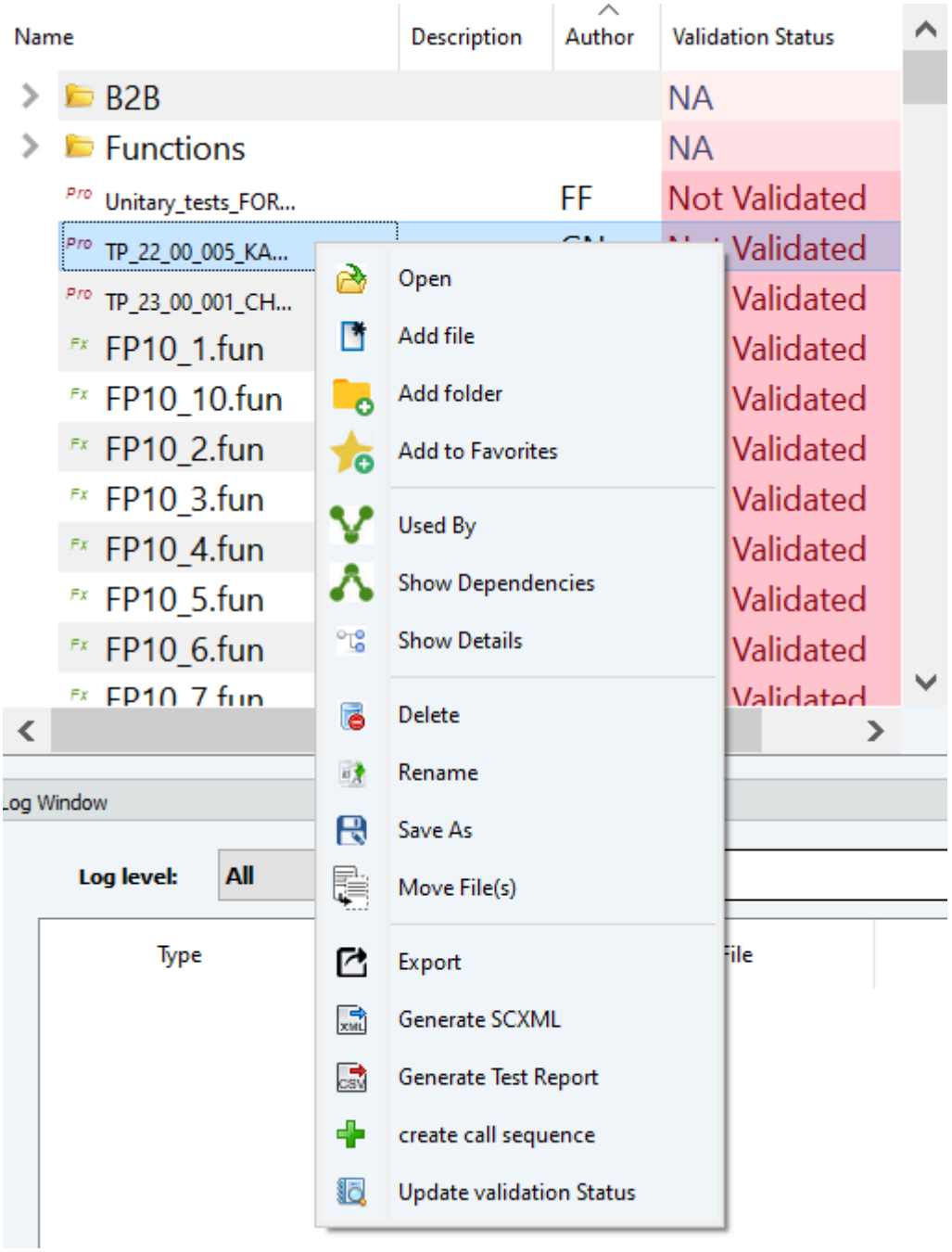


Figure: Export option

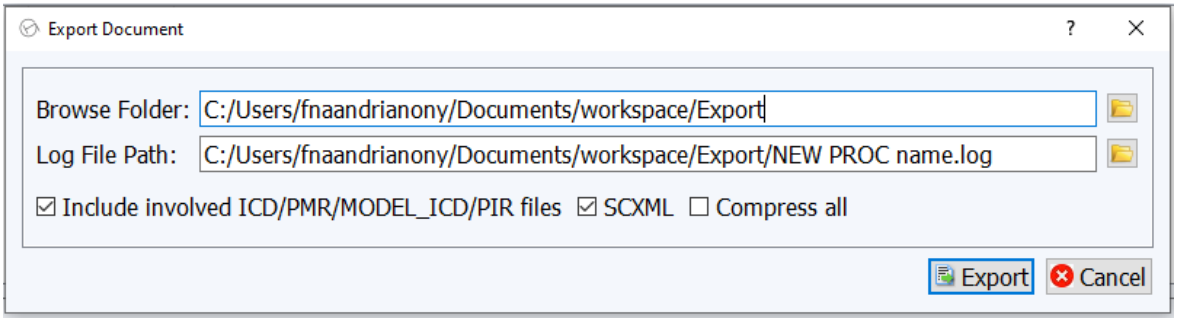


Figure: Export option – GTA default path

Options available with export:

1. Include Bench database exports all the database files to the exported folder.
2. SCXML – This will generate SCXML and export it to folder

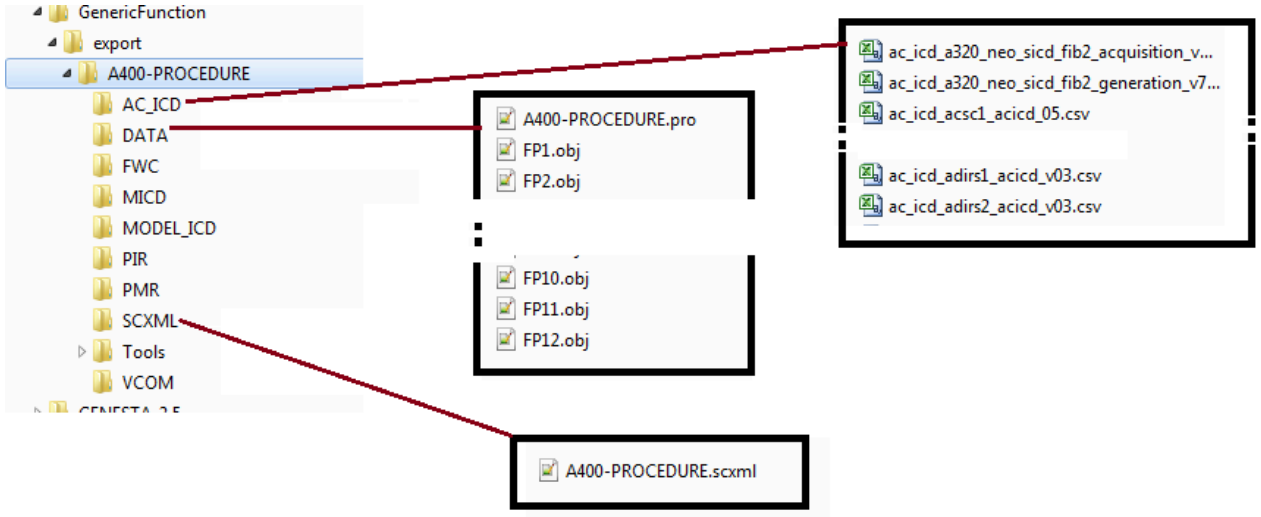
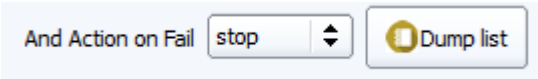


Figure: Export option – Exported folder and files

3.9.3. "Dump List" Option

For Check/Condition commands "action on fail" option is applicable, which gives users the discretion to choose the execution flow, either to continue or stop the execution, upon failure of the command.

If the user selects 'Stop' as the option for "action of fail" – GTA enables the "Dump List" option as shown below.



On clicking on the "Dump List" button, GTA provides the user with a form as below where the user can create a list of all the bench parameters for which he expects the values to be logged before stop of execution.

Note: Please refer to Action – 'Print Value' HMI to have details on usage of the HMI below.

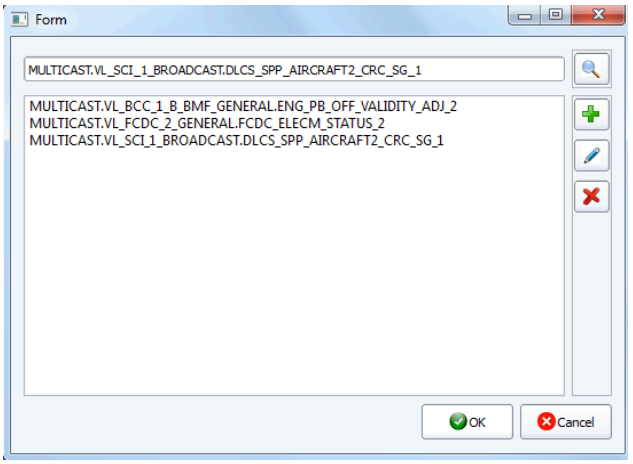


Figure: Dump List sample HMI

3.9.4. Sessions and session management in GTA

When the user exits GTA, a snapshot of the current workspace is stored as a session.
The session is a collection of all open editors. The user can manage sessions in GTA. The screenshot below shows the menu added for Session management.

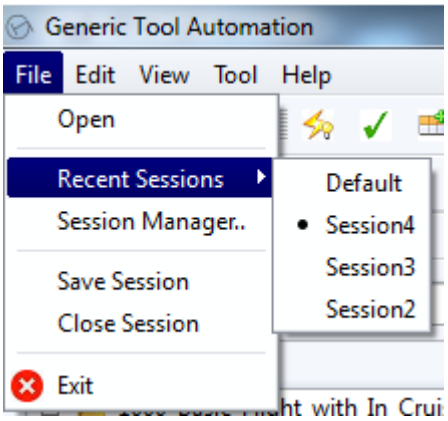


Figure: Session management

- Session Manager :
  The user can add a new session, delete a session and switch to another session.
  The session manager window supports these actions.
  There is always one session present viz. default session. If the user has not saved the session then it is considered as default.
  To create a new session the user can open any number of files and then open Session Manager and click on “New” , give a name to the session in the input box and click on the save button. The new session information will be stored and the session name will appear in a list in the Session Manager. The user can delete a session using the Delete button.
  With the “Switch to..” button the user can open/switch-to the session that has been selected in the list.

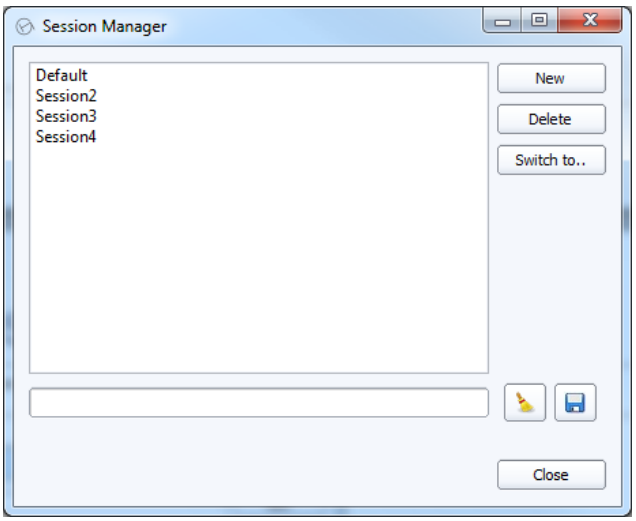


Figure: Session management

When closing a session or switching to another session or closing GTA, if there are certain files that have not been saved then a save prompt is displayed with all the files listed that have unsaved changes. The screenshot here shows the Save Prompt.

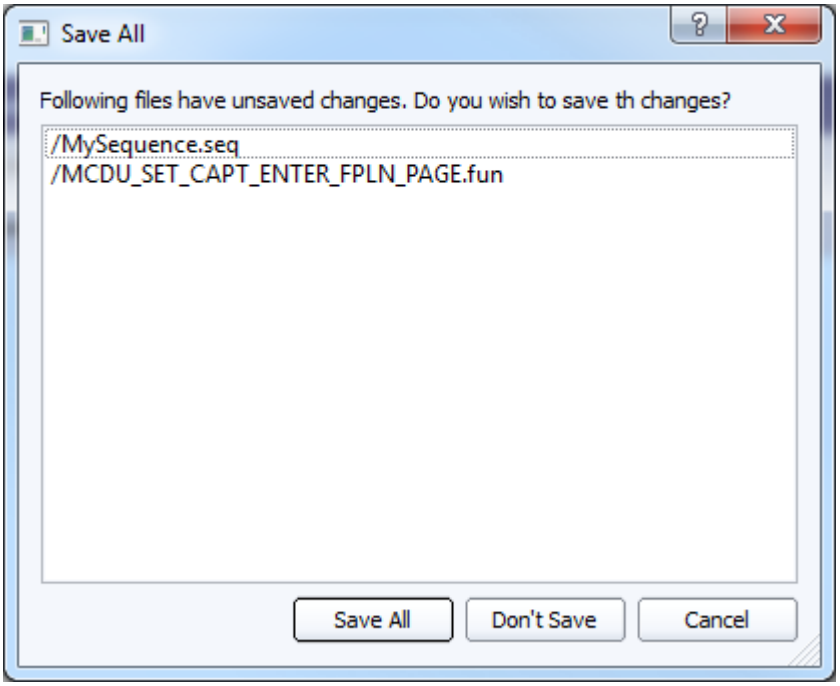


Figure: Session management - Save all

Users can say “Save All” and save all the changes or not save the changes.

- Recent Sessions: This menu shows up to 10 most recent sessions. It is kind of a shortcut to switch between recent sessions.
- Save Session: Users can use this to save the current session. The “Save All” prompt shown previously is displayed.
- Close Session: The user can close the current session. On closure of the session all editors will be closed.

3.9.5. Print a procedure

The user can save their procedures in a .docx format which they can later use to take a print. The GTA editor option for doing this is present in the File menu. File >> Save to Print.

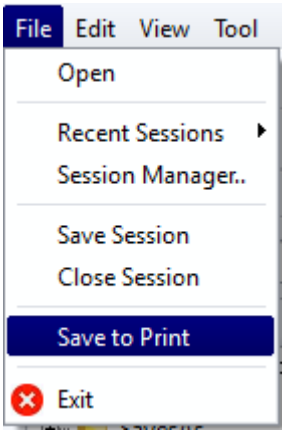


Figure: Save to print

This opens a Save As dialog, and the user can give the path where the .docx can be saved. The file type is fixed and cannot be changed.

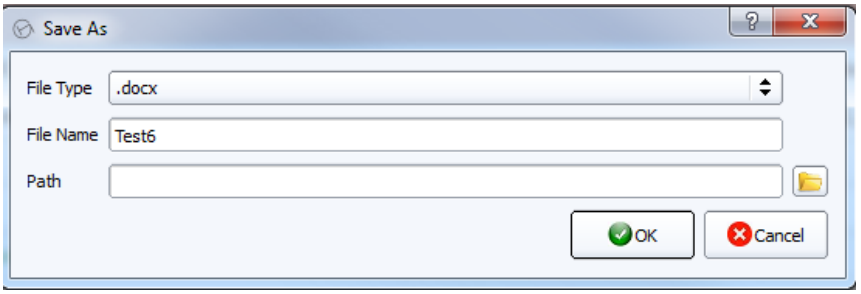


Figure: Save to print

The saved file contains the title of the procedure, header and the procedure steps. This *.docx can be now printed to have a physical copy of the procedure.

3.9.6. Delta difference between two commands

The user can check the delta difference between two commands in the result view. The difference is displayed in the log window. The difference is calculated in days, hours, min, sec and msec.

The user can select two different commands by pressing Shift key and clicking on two different commands and right clicking. The context menu gives the option of **"Display Delta"**. Click on "Display Delta" and the time difference will be displayed in the log window.

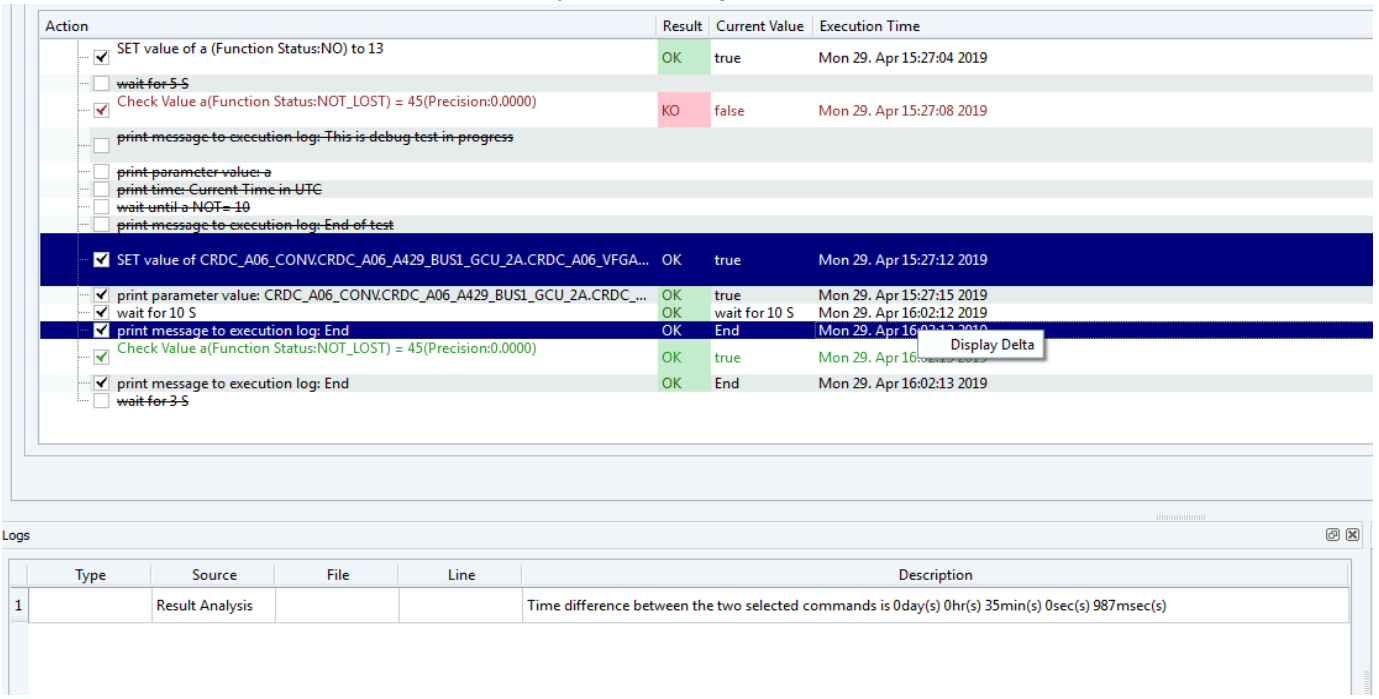


Figure: Command delta

3.9.7. Channel selection feature

Channel selection for signals is made available so that signal selection can be based on the engine and its channel.

Parameter (signals) having following string in its name:

- CHA, CHB
- EEC1_A,EEC1_B
- EEC2_A,EEC2_B
- CH_A, CH_B

Can be considered to have channel information.
Channel selection feature gives users the option to select channel information for a parameter. Note that this feature will only be available if appropriate settings are done (refer 3.8.2)

A dialog pops up on selecting parameters from the search window (double click or by pressing OK) shown below if the channel in selection information is applicable for a parameter.

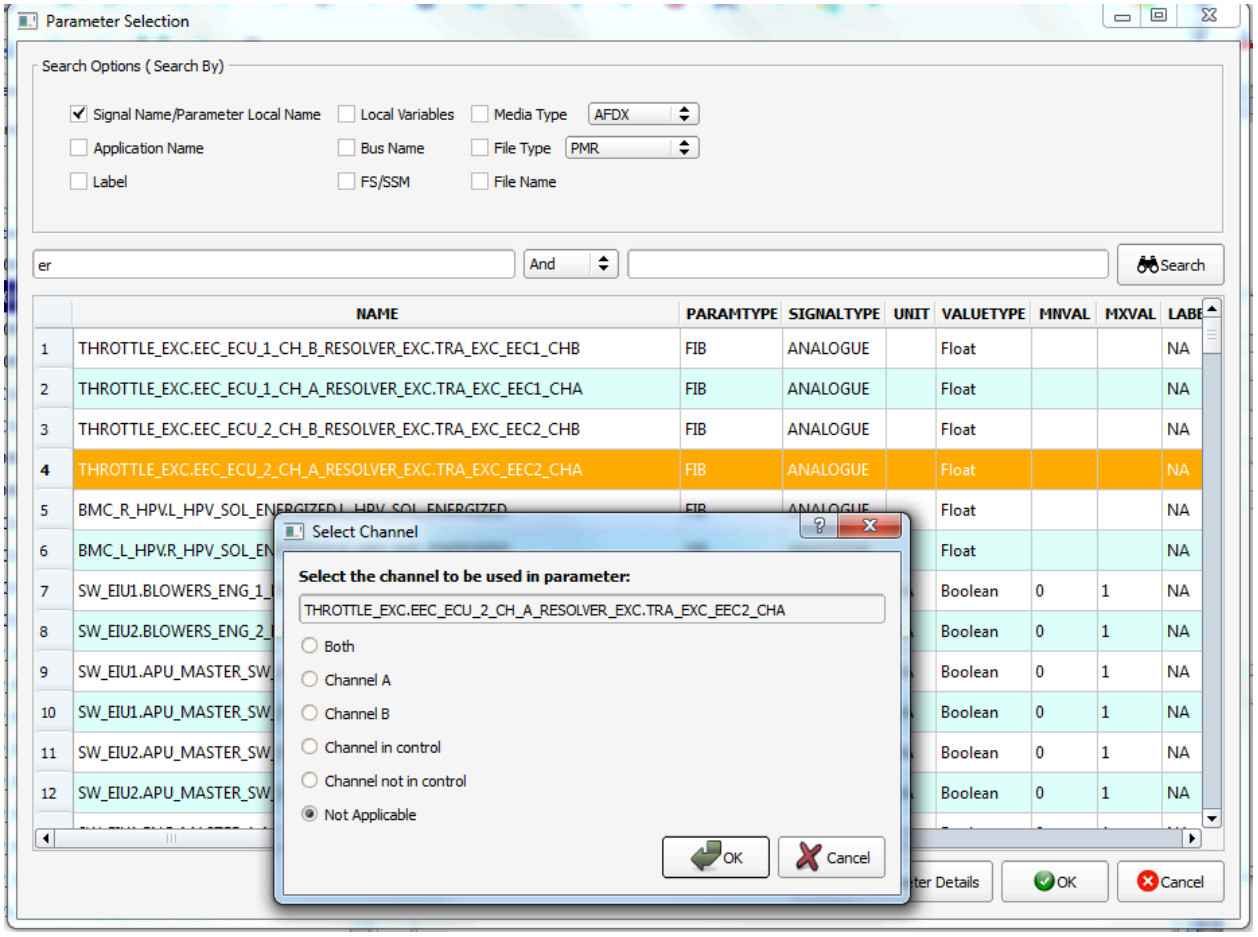


Figure: Parameter channel selection dialog

User can select the following options:

1. Both
2. Channel A
3. Channel B
4. Channel in Control
5. Channel not in Control
6. Not applicable.

Based on these selection appropriate signals along with channel information is added to the parameter line edit. Once it is added, users are advised not to edit the parameter manually through keyboard as the channel information may be lost leading to unpredictable behavior while generating SCXML.

3.9.8. Example: Show Details for Called Element

User can view the details of the called function, Procedure or Object by doing a Right-Click on the test step and selecting the ‘ Show Details’ as shown in figure below:

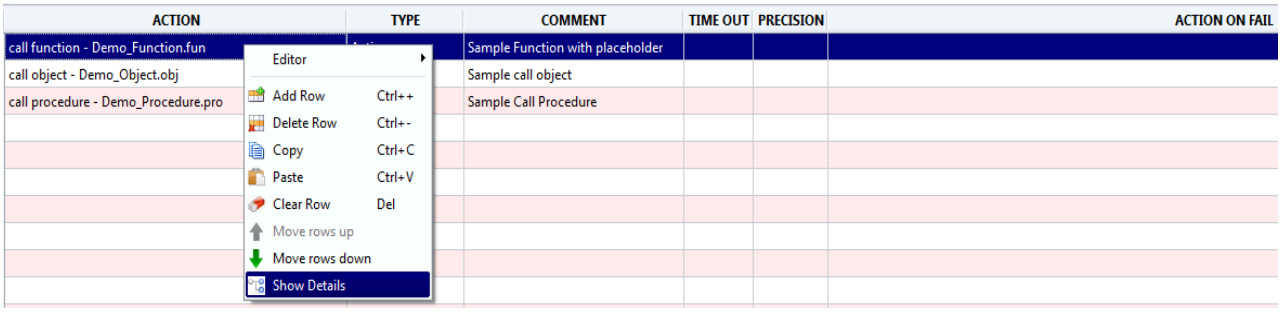


Figure: Call Function –Show Details

The ‘ Show Details’ functionality performs value replacement at the placeholder location (i.e. @placeholder@) and shows the expanded content of the element in a tree view as in figure below.

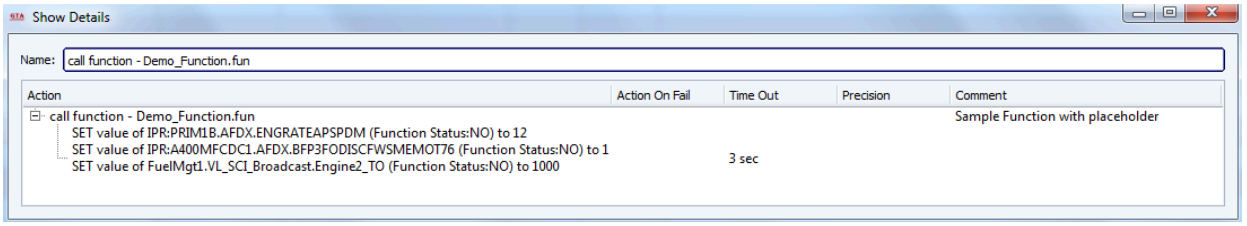
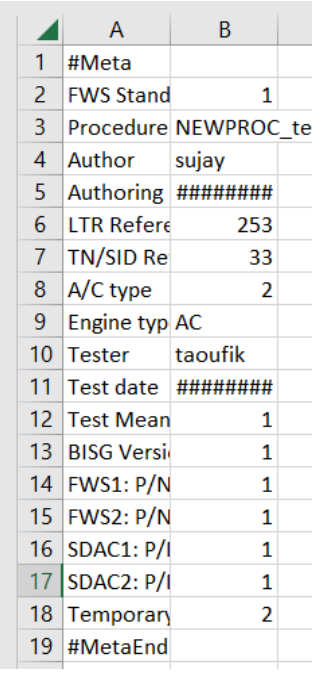


Figure: Show Details – Placeholders replaced inside Called function

3.9.9. Csv to procedure conversion

Users can convert a prepared csv file to a procedure using the GUI button or Alt + C shortcut.
The CSV file has to follow a specific format in order to be convertible.
There are two major sections of the CSV file:
First one is the meta section. Meta section has to be above the procedure section. It begins with #Meta cell and is ended with #MetaEnd cell. Users can put any meta information they deem valuable between these cells. For the purpose of conversion following cells can be used:

- Procedure: this will give name to the converted .pro file
- Author: this will be inserted into converted .pro file author section



	A	B
1	#Meta	
2	FWS Stand	1
3	Procedure	NEWPROC_te
4	Author	sujay
5	Authoring	########
6	LTR Refere	253
7	TN/SID Re	33
8	A/C type	2
9	Engine typ	AC
10	Tester	taoufik
11	Test date	########
12	Test Mean	1
13	BISG Versi	1
14	FWS1: P/N	1
15	FWS2: P/N	1
16	SDAC1: P/I	1
17	SDAC2: P/I	1
18	Temporary	2
19	#MetaEnd	

Figure: Example Meta section

Second section is the procedure section. This section is a csv representation of a test procedure.
It consists of three types of rows:

- Command name row – First row. This row contains the name of the executed command. It is used for all the cells below it. List of possible commands will be listed later in this section.
- Parameter name row – Second row. This row complements command row, it defines a variable name for which the operation specified in command name row will be executed. It is used for all the cells below it. Not all commands require this section.
- Parameter value rows – All the remaining rows. This row defines the value of the parameter which was defined above it. In general leaving this cell empty will skip the command in conversion for a given test case.

As an example a column:

SET_VAL
JML2ON
1
2

Will set the value of JML2ON to 1 in the first test case and 2 in the second test case.
New rows are generally considered to be a start of a new test case.

Limitation about cell contents: **It is forbidden to enter new line characters inside cells** (for example by using alt + enter in excel). Internally they are treated as a new line character, which causes parser to treat them as end of the current row. This makes it impossible to parse.

List of possible commands:

- **SET_VAL** – sets value of variable. Will skip in conversion if the new value is the same as the previously assigned value. Leaving this cell empty is treated as an error.
- **SET_FS** – like SET_VAL but for FS values.
- **MULTISET_VAL** – used to handle multiple sets in one procedure row. All grouped MULTISETS will be used to form one procedure row containing desired sets. Will skip in conversion if the new value is the same as the previously assigned value. Leaving this cell empty is treated as an error.

- **MULTISET_FS** – same as MULTISET_VAL but for FS values. Can be combined together with MULTISET_VAL.
- **CHECK_VAL** – checks if the value of the parameter is equal to the given value. Currently does not support other comparisons.
- **CHECK_FS** – same as CHECK_VAL but for FS values.
- **CHECK_PIR_VAL** – same as CHECK_VAL but for PIR values
- **CHECK_BITE_MESSAGE** – Check Bite message
- **TC** – Sets a test case to a given number. It is recommended to always start a row with a TC command. (does not require parameter name)
- **PRINT_VAL** – Prints given value.
- **PRINT_MESSAGE** – Print given message
- Function – Launches previously defined function with given arguments. Argument format for function is as follows:
  {var:1|other_var:2|…}
  Where *“var”*, *“other_var”* are names of parameters
  “:” is used as separator between name and value of parameter
  “1”, *“2”* are values assigned to parameter names
  “|” is used as separator between arguments name and value pairs
  “{“ and *“}”* are used to mark beginning and end of function value cell
- **To launch function with no parameters use {},** empty cells will be skipped
- **Object** – same as function but for objects, uses the same schema for assigning values to parameters.
- **CMD_WAIT_s** – launches wait function for given amount of seconds (does not require parameter name)
- **GMT_Start()** – prints current time in GMT. Used to mark the beginning of execution of a test case.
- **GMT_End()** – prints current time in GMT. Used to mark the end of execution of a test case. It is also used to mark end of procedure in csv – **cells in row after GMT_End() will not be parsed** and can be used for other purposes.
- **UTC_Start ()** and **UTC_End()** – same as GMT but in UTC. **UTC_End() is not used to mark end of procedure in csv file and cells after UTC_End() will be parsed**.

19	#MetaEnd																	
20	TC	CMD_WAI	CHECK_PI	CHECK_PI	CHECK_PI	CHECK_PI	Object	Object	Function	MULTISET	MULTISET	MULTISET	SET_FS	CHECK_VA	CMD_WAI	CHECK_VA	CHECK_FS	CHECK_VA	Function
21			PRIM1A.AI	PRIM1A.AI	PRIM1A.AI	S1C4_15.A	test_valid_	PMI_Fuel	init_AVC_(	BLWLM.2	JML2ON	SE1PFT.1	SLAILBFT.1	SE1PFT.1		FWS.FWC1	SLAILBFT.1	FWS.FWC2	GMT_End()
22	1	1	3	3	3	1	{}	{VAR1:1}	{}	1	0	NCD	FW	2	5	0	FW	0	1
23	2	3	1	5	2	1		{VAR1:2}	{}	1	0	FW	NCD		3	1		5	1
24	3	3	1	5	2	3				0	0	NO	NCD	1	4	0	NCD	0	
25																			

Figure: Example procedure section

3.9.10. Git integration

Setup

To use git functionalities within GTA Editor the user is required to first install and configure git. The workspace needs to have a git repository initialized, indicated by the .git folder existing in the workspace.

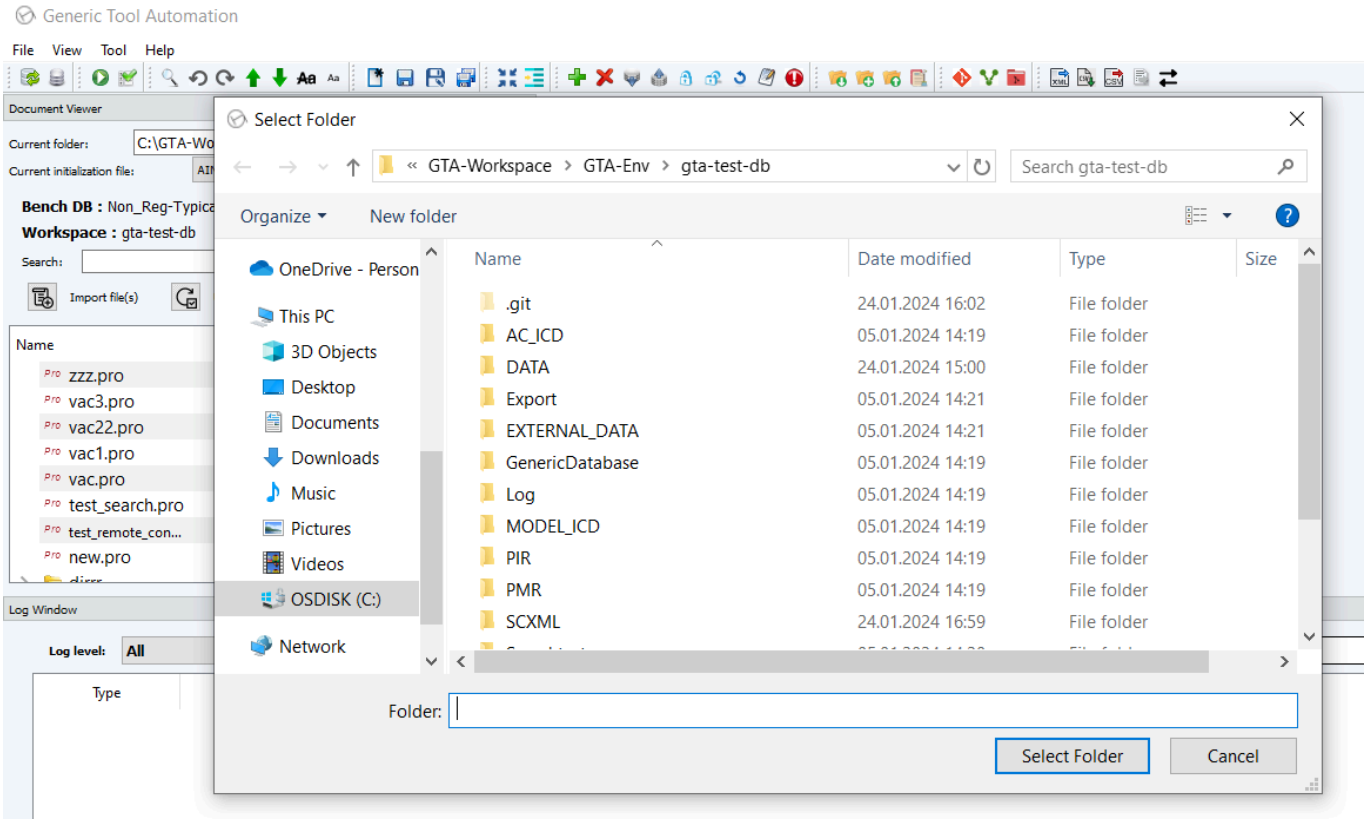


Figure: GIT function

Git init

To use Git functionalities, the user is first required to initialize Git within GTA via the Git init icon.

If the .git path was not already set in settings, this will prompt the user to select a workspace directory containing the .git folder. After successful initialization other functionalities will be available.

Git status

This command will check the status of all files in the tree view. After successful operation all files will have a new icon corresponding to their git status. The optional column git status will also be updated. Please note that there are more statuses than icons, so use git status column if you require detailed information.

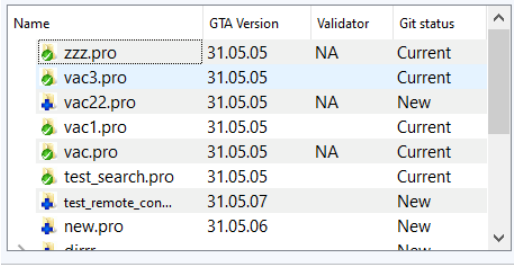


Figure: GIT status

List of icons and corresponding statuses:

New

Current

Conflicted

Ignored

Modified
Renamed

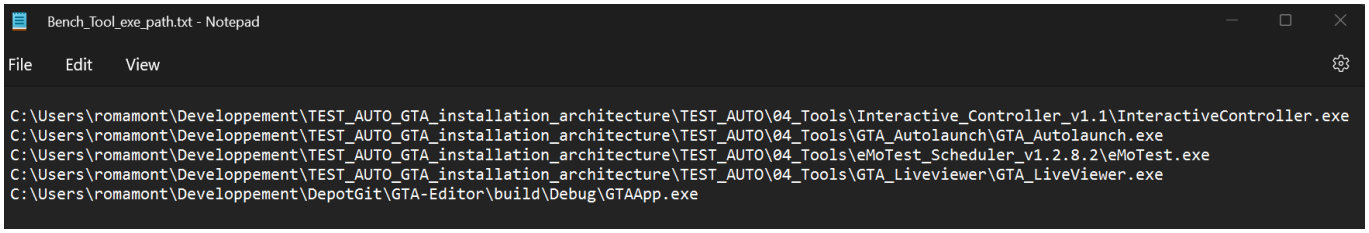
Not found

Git folder

This icon will simply open .git directory which is currently used for git operations

3.9.11. Versions Tools Bench

The first step is to redact your Bench_Tool_exe_path.txt, with different paths of exe tools on your bench.



You must place this file in the folder GTA_Local_Tool_Data.

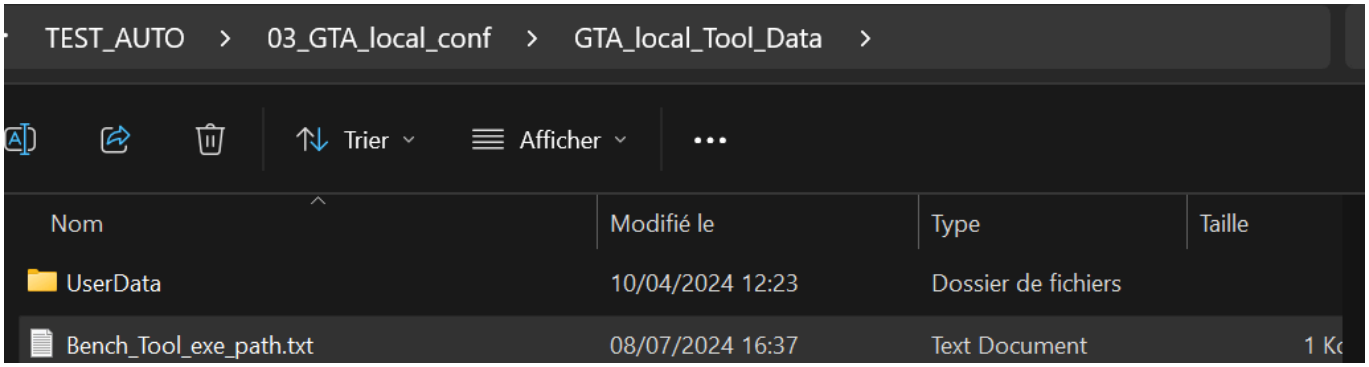


Figure: Version tool function

If you want to watch all versions of tools on your bench you can click on the onglet “Version Tools”.

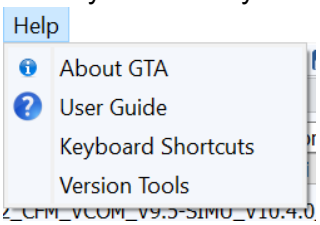


Figure: Version tool menu

When you click on it, you have a short resume of your tools and his version.

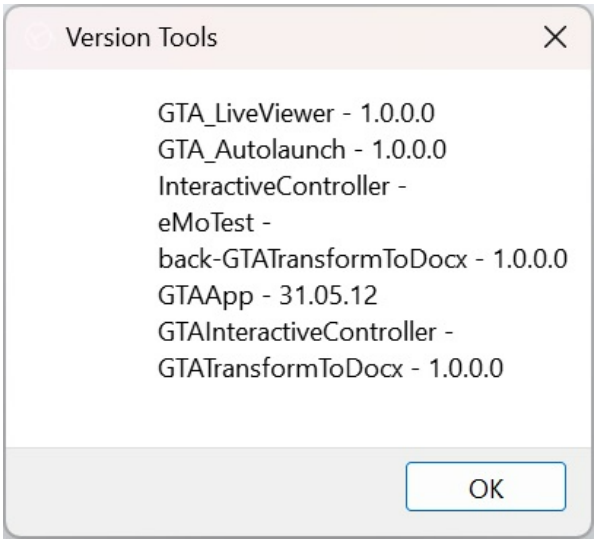


Figure: Version tool HMI

This information is added in your export of LTRA docx.
When you want to export a LTRA file, you have now a check box option to select if you want to have a new resume of versions tools that you have or keep the older.

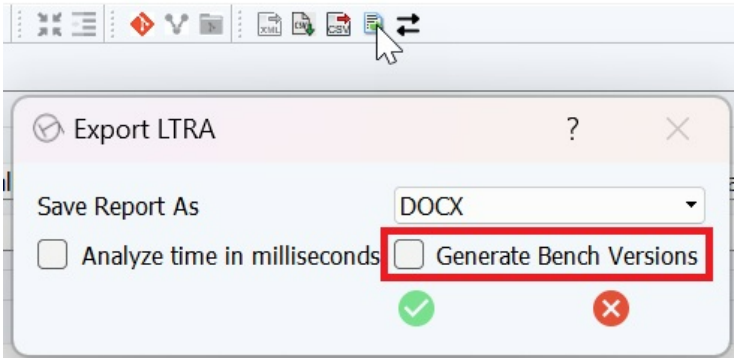


Figure: Version tool function for LTRA

If you never generated Version Tools information before export LTRA, you cannot choose. The option is forced.

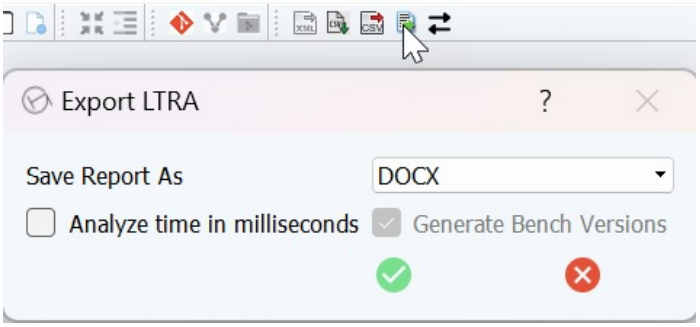
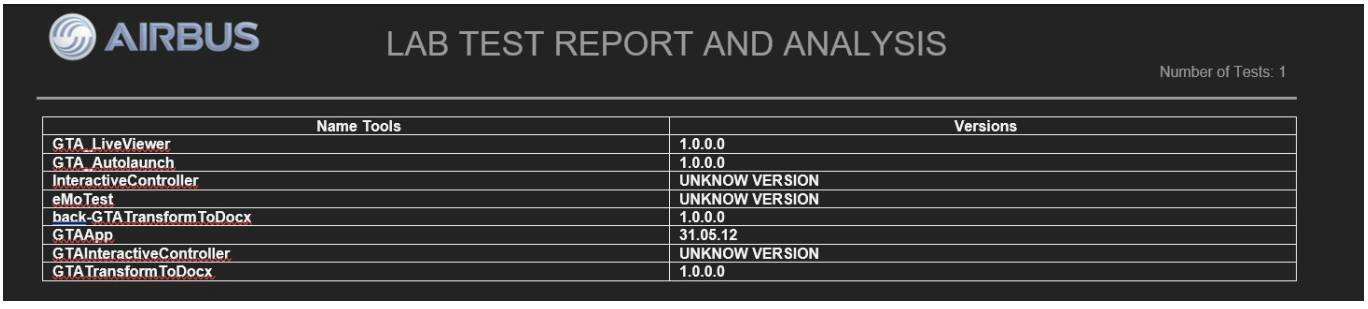


Figure: Version tool function for LTRA

At the end, when you finish exporting the docx file, you now have a page with these information of versions.



Name Tools	Versions
GTA_LiveViewer	1.0.0.0
GTA_Autolaunch	1.0.0.0
InteractiveController	UNKNOW VERSION
eMoTest	UNKNOW VERSION
back-GTATransformToDocx	1.0.0.0
GTAApp	31.05.12
GTAInteractiveController	UNKNOW VERSION
GTATransformToDocx	1.0.0.0

Figure: Version tool function for LTRA

3.9.12. Purpose and Conclusion

Purpose and Conclusion can be added in the procedure which will be displayed on the LTRA report. In the View menu; select the Purpose & conclusion menu item.

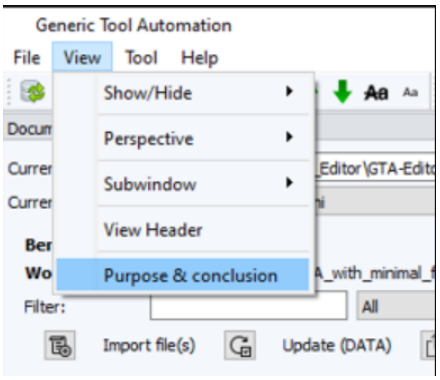


Figure: Purpose and Conclusion fields for LTRA

Purpose and Conclusion dialog will appear. Now add the text in the respective Purpose and Conclusion text boxes and Click on Ok.

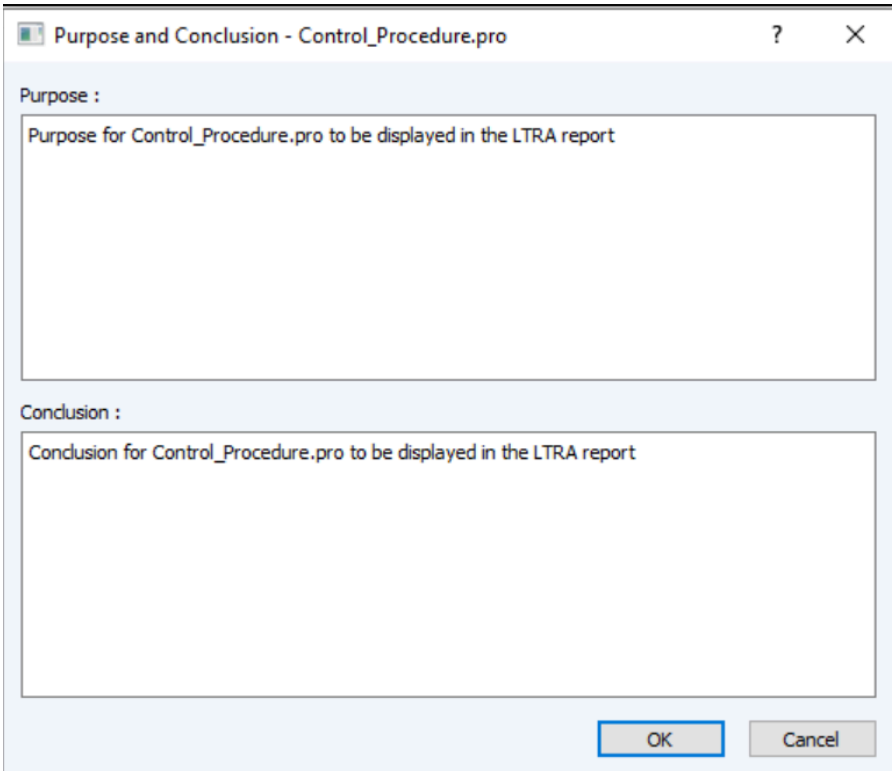
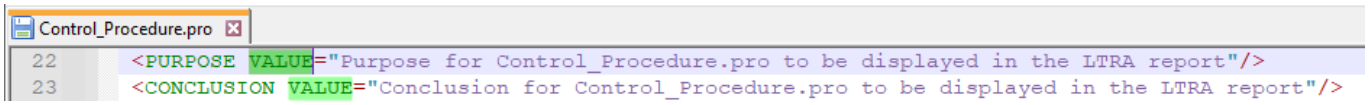


Figure: Purpose and Conclusion fields for LTRA

The procedure file is automatically updated with the entered texts for Purpose and Conclusion.



Launch the procedure and move to the Result view.

After clicking on GetLastExecution then on the Analyze button, the Result is displayed as usual, the Purpose/Conclusion checkbox gets ticked and the Purpose and Conclusion fields contain the right values. These fields are editable before exporting the LTRA report.

Notice that any changes in Purpose/Conclusion should not have any impact on the written defects.

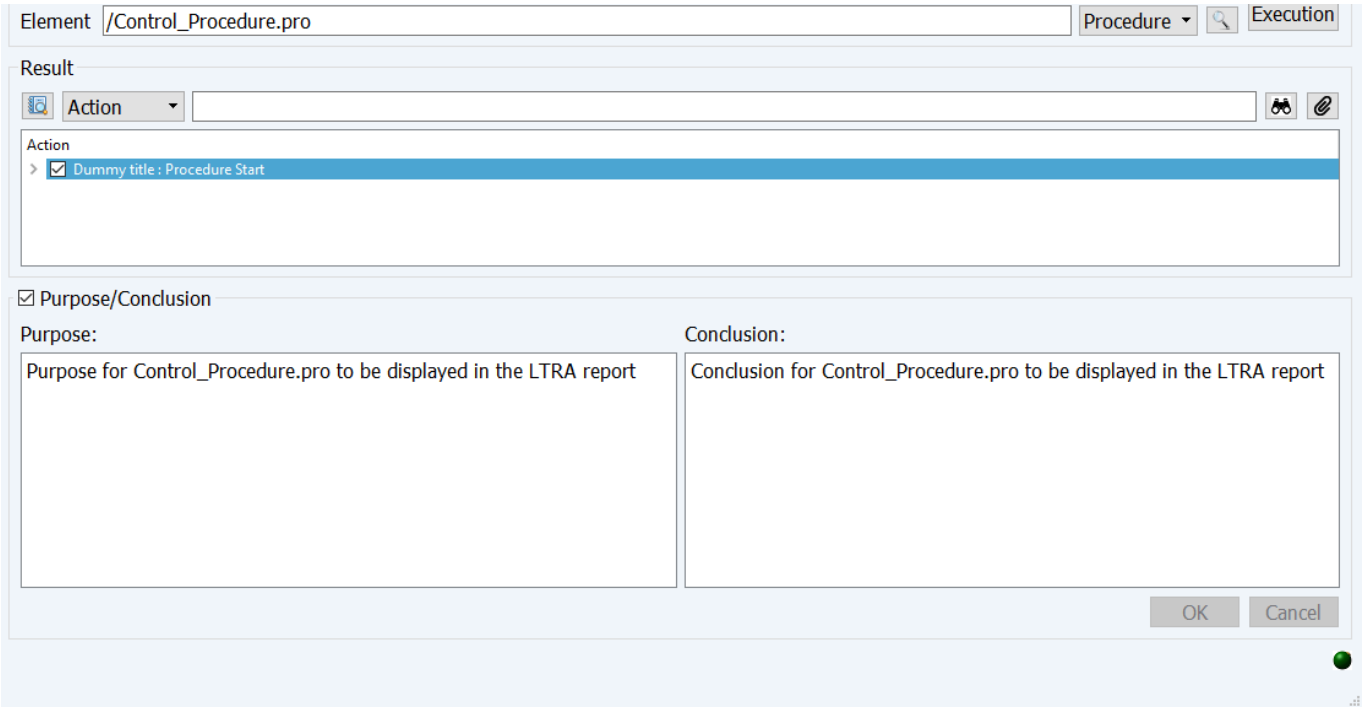


Figure: Purpose and Conclusion fields for LTRA

Purpose and Conclusion added for the procedure will be displayed in the exported document for LTRA.

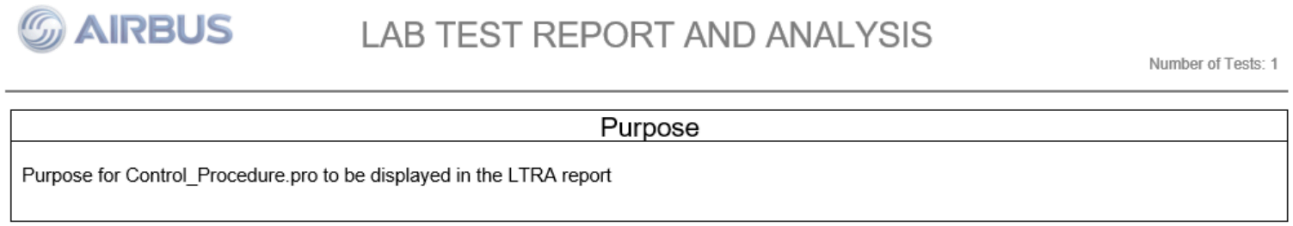


Figure: Purpose and Conclusion fields for LTRA

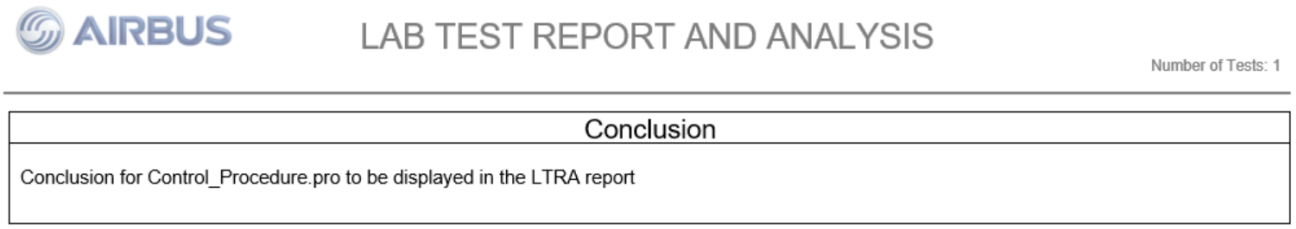


Figure: Purpose and Conclusion fields for LTRA

3.10. Terminal commands

3.10.1. Environment setup

Due to limitations of Windows Command Prompt, it is highly recommended to use a terminal backed by **MinGw64** such as **Git Bash**. Console commands will work on Windows Command Prompt, but the output will not be visible.
You can find and install Git Bash here: Git - Downloads (git-scm.com)

3.10.2. GTA terminal arguments

To launch GTA via terminal simply execute the exe file ie:

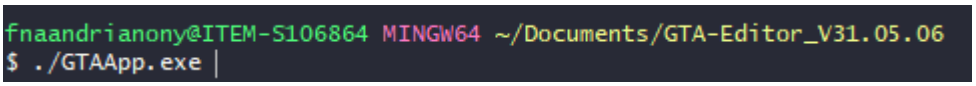


Figure: Launching GTA via terminal

Launching GTA in this way will simply launch the GUI application. Some commands will block GUI from launching and instead will only execute some specific actions. Help command is an example of action blocking GUI.
Each action has it full name as well as a shortcut, for example you can use help with “—help” or just “-h”,

To get more information about the possible arguments launch GTA with **-h or –help** command:

```
$ ./GTAApp.exe -h
List of possible arguments with description:

--help -h : Lists possible arguments and their function.
This action blocks GUI from launching.

--version -v : Returns current version of GTA Editor.
This action blocks GUI from launching.

--config= -c= : Sets init file to given value
ie. -c=C:/My/Path/my.init.

--testconfig -t : Tests if config is correct, terminates program if it's not
This action blocks GUI from launching.

--force -f : Forces overwrite of output file even if it already exists
ie. -o=existing/file.scxml --force

--input= -i= : Input for conversion.
.csv files will be converted to .pro
.pro files will be converted to scxml
Can be used together with output argument. Otherwise default output path will be used
This action blocks GUI from launching.

--enableUnsubscribeStart= -us= : Enables Unsubscribe At Start of the .scxml by default set to true
Should be used together with input argument


--enableUnsubscribeEnd= -ue= : Enables Unsubscribe At End of the .scxml by default set to true
Should be used together with input argument


--compatibilityCheck= -cc= : Checks the compatibility of the .pro file
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --compatibilityCheck=C:any.pro OR -cc=any.pro C:/Path
This action blocks GUI from launching.

--output= -o= : Output for conversion done by input command.
Can be directory (in which case filename will be copied from input) or full path with filename.
Already existing files will not be overwritten unless --force argument is also used.
ie. -o=C:/My/Path OR -o=C:/My/Path/my.scxml
This action blocks GUI from launching.

--report= -r= : Test report generation of the procedure file.
Need to be name of .pro file.
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --report=any.pro OR -r=any.pro C:/directory
This action blocks GUI from launching.

--ltra= -l= : LTRA report generation of the procedure file.
Need to be the name of .pro or .seq file.
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --ltra=any.pro OR -l=any.pro C:/directory
This action blocks GUI from launching.

Finished all actions, exiting.
```

Figure: Example output for --help command

This command will list all the other possible arguments with their explanation.

--version -v
This action will list current version of GTA

--config= -c=
This action will set config to a given path. This can be used for launching GUI but is important for file

conversion without GUI, as default path to SCXML/Procedure files is set in the config.
example:
./Gta.exe -c=C:/My/Path/my.init

--testconfig -t
This action will test if the given config is correct, should always be launched with –config
example:
./Gta.exe -c=C:/My/Path/my.init –testconfig

--force -f
Force overwrite command. By default output will not overwrite any files and will fail if a file already exists. To force overwrite attach the –force command to conversion commands.
example:
-c=C:/My/Path/my.init -i=C:/My/Input/my.csv -o=C:/My/Output/my.pro –force

--input -i=
Input for file conversion.
If this command points to a .csv file then it will be converted to a .pro file
If this command points to a .pro file then it will be converted to a .scxml file
This action can be combined with the following --output= command to specify save path. If no output is defined then default paths will be chosen. More on default paths in the output section.

--enableUnsubscribeStart= -us=
Enables Unsubscribe At Start of the .scxml by default set to true
Should be used together with (–input, -i) argument
examples :
-us=false -i=FullPathPro.exe
-us=true -i=FullPathPro.exe

--enableUnsubscribeEnd= -ue=
Enables Unsubscribe At End of the .scxml by default set to true
Should be used together with (–input, -i) argument
examples :
-ue=false -i=FullPathPro.exe
-ue=true -i=FullPathPro.exe

--compatibilityCheck= -cc=
Checks the compatibility of the .pro file
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --compatibilityCheck=C:any.pro OR -cc=any.pro C:/Path
This action blocks GUI from launching.

--output= --o=
Output for file conversion.
This command will change the save path for file conversion.
If not used then default path will be used (GTA DATA folder for saving .pro and SCXML folder for saving .scxml). Please remember that without config GTA will not know it’s default paths. Filename will be read from csv (for csv conversions) or copied from input.
It's possible to input only a directory, in this case default filename will be used.
It's also possible to enter a full path.
examples:
-c=C:/My/Path/my.init -i=C:/My/Input/my.csv
-c=C:/My/Path/my.init -i=C:/My/Input/my.csv -o=C:/My/Output/
-c=C:/My/Path/my.init -i=C:/My/Input/my.csv -o=C:/My/Output/my.pro

--report= -r=
Test report generation of the procedure file.
Need to be name of .pro file.
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --report=any.pro OR -r=any.pro C:/directory
This action blocks GUI from launching.

--ltra= -l=
LTRA report generation of the procedure file.
Need to be the name of .pro or .seq file.
Optional: The user can specify the folder where to generate the test report, by default it's generated in /output directory.
ie. --ltra=any.pro OR -l=any.pro C:/directory
This action blocks GUI from launching