To: Distribution

From: Steve Herbst

Subject: New value interfaces

Date: 010/23/80

This MTB proposes a set of command, active function, and subroutine interfaces based on the existing value command. They allow users at command level, exec_com's, and programs to reference value segments containing name-value pairs. The names are character strings, and the values can be any data type, converted to any other.

The value active function itself, with short name val, is used to return the value of a name. Commands to define and list associations, and to switch data bases, have longer names such as value_set (vs), rather than the currently used entrypoint names (value$set). The major additions to the existing value facility are:

1. The ability to maintain perprocess associations that are not stored in any value segment and disappear when the process terminates. This feature allows a value segment to be used as a nonwriteable template containing initial values, and read by the user's process to define its own values: value_set -perprocess [value foo -permanent]

   In default mode, [value] returns the perprocess value of a name if one exists; otherwise, it returns the value stored in the user's default value segment, or a specified value segment. The explicit reading and setting of private values is determined by use of the -perprocess (-pp) control argument to the various value commands, its opposite being the default -permanent (-perm).

2. A way of specifying a default value (-default) to be returned when no value is defined.

3. A way of automatically calling an active function (-call) to obtain a value to return when no value is defined for a variable.

4. An update feature (-update, -ud) causes [value_set -ud] to return the previous value of an association, so that the caller can push and pop values. The default is for

[value_set] to return the value that it sets.

I think that having previous values automatically remembered by a -push and -pop feature is unreliable because an exec_com A that calls exec_com B cannot know how many times B has pushed or popped the stack of values. The proposed interface requires each caller to maintain its own stack of values.

5. A way of causing value_set to increment or decrement the current value by an integer, assuming that the value is the character string representation of an integer.

6. A command to delete associations, value_delete, allows the user to specify that a name has no defined value. Null string is allowed as a defined value.

7. The addition of -match and -exclude to value_set, value_delete, and value_list allows the user to specify classes of variable names. The implementation of these control arguments follows the answer command: each takes a string to be searched for in the variable name; if the string is surrounded by slashes, it is a qedx regular expression to match variable names. For example, the command line "value_set -match /_count$/ 0" sets all counts to zero.

8. A value_set_lock (vsl) command/active function to set the value of a name by calling set_lock_$lock with the current process id rather than by simply copying a value. The active function returns true if the lock was locked within a specified wait time. This locking feature can be used by exec_com's to share databases.

9. A set of subroutine interfaces making all the commands' capabilities available to programs, and additionally handling various data types. The value_$set and value_$get entrypoints are called as options (variable) and convert between the data type of the value argument and the internal character string representation. Two more entrypoints value_$set_data and value_$get_data accept pointers and lengths of uninterpreted regions of storage, allowing programs to store structure data as variable values.

   The specialized entrypoints named value_$test_and_set and value_$test_and_set_data set values only if the old value, or the first N words of the old structure value (for example, a version number), match a second input value. These calls are used by a program to make sure that another program has not been using the same name for its own associations.

010/23/80 value, val

Syntax as an active function: [val name {-control_args}]

Syntax as a command: val name {-control_args}

Function: returns the character string value of a name, as set by the value_set (vs) command. If the name has no value and -default or -call is not specified, an error occurs. Values, except for perprocess values (-perprocess), are stored in a value segment with suffix "value" (see "Notes on value segment" below).

Arguments:

name
is a character string. It can -name STR to specify a name beginning with a minus sign, to distinguish it from a control argument.

Control arguments:

-default STR, -df STR
specifies a default value to be returned if none is set. The character string STR must be quoted if it contains blanks or other special characters. A null string is returned if STR is "". If this control argument is not specified and no value exists, an error occurs.

-pathname PATH, -pn PATH
specifies a value segment other than the current default one, without changing the default. See "Notes on value segment" below.

-permanent, -perm
does not look for a perprocess value. The default is to return the perprocess value if one exists, otherwise return the value stored in the value segment. If none exists, an error occurs.

-perprocess, -pp
looks only for a perprocess value, not for one stored in any value segment. If a perprocess value is not found, an error occurs. This control argument is incompatible with -pathname.

-call AF_STR
if no value is set for name, the active string AF_STR is expanded and the value of name is set to be the string's return value. Surrounding brackets must be omitted from AF_STR and the string must be quoted if it contains blanks or other special characters, for example "query What tape?" or "value last_date". If -perprocess is also specified, the value set by -call is a perprocess one. Otherwise, it goes into the value segment.

Access required: r on the value segment, except for perprocess values. Also, w is required to set a value by -call.

Notes:
Perprocess values are stored in a temporary value segment in the process directory, and disappear when the process terminates.

By default, both "value name" and "value name -pn PATH" return the perprocess value of name if there is one, otherwise the value stored in the appropriate value segment. By contrast, "value -pp" returns only the perprocess value, and "value -perm" returns only the one in the value segment.

See the related command/active functions value_defined (vdf), value_set (vs), value_delete (vdl), value_list (vls), value_set_path (vsp), and value_path (vp).

Notes on value segment:
The value segment searched is either the one specified by -pathname or the current default value segment. The default segment is initially:

    [home_dir]>[user name].value

but can be changed by means of the value_set_path (vsp) command and listed by the value_path command/active function. Use of the -pathname control argument does not change the default segment.

010/23/80 value_set, vs

Syntax as an active function: [vs {name} {value_string} {-control_args}]

Syntax as a command: vs {name} {value_string} {-control_args}

Function: associates a character string name with a character string value. The value replaces any previous value for name. If -perprocess is specified or the old value is a perprocess one, the value set is perprocess (see "Notes" below). Otherwise, the association is stored in a value segment (see "Notes on value segment" below).

Arguments:

name
is a character string. It can be -name STR to specify a name beginning with a minus sign, to distinguish it from a control argument. There is no restriction on the length of the name.

value_string
is a character string value, quoted if it contains blanks or other special characters. It can be -value STR to specify a value STR that begins with a minus sign, to distinguish it from a control argument. There is no restriction on the length of the value.

Control arguments:

-add N
adds N to the integer value of each name selected by the other control arguments. If any of the names has no value or has a value that is not the character string representation of an integer, an error occurs. The value of N is allowed to be negative or zero, as can be the resulting value.

-exclude STR, -ex STR
changes all existing associations except those for names that match STR. The character string STR is searched for in names; if it is surrounded by slashes (/), it is interpreted as a qedx regular expression to match names. Only perprocess associations are changed if -perprocess is specified, only permanent ones if -permanent is specified, and both are changed by default. The -exclude control argument is incompatible with the name argument, but can appear multiple times and in combination with -match (see "Notes" below). Neither -match not -exclude is allowed for the active function.

-if VALUE_STR
sets the value value_string only if an old value exists and is equal to VALUE_STR, otherwise returns an error. If -match and/or -exclude are also specified, all selected names with current values equal to VALUE_STR are set to value_string.

-match STR
changes all existing associations for names that match STR. The character string STR is searched for in names; if it is surrounded by slashes (/), it is interpreted as a qedx regular expression to match names. Only perprocess associations are changed if -perprocess

is specified, only permanent ones if -permanent is specified, and both are changed by default. The -match control argument is incompatible with the name argument, but can appear multiple times and in combination with -exclude (see "Notes" below). Neither -match nor -exclude is allowed for the active function.

-pathname PATH, -pn PATH
specifies a value segment other than the current default one, without changing the default. See "Notes on value segment" below.

-permanent, -perm
sets a value in the value segment, regardless of whether the old value if any is perprocess or permanent. The default is to change the perprocess value if one exists, otherwise to change the permanent value if one exists, otherwise to set a permanent value.

-perprocess, -pp
sets a perprocess value, regardless of whether the old value if any is perprocess or permanent. The default is to change the perprocess value if one exists, otherwise to change the permanent value if one exists, otherwise to set a permanent value.

-subtract N, -sub N
subtracts N from the integer value of each name selected by the other control arguments. If any of the names has no value or has a value that is not the character string representation of an integer, an error occurs.

-update, -ud
causes the value_set active function to return the previous value or null string if there was no previous value. The default is to return the value that is set.

Access required: rw on the value segment, except for perprocess values. Lack of write access results in a warning message:

```
value_set: No write permission on PATH.
Perprocess value set for NAME.
```

Notes:
Either name, -match, or -exclude must be specified.

Either value_string or -value STR must be specified.

Perprocess values are stored in a temporary value segment in the process directory, and disappear when the process terminates.

When a value is set in a value segment that does not exist, the user is asked whether to create the segment. The user's default value segment [hd]>[user name].value is created automatically and a message is printed.

The -match and -exclude control arguments are applied in the order specified. Successive -match arguments add to the set of names processed (union) and successive -exclude arguments narrow down the set (intersection). For example, assume the defined variables to be:

```
rs_seg_length, rs_area_length, rs_str_len, arg_str_len, buf_size
```

The command line:

```
     vs 0 -match /_len/ -exclude /_length/ -match /seg_length/
```

operates as follows:
The first -match /_len/ causes the set of selected names to be:

```
     rs_seg_length, rs_area_length, rs_str_len, arg_str_len
```

The following -exclude /_length/ produces the intersection of this set with the set of names NOT matching /_length/:

```
     rs_str_len, arg_str_len
```

The following -match /seg_length/ produces the union of this set with the set of names matching /_seg_length/:

```
     rs_str_len, arg_str_len, rs_seg_length
```

Finally, the value of each of these selected variables is set to 0.

See the related command/active functions value (val), value_defined (vdf), value_delete (vdl), value_list (vls), value_set_path (vsp), and value_path (vp).

Notes on value segment:
The value segment searched is either the one specified by -pathname or the current default value segment. The default segment is initially:

```
     [home_dir]>[user name].value
```

but can be changed by means of the value_set_path (vsp) command and listed by the value_path (vp) command/active function. Use of the -pathname control argument does not change the default segment.

010/23/80 value_set_lock, vsl

Syntax as an active function: [vsl name {-control_args}]

Syntax as a command: vsl name {-control_args}

Function: sets the value of a name by calling set_lock_$lock, thereby testing whether it is already locked (whether a value is defined and was set by value_set_lock). It returns false if the existing value was locked either by the calling process or by another valid (running) process, otherwise it returns true and sets a value. If no value for name exists or the value was not set by value_set_lock, a new value is set and value_set_lock returns true.

Arguments:

name
is a character string. It can be -name STR to specify a name beginning with a minus sign, to distinguish it from a control argument.

Control arguments:

-pathname PATH, -pn PATH
specifies a value segment other than the current default one, without changing the default. See "Notes on value segment" below.

-permanent, -perm
sets a lock value in the value segment. (Default)

-perprocess, -pp
sets a perprocess lock value. The default is -permanent.

-wait_time N, -wtm N
specifies the number of seconds N to wait for the lock to become unlocked (have an undefined value) if it is currently locked. The default is 10 seconds. If N is greater than 60 seconds, the value_set_lock command prints a message after waiting 60 seconds.

Access required: rw on the value segment, except for perprocess values.

Notes on value segment:

The value segment searched is either the one specified by -pathname or the current default value segment. The default segment is initially:

```
[home_dir]>[user name].value
```

but can be changed by means of the value_set_path (vsp) command and listed by the value_path (vp) command/active function. Use of the -pathname control argument does not change the default segment.

```
010/23/80  value_delete, vdl

Syntax as a command:  vdl {name} {-control_args}


Function:  causes one or more names not to have defined values, as
set by value_set and "value -call". Both perprocess values and those
stored in a value seg are deleted, unless -perprocess or -permanent is
specified.


Arguments:
name
   is a character string. It can be -name STR to specify a name
   beginning with a minus sign, to distinguish it from a control
   argument.


Control arguments:
-all, -a
   deletes all defined perprocess and permanent values. If -perprocess
   is specified, only perprocess values are deleted. If -permanent is
   specified, only values stored in the value segment are deleted.
   The -all control argument is incompatible with -match and -exclude
   and with the name argument.
-exclude STR, -ex STR
   deletes all existing values except those for names that match STR.
   The character string STR is searched for in names; if STR is
   surrounded by slashes (/), it is interpreted as a qedx regular
   expression to match names. Only perprocess values are deleted if
   -perprocess is specified, only permanent ones if -permanent is
   specified, and both are deleted by default. The -exclude control
   argument is incompatible with -all and with the name argument, but
   can appear multiple times and in combination with -match (see
   "Notes" below).
-match STR
   deletes all existing values for names that match STR. The character
   string STR is searched for in names; if it is surrounded by slashes
   (/), it is interpreted as a qedx regular expression to match names.
   Only perprocess values are deleted if -perprocess is specified, only
   permanent ones if -permanent is specified, and both are deleted by
   default. The -match control argument is incompatible with -all and
   with the name argument, but can appear multiple times and in
   combination with -exclude (see "Notes" below).
-pathname PATH, -pn PATH
   specifies a value segment other than the current default one,
   without changing the default. For more information, type:
      help value -section "Notes on value segment"
-permanent, -perm
   deletes only values stored in the value segment. The default is to
   delete the perprocess value if one exists, otherwise to delete any
   permanent value.
```

-perprocess, -pp
   deletes only perprocess values. The default is to delete the perprocess value if one exists, otherwise to delete any permanent value.

Access required: rw on the value segment, except for perprocess values.

Notes:
Either name, -all, -match, or -exclude must be specified.

The -match and -exclude control arguments are applied in the order specified. Successive -match arguments add to the set of names processed (union) and successive -exclude arguments narrow down the set (intersection). For example, assume the defined variables to be:
     rs_seg_length, rs_area_length, rs_str_len, arg_str_len, buf_size
The command line:
     vdl -match /_len/ -exclude /_length/ -match /seg_length/
operates as follows:
The first -match /_len/ causes the set of selected names to be:
     rs_seg_length, rs_area_length, rs_str_len, arg_str_len
The following -exclude /_length/ produces the intersection of this set with the set of names NOT matching /_length/:
     rs_str_len, arg_str_len
The following -match /seg_length/ produces the union of this set with the set of names matching /_seg_length/:
     rs_str_len, arg_str_len, rs_seg_length
Finally, the value of each of these selected variables is deleted.

See the related command/active functions value (val), value_defined (vdf), value_set (vs), value_list (vls), value_set_path (vsp), and value_path (vp).

```
C10/23/80  value_defined, vdf

Syntax as an active function:  [vdf name {-control_args}]

Syntax as a command:  vdf name {-control_args}


Function:  returns true if name has a value set by the value_set (vs)
command or by "value -call", false otherwise. The value can be
perprocess or reside in a value segment (type "help value").


Arguments:
name
   is a character string. It can be -name STR to specify a name
   beginning with a minus sign, to distinguish it from a control
   argument.


Control arguments:
-pathname PATH, -pn PATH
   specifies a value segment other than the current default one,
   without changing the default. For more information, type:
      help value -section "Notes on value segment"
-permanent, -perm
   returns true only if a value is defined in the value segment,
   regardless of whether a perprocess value exists. The default is to
   return true for either a perprocess or a permanent value.
-perprocess, -pp
   returns true only if a perprocess value is defined.


Access required:  r to the value segment, except for perprocess values.
Lack of r access is equivalent to no value defined in the segment.


Notes:
See the related command/active functions value (val), value_set (vs),
value_delete (vdl), value_list (vls), value_set_path (vsp), and
value_path (vp).
```

```
010/23/80  value_list, vls

Syntax as an active function:  [vls {name} {-control_args}]

Syntax as a command:  vls {name} {-control_arg}


Function:  lists one or more name-value pairs as set by value_set and
"value -call".


Arguments:
name
   is a character string. It can be -name STR to specify a name STR
   beginning with a minus sign, to distinguish it from a
   control argument.


Control arguments:
-all, -a
   lists all defined values. Only perprocess values are listed if
   -perprocess is specified, only permanent ones if -permanent is
   specified, and both are listed by default. The -all control
   argument is incompatible with -match and -exclude and with the name
   argument.
-exclude STR, -ex STR
   lists all values except those for names that match STR. The
   character string STR is searched for in names; if it is surrounded
   by slashes (/), it is interpreted as a qedx regular expression to
   match names. Only perprocess values are listed if -perprocess is
   specified, only permanent ones if -permanent is specified, and both
   are listed by default. The -exclude control argument is incompatible
   with -all and with the name argument, but can appear multiple times
   and in combination with -match (see "Notes" below).
-match STR
   lists all values for names that match STR. The character string STR
   is searched for in names; if it is surrounded by slashes (/), it is
   interpreted as a qedx regular expression to match names. Only
   perprocess values are listed if -perprocess is specified, only
   permanent ones if -permanent is specified, and both are listed by
   default. The -match control argument is incompatible with -all and
   with the name argument, but can appear multiple times and in
   combination with -exclude (see "Notes" below).
-pathname PATH, -pn PATH
   specifies a value segment other than the current default one,
   without changing the default. For more information, type:
      help value -section "Notes on value segment"
   Multiple -pn arguments are allowed to list values in more than one
   value segment.
-permanent, -perm
   lists only values stored in the value segment. The default is to
   list both permanent and perprocess values.
```

-perprocess, -pp
lists only perprocess values. The default is to list both perprocess values and those stored in the value segment.

Access required: r on the value segment, except for perprocess values.

Notes:
Either name, -all, -match, or -exclude must be specified.

The list is sorted alphabetically by name.

The value_list command by default lists perprocess and permanent values interspersed, the perprocess names preceded by "PP ".

The value_list active function returns the selected names separated by spaces, and no values.

The -match and -exclude control arguments are applied in the order specified. Successive -match arguments add to the set of names processed (union) and successive -exclude arguments narrow down the set (intersection). For example, assume the defined variables to be:

```
rs_seg_length, rs_area_length, rs_str_len, arg_str_len, buf_size
```

The command line:

```
vls -match /_len/ -exclude /_length/ -match /seg_length/
```

operates as follows:
The first -match /_len/ causes the set of selected names to be:

```
rs_seg_length, rs_area_length, rs_str_len, arg_str_len
```

The following -exclude /_length/ produces the intersection of this set with the set of names NOT matching /_length/:

```
rs_str_len, arg_str_len
```

The following -match /seg_length/ produces the union of this set with the set of names matching /_seg_length/:

```
rs_str_len, arg_str_len, rs_seg_length
```

Finally, the value of each of these selected variables is listed.

See the related command/active functions value (val), value_defined (vdf), value_set (vs), value_delete (vdl), value_set_path (vsp), and value_path (vp).

010/23/80 value_set_path, vsp

Syntax: vsp path {-control_arg}

Function: sets the default value segment used by the value commands without -pathname.

Arguments:
path
is the pathname of a value segment or a nonexistent segment, which is created. The value suffix is assumed.

Control arguments:
-brief, -bf
suppresses the warning printed when the user lacks write access to the value segment.

Access required:
At least r access to the value segment is required, and rw is preferred. If the user lacks r access, the default path is not changed and an error message is printed. If the user lacks rw, the default path is changed, but a warning is printed. The -brief control argument can be used to suppress this warning.

Notes:
The default value segment in a process is initially:
[home_dir]>[user name].value

010/23/80 value_path, vp

Syntax as an active function: [vp]

Syntax as a command: vp

Function: returns the pathname of the current default value segment used by the value commands without -pathname.

```
010/23/80  value_

Function: reads and maintains value segments containing name-value
pairs.


Entry points in value_:


:Entry: get: 010/23/80  value_$get

Syntax:
dcl value_$get entry options (variable);

call value_$get (seg_ptr, switches, name, value_arg, code);


Function: returns the defined value of a name.


Arguments:
seg_ptr
   is a pointer to the base of a value segment. To initialize a new
   value segment, create a segment with suffix "value" and call
   value_$init_seg with a pointer to its base. If seg_ptr is null, the
   default value segment is used, which is initially:
     [home_dir]>[user name].value
   but can be changed by value_$set_path or the value_set_path (vsp)
   command. (Input)
switches
   is a bit (36) word of switches: (Input)
     perprocess
       looks only for a perprocess value, not for one stored in any
       value segment. This switch is incompatible with "permanent".
       The default if both switches are off is to return the
       perprocess value if one exists, otherwise return the value
       stored in the value segment.
     permanent
       looks only for a value stored in the value segment.
name
   is a fixed-length or varying character string. If fixed-length,
   trailing blanks are trimmed. There must be at least one character.
   (Input)
value_arg
   is the returned value, having any data type. If conversion from the
   internal character string representation cannot be performed,
   error_table_$bad_conversion is returned. Conversion errors cannot
   occur if value_arg is a character string, but if it has a
   maxlength > 0, the error code error_table_$smallarg is returned if
   truncation occurs. (Output)
code
   is a standard error code. It is error_table_$oldnamerr ("Name not
```

```
53      found.") if no value is defined. (Output)
54
55
56    Access required:  r access to the value segment, except for perprocess
57    values.
58
59
60    Notes:
61    Perprocess values are stored in a temporary value segment in the
62    process directory, and disappear when the process terminates.
63
64
65    :Entry: get_data: 010/23/80  value_$get_data
66
67    Syntax:
68    dcl value_$get_data entry (ptr, bit (36), char (*),
69         ptr, ptr, fixed bin (18), fixed bin (18), fixed bin (35));
70
71    call value_$get_data (seg_ptr, switches, name,
72         area_ptr, buffer_ptr, buffer_size, data_size, code);
73
74
75    Function: returns, into a caller-supplied buffer, the region of storage
76    that is defined as the value of a name, as set by either
77    value_$set_data or value_$test_and_set_data.
 8
79
80    Arguments:
81    seg_ptr
82       is a pointer to the base of a value segment. To initialize a new
83       value segment, create a segment with suffix "value" and call
84       value_$init_seg with a pointer to its base. If seg_ptr is null, the
85       default value segment is used, which is initially:
86         [home_dir]>[user name].value
87       but can be changed by value_$set_path or the value_set_path (vsp)
88       command. (Input)
89    switches
90       is a bit (36) word of switches: (Input)
91         perprocess
92           looks only for a perprocess value, not for one stored in any
93           value segment. This switch is incompatible with "permanent".
94           The default if both switches are off is to return the
95           perprocess value if one exists, otherwise return the value
96           stored in the value segment.
97         permanent
98           looks only for a value stored in the value segment.
99    name
100      is a character string with at least one nonblank character.
101      Trailing blanks are trimmed. (Input)
102   area_ptr
103      if nonnull, points to an area in which the value can be allocated.
 4       If null, buffer_ptr and buffer_size are used. (Input)
```

```
105  buffer_ptr
106     if area_ptr is null, points to a region of storage into which the
107     value can be copied. (Input)
108  buffer_size
109     is the number of words in the buffer pointed to by buffer_ptr.
110     (Input)
111  data_size
112     is the number of words in the value. If it is greater than
113     buffer_size, only buffer_size words are returned, data_size is set
114     to the full size of the value, and error_table_$smallarg is
115     returned. (Output)
116  code
117     is a standard error code. It is error_table_$oldnamerr ("Name not
118     found.") if no value is defined. (Output)
119
120
121  Access required:  r on the value segment, except for perprocess
122  values.
123
124
125  Notes:
126  Perprocess values are stored in a temporary value segment in the
127  process directory, and disappear when the process terminates.
128
129
130  :Entry: set: 010/23/80  value_$set
131
132  Syntax:
133  dcl value_$set entry options (variable);
134
135  call value_$set (seg_ptr, switches, name, new_value, old_value, code);
136
137
138  Function: defines a value for a name, readable by value_$get.
139
140
141  Arguments:
142  seg_ptr
143     is a pointer to the base of a value segment. To initialize a new
144     value segment, create a segment with suffix "value" and call
145     value_$init_seg with a pointer to its base. If seg_ptr is null, the
146     default value segment is used, which is initially:
147       [home_dir]>[user name].value
148     but can be changed by value_$set_path or the value_set_path (vsp)
149     command. (Input)
150  switches
151     is a bit (36) word of switches: (Input)
152       perprocess
153         sets a perprocess value. This switch is incompatible with
154         "permanent". The default if both switches are off is to set
155         a perprocess value if one already exists, otherwise to set
156         a value in the value segment.
```

```
157       permanent
158         sets a value in the value segment.
159  name
160     is a fixed-length or varying character string. If fixed-length,
161     trailing blanks are trimmed. There must be at least one character.
162     (Input)
163  new_value
164     is the value to be set, having any data type. If conversion to the
165     internal character string representation cannot be performed,
166     error_table_$badcall is returned. (Input)
167  old_value
168     is the current value, having any data type. If no value is currently
169     defined, the value of this argument is not changed. If conversion
170     from the internal character string representation cannot be
171     performed, error_table_$bad_conversion is returned. (Output)
172  code
173     is a standard error code. Having no previous value defined does not
174     cause an error code to be returned.
175
176
177  Access required:  rw to the value segment, except for perprocess
178  values.
179
180
181  Notes:
182  Perprocess values are stored in a temporary value segment in the
183  process directory, and disappear when the process terminates.
184
185
186  :Entry: set_data: 010/23/80  value_$set_data
187
188  Syntax:
189  dcl value_$set_data entry (ptr, bit (36), char (*),
190       ptr, fixed bin (18),
191       ptr, ptr, fixed bin (18),
192       ptr, fixed bin (18), fixed bin (35));
193
194  call value_$set_data (seg_ptr, switches, name,
195       new_data_ptr, new_data_size,
196       area_ptr, buffer_ptr, buffer_size,
197       old_data_ptr, old_data_size, code);
198
199
200  Function: defines the value for a name to be a specified number of
201  words of data, readable by value_$get_data.
202
203
204  Arguments:
205  seg_ptr
206     is a pointer to the base of a value segment. To initialize a new
207     value segment, create a segment with suffix "value" and call
208     value_$init_seg with a pointer to its base. If seg_ptr is null, the
```

```
209      default value segment is used, which is initially:
210        [home_dir]>[user name].value
211      but can be changed by value_$set_path or the value_set_path (vsp)
212      command. (Input)
213   switches
214      is a bit (36) word of switches: (Input)
215        perprocess
216          sets a perprocess value. This switch is incompatible with
217          "permanent". The default if both switches are off is to set
218          a perprocess value if one already exists, otherwise to set
219          a value in the value segment.
220        permanent
221          sets a value in the value segment.
222   name
223      is a character string with at least one nonblank character.
224      Trailing blanks are trimmed. (Input)
225   new_data_ptr
226      is a pointer to the value to be set. (Input)
227   new_data_size
228      is the number of words in the value to be set. (Input)
229   area_ptr
230      if nonnull, points to an area in which the old (return) value is
231      to be allocated. If null, buffer_ptr and buffer_size are used.
232      (Input)
233   buffer_ptr
234      if area_ptr is null, points to a region of storage into which the
235      old value can be copied. If both area_ptr and buffer_ptr are null,
236      the old value is not returned. (Input)
237   buffer_size
238      is the number of words in the buffer pointed to by buffer_ptr.
239      If the old value is too large to fit, error_table_$smallarg is
240      returned but old_data_size is correct. (Input)
241   old_data_ptr
242      is a pointer to the old value. (Output)
243   old_data_size
244      is the number of words returned as the old value. (Output)
245   code
246      is a standard status code.  Having no previous value defined does
247      not cause an error code to be returned. (Output)
248
249
250   Access required:  rw on the value segment, except for perprocess
251   values.
252
253
254   Notes:
255   Perprocess values are stored in a temporary value segment in the
256   process directory, and disappear when the process terminates.
257
258
259   :Entry: defined: 010/23/80  value_$defined
260
```

```
Syntax:
dcl value_$defined entry (ptr, bit (36), char (*), fixed bin (35))
     returns (bit (1));

defined_sw = value_$defined (seg_ptr, switches, name, code);


Function: returns "1"b if a value is defined for name, "0"b otherwise.


Arguments:
seg_ptr
   is a pointer to the base of a value segment. To initialize a new
   value segment, create a segment with suffix "value" and call
   value_$init_seg with a pointer to its base. If seg_ptr is null, the
   default value segment is used, which is initially:
     [home_dir]>[user name].value
   but can be changed by value_$set_path or the value_set_path (vsp)
   command. (Input)
switches
   is a bit (36) word of switches: (Input)
     perprocess
       looks only for a perprocess value, not for one stored in any
       value segment. This switch is incompatible with "permanent".
       The default if both switches are off is to return the
       perprocess value if one exists, otherwise return the value
       stored in the value segment.
     permanent
       looks only for a value stored in the value segment.
name
   is a character string with at least one nonblank character.
   Trailing blanks are trimmed. (Input)
code
   is a standard status code. (Output)


Access required:  r on the value segment, except for perprocess values.


:Entry: delete: 010/23/80  value_$delete

Syntax:
dcl value_$delete entry (ptr, bit (36), char (*), fixed bin (35));

call value_$delete (seg_ptr, switches, name, code);


Function: causes there to be no value defined for name.


Arguments:
seg_ptr
```

```
313      is a pointer to the base of a value segment. To initialize a new
314      value segment, create a segment with suffix "value" and call
315      value_$init_seg with a pointer to its base. If seg_ptr is null, the
316      default value segment is used, which is initially:
317        [home_dir]>[user name].value
318      but can be changed by value_$set_path or the value_set_path (vsp)
319      command. (Input)
320   switches
321      is a bit (36) word of switches: (Input)
322        perprocess
323          sets a perprocess value. This switch is incompatible with
324          "permanent". The default if both switches are off is to set
325          a perprocess value if one already exists, otherwise to set
326          a value in the value segment.
327        permanent
328          sets a value in the value segment.
329   name
330      is a character string with at least one nonblank character.
331      Trailing blanks are trimmed. (Input)
332   code
333      is a standard status code. (Output)
334
335
336   Access required:  rw on the value segment, except for perprocess
337   values.
338
339
340   :Entry: set_lock: 010/23/80  value_$set_lock
341
342   Syntax:
343   dcl value_$set_lock entry (ptr, bit (36), char (*), fixed bin,
344        fixed bin (35));
345
346   call value_$set_lock (seg_ptr, switches, name, wait_time,
347        code);
348
349
350   Function: sets the value of a name by calling set_lock_$lock, thereby
351   testing whether it is already locked (whether a value is defined and
352   was set by value_$set_lock). This entry point locks the lock if it is
353   not already locked.
354
355
356   Arguments:
357   seg_ptr
358      is a pointer to the base of a value segment. To initialize a new
359      value segment, create a segment with suffix "value" and call
360      value_$init_seg with a pointer to its base. If seg_ptr is null,
361      the default value segment is used, which is initially:
362        [home_dir]>[user name].value
363      but can be changed by value_$set_path or the value_set_path (vsp)
364      command. (Input)
```

```
365  switches
366     is a bit (36) word of switches: (Input)
367       perprocess
368         sets a perprocess lock value. This switch is incompatible
369         with "permanent". The default is to set a permanent value.
370       permanent
371         sets a lock value in the value segment. This is the default.
372  name
373     is a fixed-length or varying character string. If fixed-length,
374     trailing blanks are trimmed. There must be at least one character.
375     (Input)
376  wait_time
377     is the number of seconds to wait for the lock to become unlocked
378     (have an undefined value) if it is currently locked. After that
379     time has elapsed and the lock is still locked, an error code is
380     returned. (Input)
381  code
382     is a standard status code. It is error_table_$locked_by_this_process
383     if the lock is already locked by the caller's process,
384     error_table_$lock_wait_time_exceeded if the lock is locked by
385     another process, or zero if this entry point was able to lock
386     the lock. (Output)
387
388
389  Access required:  rw on the value segment, except for perprocess.
390
391
392  :Entry: test_and_set: 010/23/80  value_$test_and_set
393
394  Syntax:
395  dcl value_$test_and_set entry options (variable);
396
397  call value_$test_and_set (seg_ptr, switches, name,
398       new_value, old_value, code);
399
400
401  Function: defines a new value for a name, only if the name has a
402  specified current value.
403
404
405  Arguments:
406  seg_ptr
407     is a pointer to the base of a value segment. To initialize a new
408     value segment, create a segment with suffix "value" and call
409     value_$init_seg with a pointer to its base. If seg_ptr is null, the
410     default value segment is used, which is initially:
411       [home_dir]>Person_id.value
412     but can be changed by value_$set_path or the value_set_path (vsp)
413     command. (Input)
414  switches
415     is a bit (36) word of switches: (Input)
416       perprocess
```

```
417         tests and sets a perprocess value. This switch is
418         incompatible with "permanent". The default if both switches
419         are off is to test the perprocess value if one is defined,
420         otherwise to test the value in the value segment. The value
421         set is perprocess or permanent depending on the value tested.
422       permanent
423         tests and sets the value in the value segment.
424  name
425     is a fixed-length or varying character string. If fixed-length,
426     trailing blanks are trimmed. There must be at least one character.
427     (Input)
428  new_value
429     is the value to be set, having any data type. If conversion to the
430     internal character string representation cannot be performed, the
431     error code error_table_$badcall is returned. (Input)
432  old_value
433     is the caller-supplied value that must equal the value currently
434     defined in order for the new value to be set. (Input)
435  code
436     is a standard status code. It is error_table_$action_not_performed
437     if old_value does not match the currently defined value.
438
439
440  Access required:  rw to the value segment, except for perprocess
441  values.
442
443
444  Notes:
445  If the value tested is perprocess, the value set is also perprocess,
446  and vice-versa.
447
448
449  :Entry: test_and_set_data: 010/23/80  value_$test_and_set_data
450
451  Syntax:
452  dcl value_$test_and_set_data entry (ptr, bit (36), char (*),
453       ptr, fixed bin (18),
454       ptr, fixed bin (18), fixed bin (35));
455
456  call value_$test_and_set_data (seg_ptr, switches, name,
457       new_data_ptr, new_data_size,
458       old_data_ptr, old_data_size, code);
459
460
461  Function: defines the value for a name to be a specified number of
462  words of data, readable by value_$get_data, only if the first N words
463  of the name's current value have specified contents.
464
465
466  Arguments:
467  seg_ptr
468     is a pointer to the base of a value segment. To initialize a new
```

```
469      value segment, create a segment with suffix "value" and call
470      value_$init_seg with a pointer to its base. If seg_ptr is null, the
471      default value segment is used, which is initially:
472        [home_dir]>[user name].value
473      but can be changed by value_$set_path or the value_set_path (vsp)
474      command. (Input)
475   switches
476      is a bit (36) word of switches: (Input)
477        perprocess
478          looks only for a perprocess value, not for one stored in any
479          value segment. This switch is incompatible with "permanent".
480          The default if both switches are off is to return the
481          perprocess value if one exists, otherwise return the value
482          stored in the value segment.
483        permanent
484          looks only for a value stored in the value segment.
485   name
486      is a character string with at least one nonblank character.
487      Trailing blanks are trimmed. (Input)
488   new_data_ptr
489      is a pointer to the value to be set. If null, the current value is
490      deleted and no value is defined. (Input)
491   new_data_size
492      is the number of words in the value to be set. (Input)
493   old_data_ptr
494      is a pointer to some data, whose first old_data_size words must
495      equal the first old_data_size words of the name's current value in
496      order for the new value to be set. (Input)
497   old_data_size
498      is the number of words to be compared. This number can be less
499      than the number of words in the name's current value (used, for
500      example, to compare only the header of a structure), but an
501      error code is returned if it is greater. (Input)
502   code
503      is a standard status code. It is error_table_$action_not_performed
504      if the old-value match fails. (Output)
505
506
507   Access required: rw on the value segment, except for perprocess
508   values.
509
510
511   Notes:
512   If the value tested is perprocess, the value set is also perprocess,
513   and vice-versa.
514
515   The value of a name can be conditionally deleted by passing a null
516   new_data_ptr.
517
518
519   :Entry: set_path: 010/23/80  value_$set_path
520
```

```
Syntax:
dcl value_$set_path entry (char (*), bit (1), fixed bin (35));

call value_$set_path entry (path, create_sw, code);


Function: sets the default value segment used by the value commands
with no -pathname argument.


Arguments:
path
   is the pathname. The value suffix is assumed. (Input)
create_sw
   is ON to create a value segment if none exists. (Input)
code
   is a standard status code. (Output)


:Entry: get_path: 010/23/80  value_$get_path

Syntax:
dcl value_$get_path entry (char (*), fixed bin (35));

call value_$get_path (path, code);


Function: returns the pathname of the current default value segment
used by value commands without -pathname.


Arguments:
path
   is the pathname. (Output)
code
   is a standard status code. (Output)


:Entry: init_seg: 010/23/80  value_$init_seg

Syntax:
dcl value_$init_seg entry (ptr, fixed bin, ptr,
     fixed bin (19), fixed bin (35));

call value_$init_seg (seg_ptr, seg_type, remote_area_ptr,
     seg_size, code);


Function: initializes a segment to be a value segment.


Arguments:
```

```
573  seg_ptr
574     is a pointer to a segment. (Input)
575  seg_type
576     determines the type of use to which the value segment will be put,
577     and therefore the method of allocating values: (Input)
578       0 - permanent: shareable by multiple processes and therefore
579           locked when modified, with values always stored in the
580           value segment itself.
581       1 - perprocess: for use only by the calling process and
582           therefore never locked, with values optionally stored
583           in an area outside the "value segment" (see the
584           remote_area_ptr argument below).
585  remote_area_ptr
586     for a perprocess segment only, points to an area outside the value
587     segment in which values are to be allocated. For example, the
588     "value segment" can be a region of storage 72 words long consisting
589     only of a header, and remote_area_ptr can point to system_free_4_.
590     (Input)
591  seg_size
592     is the number of words available to the value segment, or to the
593     remote area if remote_area_ptr is nonnull. If seg_size is 0, the
594     available size is an entire segment. (Input)
595  code
596     is a standard status code. (Output)
597
598
599  Access required:  rw on the segment.
```