Lockheed Martin Advanced Technology Laboratories
One Federal Street, A&E Building
Camden, NJ 08102

LOCKHEED MARTIN

Interoffice Memo

RASSP-EN-068

Date: 03/03/97

To: Distribution

From: J. J. Welsh Telephone: (609)-338-4223

Subject: ***Misc Enterprise Software Documentation (11/95)***

The following text (generated by Intergraph) describes custom developments / commands developed for the enterprise framework.

_____

John J. Welsh

xc: Enterprise Internal Distribution

J. Brown

B. Chadha

B. Kalathil

I. Holmes

P. Holmes

J. Stavash

ProjList Documentation 11/95

Reference

RASSP document on Project/ACL

Assumptions

Environment variables:

RASSP_PROJECT_LIST points to RASSP project list file. This can be
overridden using command line option -l
e.g. RASSP_PROJECT_LIST=$RASSP_CUSTOM_PATH/ingr/config/rassp_proj.lst

RASSP_PROJECT_LOG points to a log file where all error/messages are
saved. This can be overridden using command line option -e
e.g. RASSP_PROJECT_LOG=/tmp/rassp_proj.log
NIS:

The NIS group file should contain a group named rassp_proj_mgr

Syntax

Usage: projlist [-hian] [-p proj_name] [-l list_file] [-e log_file]

-h: print this message

-i: run the program in interactive mode

-a: run the program in automatic mode

-n: do not start DM2.0

-p proj_name: start Project Manager for project 'proj_name'

-l list_file: specify the project list file name

-e log_file: specify the project log file

Notes

1. In interactive mode, users are presented with menus for any action

2. In automatic mode, the Workflow Displayer and the DM2.0 interface will be started automatically if possible.

3. The -n option will suppress the automatic invocation of DM2.0 interface. This option is valid with -a option only.

4. The -p option allows users to start a project directly, bypassing the project list manager.

To find out how many objs match query:

qrycli -a count -z <class> -t <# of sets> -r "<attr/value sets>"

To retrieve the value of a given attribute:

qrycli -a getattr -z <class> -t <# of sets> -r "<attr/value sets>"

-s <attribute to retrieve>

To retrieve the value of a given attribute for a related item:

qrycli -a getrelattr -z <class> -t <# of sets> -r "<attr/value sets>"

-v "<relation class>,<relationship>"

-s <attribute to retrieve>

Options:

-a <action>

-z <class>

-t <number of attribute/value sets>

-r "attribute1{attribute value1},attributeN{attribute valueN}"

-s <attribute to retrieve>

-v "<relation class>,<relationship>"

Examples:

qrycli -a count -z WorkItem -t 1 -r "OwnerName{pdm}"

qrycli -a getattr -z TextItem -t 2 -r "OwnerName{pdm},OwnerDirName{PDM Work}"

-s FullPath

qrycli -a getrelattr -z Document -t 2 -r "OwnerName{pdm},OwnerDirName{PDM Work}"

-v "Attach,DataItemsAttachedToBusItem"

-s FullPath

Assumptions

Environment variables:

RASSP_TOOL_FILE_PATH points to the directory where all the tool

files are. This can be overridden using command line

option -l

e.g. RASSP_TOOL_FILE_PATH=$RASSP_CUSTOM_PATH/ingr/tools

RASSP_TOOL_ICON_PATH points to a directory which holds all icon
bitmap files. This can be overridden using command line
option -i
e.g. RASSP_TOOL_ICON_PATH=$RASSP_CUSTOM_PATH/ingr/icon

Syntax

Usage: toolpad [-h] -d dir -p proc_name -t tool_arg [-l tool_file_path]
[-i tool_icon_path]
-h: print this message
-d dir: directory to which the workflow attaches.
-p proc_name: name of the process from which this application is
invoked. Put the proc_name in quotes if it contains spaces.
-t tool_arg: specify the tools to be invoked. Put the tool_arg
in quotes if it contains spaces.
tool_arg := tool {tool}
tool := tool_name [({arg})]
tool_name := tool name
arg := command line argument for the tool. Cannot
contain '(' or ')'.
-l tool_file_path : full path to the dir where tool files locate.
This overrides the env RASSP_TOOL_FILE_PATH

-i tool_icon_path : full path to the dir where tool icon files locate.

This overrides the env RASSP_TOOL_ICON_PATH

Example

toolpad -d /opt/dm2.0/project/test.prj -p "Edit Schematic" -t "aceplus xterm (-e vi afile.txt)"

this example specifies the workflow attached to a project directory

"/opt/dm2.0/project/test.prj", process named "Edit Schematic", and 2

tools "aceplus" and "xterm", aceplus will be launched without argument,

while xterm will be started with vi editor running on file "afile.txt".

Two tool files, aceplus.tol and xterm.tol, are assumed to have been

defined in RASSP_TOOL_FILE_PATH.

Tool File Format - 11/95

I. Format of a tool file:

Tool File Format 1.0.0

tool_name:

$TOOL_ARG_LIST

arg1_name:arg1_value:

arg2_name:arg2_value:

...

$END

displayed tool name:tool_path_name::icon_file:command argument:

$TOOL_ENV_LIST

env1_name:env1_value::

env2_name:env2_value::

...

$END

II. Example: content of tool file da.tol

Tool File Format 1.0.0

da:

$TOOL_ARG_LIST

MentrDir::

A_Label:default label:

$END

MGC Design Architect ($A_Label):new_da::da.bmp:$MentrDir:

$TOOL_ENV_LIST

MGC_LOCATION_MAP:$MentrDir/mgc_loc_map::

$END

III. Comments:

1. tool file must have extension .tol

e.g. da.tol

2. all line starts from the first column

3. fields are separated by ':', therefore no ':' is allowed in any field

4. tool_name ususlly is the base name of the tool file

e.g. da vs da.tol

5. the number of arguments in TOOL_ARG_LIST section indicates the number of arguments required when the tool is invoked. The values of the arguments will be passed in when the tool is invoked. The order of the pass-in arguments is the order of arguments in TOOL_ARG_LIST section.

6. arguments in TOOL_ARG_LIST section can be referenced in fields 'displayed tool name', and/or 'command argument', and/or 'env_value' with the format '$arg_name'. The value of the argument will be used wherever it is referenced.

e.g. argument MentrDir is referenced as $MentrDir, as the result

$MentrDir will be replaced by the value of argument MentrDir,
whatever it will be, in the runtime

7. no space is allowed in fields arg_name, tool_path_name, icon_file, env_name

8. the field 'displayed tool name' will be used to describe the tool in
the tool launch pad. If multiple instances of a tool is needed for
different data/argument, it would be helpful to use a label in the
'displayed tool name' to distinguish one from the other.

e.g. if the da tool is invoked for 2 schematics: sch1 and sch2,
then invoke the toolpad with two instances of da tool and
pass in sch1 and sch2 as the second argument (A_Label) of
da instance1 and da instance2, respectively. As the result,
the icons in the toolpad will be labeled as 'MGC Design
Architect (sch1)' and 'MGC Design Architect (sch2)',
respectively.

9. tool_path_name is the executable path name. full path is not necessary
if the tool_path_name is in the search PATH

10. the final command line will be formed as
'tool_path_name' 'command argument'

11. the env_name are set to env_value before the tool is invoked and they become part of the environment variables

Assumptions

Environment variables:

RASSP_TOOL_LIST points to a tool list file. This can be overridden using command line option -l
e.g. RASSP_TOOL_LIST=$RASSP_CUSTOM_PATH/ingr/config/rassp_tool.lst

RASSP_TOOL_ICON_PATH points to a directory which holds all icon bitmap files. This can be overridden using command line option -i
e.g. RASSP_TOOL_ICON_PATH=$RASSP_CUSTOM_PATH/ingr/icon
CURRENT_PROJ_DIR points to the current RASSP project's full path
CURRENT_PROJ_NAME holds the name of the current RASSP project
These two variables are set by the project manager. Thus if toolwin is invoked within workflow displayer which

is in turn invoked by projlist application, user does not need to set these two variables. If toolwin is invoked in a user's own environment (e.g. user's login shell), then these two variables have to be set.

Syntax

Usage: toolwin [-h] -p proc_name -t tool_arg [-l tool_list]

-h: print this message

-p proc_name: name of the process from which this application is invoked. Put the proc_name in quotes if it contains spaces.

-t tool_arg: specify the tools to be invoked. Put the tool_arg in quotes if it contains spaces.

tool_arg := tool {tool}

tool := toolcmd [(user_arg {user_arg})]

toolcmd := command line string to invoke the tool

user_arg := username [(arg_list)]

username := user login name. Wild card '*' for any user.

arg_list := command line argument for the toolcmd. Cannot contain '(' or ')'.

-l tool_list: tool list file name

-i icon_path: tool icon path

Notes

1. Username feature is not supported yet, just use '*' for username

Example

toolwin -p "Edit Schematic" -t "aceplus(*()) xterm (*(-e vi afile.txt))"

this example specifies the workflow process name to be "Edit Schematic", and 2 tools "aceplus" and "xterm", aceplus will be launched without argument, while xterm will be started with vi editor running on file "afile.txt".

Vault Location CLI (VLCLI)

The following are the valid ways to use vlcli:

vlcli -a delete -v <vault> -o <vault location>

vlcli -a create -v <vault> -o <vault location> -n <full path name> -t <host name>

Remember: "rulefile" is invoked async from vlcli (i.e. ignore the WARNING about Rule Cache out of date)