C L S T o o l s
- Quick Reference -

INHALTSVERZEICHNIS / TABLE OF CONTENT

1. THE MDB TOOL 3

SUBCOMMANDS: 4
PARAMETERS: 4
OPTIONS: 5
PLACEHOLDERS FOR GENERAL ATTRIBUTES: 6
PLACEHOLDERS FOR SOURCE CODE: 6
INPUT/OUTPUT REDIRECTION: 6

2. THE DECODER 7

PARAMETER: 7
OPTIONS: 7

3. THE ASSEMBLER 8

PARAMETER: 8
OPTIONS: 8

4. THE DEBUGGER 9

PARAMETER: 9
OPTIONS: 9
INPUT COMMANDS 9

5. THE ADA2TPS TOOL 10

PARAMETER: 10
OPTIONS: 10

6. THE UCL2TPS TOOL 11

PARAMETER: 11
OPTIONS: 11

7. THE FORMAT TOOL 12

PARAMETER: 12
OPTIONS: 12

8. DATABASE ENVIRONMENT SPECIFICATION 13

1. The MDB Tool

The MDB tool retrieves, stores, clears or lists items in the database.

Mdb **<- MDB access tool**
[-environment <value>...="$MDA_ENVIRONMENT"]
[-verbose] <- print verbose output

list **<- list (sub)tree or end item**
[<item>="/"] <- path name or SID of (sub)tree or item
or object to be listed
[-selection <value>...] <- restrict output to subset of
item types or pathnames

compare **<- compare two CCUs or CDUs**
<item> <- path name or SID of (sub)tree or item
<2nd_environment> <- second environment
[<filename>] <- output file name
[-selection <value>...] <- restrict iteration to subset of
item types or pathnames
[-source] <- compare source code too
[-onboard_view] <- use database on-board view
[-temp_directory <value>="/tmp"] <- directory for temporary files

retrieve **<- retrieve item contents into files**
<item> <- path name or SID of item
[<filename>] <- target file name
[-source [<value>]] <- retrieve source
[-specification [<value>]] <- retrieve spec. source
[-code [<value>]] <- retrieve I-code
[-symbol_table [<value>]] <- retrieve symbolic table
[-debug_table [<value>]] <- retrieve debug table
[-xref_list [<value>]] <- retrieve xref list
[-spec_xref_list [<value>]] <- retrieve spec. xref list
[-parameter_list [<value>]] <- retrieve parameter list
[-onboard_flag] <- retrieve onboard flag
[-id] <- retrieve system library id
[-interactive_flag] <- retrieve interactive flag

store **<- store item contents from files**
<item> <- path name or SID of item
[<filename>] <- target file name
[-source [<value>]] <- store source
[-specification [<value>]] <- store spec. source
[-code [<value>]] <- store I-code
[-symbol_table [<value>]] <- store symbolic table
[-debug_table [<value>]] <- store debug table
[-xref_list [<value>]] <- store xref list
[-spec_xref_list [<value>]] <- store spec. xref list
[-parameter_list [<value>]] <- store parameter list
[-onboard_flag <value>] <- store onboard flag
[-id <value>] <- store system library id
[-interactive_flag <value>] <- store interactive flag

clear **<- delete item contents**

	<item>	<- path name or SID of item
	[-source]	<- delete source
	[-specification]	<- delete spec. source
	[-code]	<- delete I-code
	[-symbol_table]	<- delete symbolic table
	[-debug_table]	<- delete debug table
	[-xref_list]	<- delete xref list
	[-spec_xref_list]	<- delete spec. xref list
	[-parameter_list]	<- delete parameter list
extract		**<- extract name tree definition file**
	<item>	<- path name or SID of (sub)tree or item
	[<filename>="/dev/tty"]	<- output file name
	[-selection <value>...]	<- restrict output to subset of item types or pathnames
	[-parents [<value>]]	<- list parent subtrees
iterate		**<- iterate a program over items in (sub)tree**
	[<item>="/"]	<- path name or SID of (sub)tree or item
	[-selection <value>...]	<- restrict iteration to subset of item types or pathnames
	[-execute <value>]	<- program command line including placeholders (in quotes!)

Subcommands:

List Lists items or subtrees from the database.

mdb list	same as list "\"
mdb list "\"	list uppermost level of database tree
mdb list "\..."	list database tree recursively
mdb list "\some\path"	list first level of a subtree
mdb list "\some\path..."	list subtree recursively
mdb list "\some\path\some_item"	list details of an item
mdb list item_types	list all existing item types

compare Compares the contents of two CCU/CDU versions.

Retrieve Retrieves data components from an item and stores them in files.

Store Stores data components of an item from file into the database.

Clear Deletes the contents of an item in the database.

Extract Extracts the name tree definition in a form that can be used to configure the emulated database.

Iterate Iterates over all items in a selected region of the database (see list) and executes a command line for each item.

Parameters:

Item path name or SID of a database subtree or end item

Filename	Name of the target or source file, respectively. If omitted, the file name will be derived from the item name with a standard extension specific to the data component.
2nd_environment	The second environment (CCU/CDU version) for the compare subcommand. See "Database Environment Specification" at the end of this file.

Options:

-environment <value> ...	Selects the database environment. See "Database Environment Specification" at the end of this file.
-source [<value>]	Read/write source code from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix ".ucl".
-specification [<value>]	Read/write specification source from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix "_.ucl".
-code [<value>]	Read/write I-code from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix ".bin".
-symbol_table [<value>]	Read/write symbol table from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix "_.sym".
-debug_table [<value>]	Read/write debug table from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix ".sym".
-xref_table [<value>]	Read/write cross-reference table for the body from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix "_.xref".
-spec_xref_table [<value>]	Read/write cross-reference table for the specification from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix ".xref".
-parameter_list [<value>]	Read/write formal parameter list from/to database. <value> = optional file name. If omitted, the file name will be derived from the item name with the suffix ".par".
-onboard_flag <value>	Sets onboard flag.
-onboard_view	Uses the onboard (unmapped) view on item types and attributes.
-temp_directory <value>	Directory for temporary files, default: /tmp.
-id <value>	Sets ID of a system library.
-interactive_flag <value>	Sets the interactive flag for a library.
-selection <value> ...	Restricts the output to a selected group of items.

Each <value> in a list of values can either be a pathname pattern (if it starts with '\') or an item type pattern, both can use wildcards ('*', '?').

-execute <value> Enter a UNIX command line to be executed.
The command line must contain at least one of the following placeholders that will be replaced before executing the command:

Placeholders for general attributes:

{}	pathname (short form for {pathname})
{pathname}	pathname
{sid}	SID
{item_type}	item type
{sw_type}	software type in UCL syntax
{access_class}	access class
{owner}	owner
{ci_number}	CI number
{status}	status
{subitem_flag}	subitem flag (TRUE, FALSE)
{parameter_flag}	parameter flag (TRUE, FALSE)
{onboard_flag}	onboard flag (TRUE, FALSE)

Placeholders for source code:

{source}	name of a temporary file that contains the source text of the body (file is read-only)
{specification}	name of a temporary file that contains the source text of the specification (file is read-only)

Input/output redirection:

The called program may read the source text from standard input and write modified source text on standard output, using the following placeholders:

{<source}	program will read the source of the body from standard input
{<specification}	program will read the source of the specification from standard input
{>source}	program will write the source of the body to standard output
{>specification}	program will write the source of the specification to standard output

Other input/output redirection may be used in the usual way, but only one input redirection or output redirection, resp., may be used in a command call.

2. The Decoder

The decoder decodes a binary I-code unit, symbol table, debug table cross-reference table or formal parameter list to its symbolic (assembler) equivalent from the database. The files get a standard suffix:

- I-Code assembler file: '.ass'
- symbol table assembler file: '_.tab'
- debug table assembler file: '.tab'
- XREF table (body): '.xref'
- XREF table (specification): '.xref'
- formal parameter list: '_.ucl'

decode <- translate binaries to symbolic text
<item> <- MDB item pathname or SID
[-environment <value>...="$MDA_ENVIRONMENT"] <- MDA environment spec.
[-code [<value>]] <- disassemble I-code
[-symbol_table [<value>]] <- disassemble symbol table
[-debug_table [<value>]] <- disassemble debug table
[-spec_xref_table [<value>]] <- disassemble spec xref table
[-xref_table [<value>]] <- disassemble xref table
[-parameter_list [<value>]] <- disassemble parameter list
[-binary] <- include binary representation
[-source] <- include source lines

Parameter:

item: path name or SID of the item to be decoded.

Options:

If no option is given the decoder will decode all components present in the item.

-environment <value> ... Selects the database environment. See "Database Environment Specification" at the end of this file.

-code [<value> ...] Decode the I-code.

-symbol_table [<value> ...] Decode the symbol table.

-debug_table [<value> ...] Decode the debug table.

-xref_table [<value> ...] Decode the cross-reference table for the body.

-spec_xref_table [<value> ...] Decode the cross-reference table for the specification.

-binary Inserts hex dump as comments into the assembler file.

-source Inserts UCL source lines as comments into the assembler file.

3. The Assembler

The assembler translates a symbolic (assembler) I-code unit, symbol table, debug table or cross-reference table into its binary equivalent and stores it in the database. The assembler text is expected in a file with a specific suffix:

- I-Code assembler file: '.ass'
- symbol table assembler file: '_.tab'
- debug table assembler file: '.tab'
- XREF table (body): '.xref'
- XREF table (specification): '.xref'

Assemble <- generate binaries from symbolic text
<pathname> <- MDB item pathname
[-environment <value>...="$MDA_ENVIRONMENT"] <- MDA environment spec.
[-code [<value>]] <- assemble I-code assembler
[-symbol_table [<value>]] <- assemble symbol table
[-debug_table [<value>]] <- assemble debug table
[-spec_xref_table [<value>]] <- assemble spec xref table
[-xref_table [<value>]] <- assemble xref table

Parameter:

pathname: path name of the item to be assembled.

Options:

If no option is given the assembler will assemble all components for which it finds a corresponding assebler file.

-environment <value> ... Selects the database environment. See "Database Environment Specification" at the end of this file.

-code [<value> ...] Assemble I-Code.

-symbol_table [<value> ...] Assemble a symbol table.

-debug_table [<value> ...] Assemble a debug table.

-xref_table [<value> ...] Assemble a cross-reference table of a body.

-spec_xref_table [<value> ...] Assemble a cross-reference table of a specification.

4. The Debugger

This tool is a low-level I-Code debugger.

Debug <- low-level I-code debugger
[-environment <value>...="$MDA_ENVIRONMENT"] <- MDB environment spec.
[-help] <- list debug commands

Parameter:

pathname: path name of the item to be debugged.

Options:

-environment <value> ... Selects the database environment. See "Database Environment Specification" at the end of this file.

-help Shows the debug options.

Input commands

AL address, value [, ...]	ASSIGN_LOCAL
AG address, value [, ...]	ASSIGN_GLOBAL
AE module, address, value [, ...]	ASSIGN_EXTERNAL
AM address, value [, ...]	ASSIGN_MEMORY
AB address, offset, value [, ...]	ASSIGN_BYTE
AR end item, subitem, value	ASSIGN_REMOTE
B	BREAK
BV address, offset	BYTE_VALUE
E	EXECUTE
ES	EXECUTE single step
EV module, address, [size | identifier | {size, identifier}]	EXTERNAL_VALUE
GV address, [size | identifier | {size, identifier}]	GLOBAL_VALUE
LA destination, ap, [parameter, parameter [, ...]]	LOAD_AP
LV address, [size | identifier | {size, identifier}]	LOCAL_VALUE
MV address, [size | identifier | {size, identifier}]	MEMORY_VALUE
RA	RELOAD_AP
RV end item, subitem	REMOTE_VALUE
SBI instruction [, ...]	SET_BREAK_INSTRUCTIONS
SBP module, position [, ...]	SET_BREAK_POSITIONS
SV size	STACK_VALUE
UA	UNLOAD_AP
H [command]	HELP
X	EXIT

5. The Ada2TPS Tool

The Ada2TPS tool converts an Ada source file to a TPS ASCII file.

ada2tps	<- convert Ada source file to TPS ASCII file
<input_file>	<- name of Ada source file
[-output_file <value>]	<- name of TPS file
[-ada_component <value>]	<- name of TPS Ada component
[-space_component <value>]	<- name of TPS space component
[-tab [<value>]]	<- replace <value> spaces with tab
[-untab [<value>]]	<- replace tabs in source file with spaces
[-language <value>]	<- language version in source file (Ada83 or Ada95)
[-identifiers [<value>]]	<- convert identifiers to Ada83 or Ada95 style
[-no_bold_keywords]	<- no conversion of keywords to bold + lower case

Parameter:

input_file:	file name of the Ada source file.

Options:

-output_file <value>	Filename of TPS ASCII file (Default: <input_file>.doc).
-**ada_component <value>**	Name of the TPS Ada component (Default: Ada).
-**space_component <value>**	Name of the TPS space component (Default: space).This is an empty component for vertical spacing.
-tab [<value>]	Replace <value> spaces with one tab (Default: 2).
-untab [<value>]	Tabs in source file are filled up with spaces up to the next tab position. <value> gives the tab width in spaces (Default: 8).
-language <value>	Language (Ada83 or Ada95) used in Ada source file (Default: Ada83)
-identifiers [<value>]	Ada83 Converts identifiers to upper case. Ada95 Converts identifiers to mixed case. (first character of each word capitalized) If no <value> is given, ada2tps uses the matching style in dependence of the language.
-no_bold_keywords	Without this option, the keywords are converted to lower case and bold style.

6. The UCL2TPS Tool

The UCL2TPS tool converts a UCL source file to a TPS ASCII file.

ucl2tps	<- convert UCL source file to TPS ASCII file
<input_file>	<- name of UCL source file
[-ucl_component <value>]	<- name of TPS UCL component
[-space_component <value>]	<- name of TPS space component
[-tab [<value>]]	<- replace <value> spaces with tab
[-untab [<value>]]	<- replace tabs in source file with spaces
[-language <value>]	<- language version in source file UCL, HLCL or CPL
[-identifiers [<value>]]	<- convert identifiers to UCL, HLCL, CPL reference manual style
[-no_bold_keywords]	<- no conversion of keywords to bold + lower case

Parameter:

input_file:	file name of the UCL source file.

Options:

-output_file <value>	File name of TPS ASCII file (Default: input_file.doc).
-ucl_component <value>	Name of the TPS UCL component (Default: UCL).
-space_component <value>	Name of the TPS space component (Default: space).This is an empty component for vertical spacing.
-tab [<value>]	Replace <value> spaces with one tab (Default: 2).
-untab [<value>]	Tabs in source file are replaced with spaces up to the next tab position. <value> gives the tab width in spaces (Default: 8).
-language <value>	Language (UCL, HLCL or CPL) used in source file (Default: UCL)
-identifiers [<value>]	Converts identifiers to the style used in the UCL Reference Manual: Predefined identifiers in all upper case, user defined identifiers in mixed case (first character of each word capitalized).
-no_bold_keywords	Without this option, keywords are converted to lower-case and bold style.

7. The FORMAT Tool

The FORMAT tool reformats the source code of UCL compilation units. The reformatted code may be stored back in the database or just listed on standard output.

Format <- format CLS compilation units
<item> <- MDB item pathname or SID
[-environment <value>...="$MDA_ENVIRONMENT"] <- MDA environment spec.
[-specification] <- format spec only if library is selected
[-body] <- format body only if library is selected
[-output] <- don't store, just output on screen
[-keywords <value>] <- keywords in lower/upper/mixed_case
[-identifiers <value>] <- identifiers in lower/upper/mixed_case
[-predefined_identifiers <value>] <- predef. ident. in lower/upper/mixed_case
[-named_parameters [<value>...]] <- actual parameters in named notatation (for procedures, functions, pathnames)

Parameter:

item: path name or SID of the item to be formatted.

Options:

If no option is given, format will not change the source code at all. All changes have to be explicitly requested via options.

-environment <value> ... Selects the database environment. See "Database Environment Specification" at the end of this file.

-specification For libraries, format only the specification.

-body For libraries, format only the body.

-output Don't store the formatted source code back in the database, but list it on standard output.

-keywords <value> Change UCL keywords to lower case, upper case or mixed case. <value> is one of "lower_case", "upper_case" or "mixed_case".

-identifiers <value> Change identifiers to lower case, upper case or mixed case. <value> is one of "lower_case", "upper_case" or "mixed_case".

-predefined_identifiers <value> Change predefined UCL identifiers to lower case, upper case or mixed case. <value> is one of "lower_case", "upper_case" or "mixed_case".

-named_parameters [<value>...] Rewrite procedure, function and pathname parameters in named notation. <value> selects the class if entities whose parameters are to be modified, it is one of "procedures", "functions" or "pathnames". More than one value may be given. If the value is omitted, the parameters of all entities will be rewritten.

8. Database Environment Specification

All tools that access the database require an option -environment to specify the database environment. The following describes the syntax of the environment specification. Optionally, the environment variable MDA_ENVIRONMENT may be set to the environment string, the -environment option can then be omitted.

-environment <value> ...

Selects the database environment. Syntax:
CCU <element_configuration> <mission_name> <system_tree_version> <ccu_pathname> <ccu_name> <ccu_version.ccu_issue.ccu_revision>

or

CDU <element_configuration> <mission_name> <system_tree_version> <cdu_pathname> <ccu_version.ccu_issue.ccu_revision> <cdu_test_version> <cdu_instance>)

Example:

```
toolname -env 'CCU APM SOME_MISSION 1 \APM\SOME\PATH TEST 1.0.0' ...
```

or

```
setenv MDA_ENVIRONMENT 'CCU APM SOME_MISSION 1 \APM\SOME\PATH TEST 1.0.0'
toolname ...
```