To: MTB Distribution

From: Bonnie Braun

Date: 03-01-82

Subject: A Query Interface for the TR Database.

INTRODUCTION

This revised MTB describes the command tr_query, an interface to the TR system. It is the user interface to access specific TR reports given certain selection criteria. With the installation of the new TR system, there is a need to examine the TR database. tr_query replaces the tools "display_tr1" and "display_tr2", which were interim TR DB display tools.

PURPOSE

The use of tr_query allows a wide range of selection criteria for TR data base access. The user can designate the format and content of the displayed TRs.

IMPLEMENTATION

The utility ssu_ is used for tr_query request loop handling. Given a set of selected trouble reports, the user can look at the entire TR record or only certain fields. In addition, the user can format the selected fields via ioa_-like controls. The lrk parser is be used to parse the selection expressions. An edit feature is provided to edit the last selection expression.

SELECTION EXPRESSION CONSTRUCT

Selection expressions are composed of relationships between field names and values. There are two types of field_names, those which select TRs by using keys (TR database is an indexed file), and others (area, summary and text) which select by doing string searches within a given TR field.

DOCUMENTATION

tr_query, trq

SYNTAX AS A COMMAND:

trq {TR_IDS} {-control_args}

FUNCTION:

The tr_query command allows a user to reference Trouble Reports selected from and maintained in the TR database. In the default mode of operation, tr_query enters a request loop where it reads user requests from the terminal until the quit request causes it to exit.

ARGUMENTS:

TR_IDS may be a list: tr_id1 tr_id2 tr_id3... tr_idN

tr_query selects the records specified by TR_IDS. Valid TR_IDS are TR numbers that have the form phxNNNNN, trNNNNN, TRNNNNN or NNNNN. Insignificant digits need not be given. For example, the following TR_IDS all identify the same report: TR891 00891 phx891 phx00891 891.

CONTROL ARGUMENTS:

-abbrev, -ab
enables abbreviation expansion and editing of request lines.

-brief, -bf
supresses the message "N records selected".

-input_file PATH, -if PATH
specifies that the selection expression be obtained from an ASCII file designated by path. A suffix of .trsl is assumed if not specified. The user inputs the selection expression on one or more lines.

-long, -lg
prints the message "N records selected" after a selection occurs. This is the default.

-no_prompt
stops prompting for tr_query requests. The default prompt is "tr_query(N):", where N is the recursion level if greater than one. Equivalent to -prompt "".

-profile PATH, -pf PATH
specifies the pathname of the profile to use for abbreviation expansion. The suffix "profile" is added if necessary. This control argument implies -abbrev.

-prompt STR
changes the prompt for tr_query request lines to STR. If STR is "", the user is not prompted. -prompt accepts "STR^[(^d)^]:" which takes the form STR(N):, where N is the recursion level if greater than one.

-request STR, -rq STR
specifies an initial request line to be executed prior to entering the request loop. For example:

-request "select submitter=Grey; print all; quit"

EXAMPLE: (! precedes the user's input)

Sequence	Explanation
! trq	Invoke tr_query with no arguments and enter the request loop.
tr_query: !select	Issue the select request with no arguments, invoking terminal input mode
Input:	to type in the selection expression.
! to = Wallman	Type in the selection expression which requests all TRs that belong to Wallman.
! .	A "." on a line by itself signifies end of input.
tr_query (select): 37 TR records selected.	Three TRs assigned to Wallman are found in the database. This is called the current subset.
tr_query: ! list 1:3	List the first three TRs in a format showing the the TR number, the area in parentheses and as much of the summary that fits on one line.
1 phx02542 (compose)	is throwing away contents of small include
2 phx02574 (compose)	has error in page breaks with -lf request.
3 phx02591 (compose)	fails to keep a keep block.
tr_query: ! write all my_trs	Writes each entire TR to the file my_trs.trw. The .trw suffix is assumed.

```
tr_query: ! quit        Exit tr_query.
```

REQUEST LINES

The semicolon is used to separate multiple requests on a line. Request lines use () for iteration, "" for quoting, and [] to invoke tr_query active requests.

Any request line which begins with ".." will be passed directly to the Multics command processor with the leading ".." stripped off. This is the recommended method for executing Multics commands from within the tr_query subsystem.

DESCRIPTION OF REQUESTS: (grouped according to function)

1. HELP REQUESTS
   - help
   - list_field_values
   - list_help
   - list_requests
   - ?
   - .

2. The SELECT REQUEST
   - select

3. PRINT REQUESTS
   - list
   - print
   - print_selection_expression

4. COPY REQUESTS
   - append
   - preface
   - save
   - save_selection_expression
   - write

5. EDIT REQUESTS
   - edit
   - edm
   - emacs
   - qedx
   - teco
   - ted
   - set_editor
   - tr_edit

6. ADVANCED REQUESTS
   - set_output
   - do
   - execute
   - ..

7. SUBSYSTEM REQUESTS
   - abbrev
   - answer
   - if
   - quit
   - ready
   - ready_off
   - ready_on

1. HELP REQUESTS

help, h

h topic

Provides information on the various tr_query requests and

other related topics. If a request name is given, then information on that request is provided.

list_field_values, lfv

lfv tr_field1 tr_field2 ...tr_fieldN

Prints the valid field values for each tr field name given. See TR FIELD NAMES below for a list of field names.

list_help, lh

lh {str1 str2 ... strN}

Prints the titles of available help info files known within tr_query. If str1 or more are specified, only those topics containing that str are listed.

list_requests, lr

lr {str1 str2 ... strN} {-control _args}

Prints brief information about tr_query requests which have a documentation string associated with them. When given non-control arguments of str, it lists all requests with matching names (similarly to list_help).

Control arguments:

-all, -a
can be used to list all defined requests including those which have no documentation string.

-exact
can be used to list only exact matches of strN specified.

?
Lists short and long names of trq requests.

.
tr_query identification request. This prints the tr_query version, whether the abbrev processor is on, the number of TRs selected and the current TR referenced.

2. The SELECT REQUEST:

select, sel, sl

sl {selection_expression} {-control_args},

Selects TR reports from the TR database. This allows one to selectively look at TRs according to the selection_expression construct. The selection expression can be expressed on the request line, an input file or in terminal input mode. Control arguments can appear only on the request line.

Terminal input is automatically invoked when no selection expression is specified. Once in input mode, the user types the selection expression on one or more lines. A "." typed on a line by itself terminates input and the selection takes place. A "\f" on a line by itself causes the ted editor to be invoked so the user can edit the expression. When the editor is exited, she is returned to request loop level. A "\q" or "\fq" typed on a line by itself terminates input and returns the user directly to request level without selecting.

The select request deals in subsets and keeps track of two subsets at all times, the current subset and the previous subset. Initially, the first selection selects from the TR database. This creates a subset of the selected TRs and is called the current subset. At this point there is no previous subset. The next selection can select from either the current subset or the TR database. The resulting subset from the second selection becomes the current subset and the first subset is bumped to previous subset status. So, at every successful selection, the current becomes the previous and the new subset becomes the current.

Arguments:

selection_expression

The basic forms are: a = b, a = |{b c d}, a = &{b c d}, a = {b | c | d}, a = {b & c & d}, b < a < c
where "a" is a field name and "b c d" are field values.

Basic forms can be connected by ands (&) and ors (|). For example: a = b & c = d | (a = b & f = g)

Selection expressions are constructed of TR field names, relational operators and field values. Here it helps to be familiar with the format and content of a Trouble Report. For example, below are the first few lines of TR phx06493.

TR:	phx06493	State:	verified
Type:	problem	Stage:	verified
Priority:	normal	Entered:	1980-06-13_09:41:38

TR field names correspond to the headings in a TR. The field values follow the headings: eg. tr = 6493, state = verified, type = problem. All field names are lowercase. A complete list of field names, their short names and descriptions are below under TR FIELD NAMES. Selection expression constructs are discussed further under SELECTION EXPRESSION SYNTAX below.

Control Arguments:

-database, -db
specifies that the selection takes place from the database. This is the default.

-previous, -prev
specifies go back to the previous subset. This is useful when a select produces undesirable results.

-input_file path, -if path
specifies that the subset to be created be obtained from an ascii file containing a selection expression. This segment can be created initially via an editor or by the save_selection_expression request. A suffix of .trsl is assumed if not specified. -if is incompatible with -save, or a selection expression.

-save path, -sv path
specifies that the subset to be selected be obtained from an ascii file containing TR numbers. This segment is initially created via the save request. A suffix of .trsv is assumed if not specified. save is incompatible with -if, or a selection expression.

-process_select, -psl
creates a subset by processing the selection expression previously entered or edited. The default action is as follows: (1) if the control_args -union, -differ or -inter were specified as part of the original select request, the -psl will apply to the current subset, 2) if no control_arg or the -db was specified, the -psl applies to the database.

inter, -differ, -union and -db can be used with -psl to override any of these same control_args if they were part of the original select request.

-no_process_select, -npsl
which specifies that no selection is to be made. Instead, the select request saves the selection expression and returns to request loop level. This will prevent processing for terminal input ending in a "." and selection expressions specified on the select request line.

-brief_errors, -bfe
prints the first error message associated with the first error found in the selection expression and stops.

-long_errors, -lge
prints all error messages associated with all the errors found in the selection expression. This is the default.

-force, -fc
does not first query the user before a string search select of the database (ie. using the tilde "~" operator). The default is to query.

Arguments to manipulate the subset:

-union
selects from the database and adds its results to the current subset creating a larger subset.
-inter
selects from the current subset to create a smaller subset.
-differ
excludes from the current subset.

The above three control_args are used to manipulate the subset. Otherwise, a select will use the database.

-no_union
overrides any -union control previously on the request line.
-no_inter
overrides any -inter control previously on the request line.
-no_differ
overrides any -differ control previously on the request line.
-no_database, -no_db
overrides any -database control previously on the request line.

Select Request Example: (! precedes user input)

```
! trq                        Invoke tr_query at command level.

tr_query:  ! sl type = problem
                             Select all problem TRs from the
                             TR database.

tr_query (select): 4798 TR records selected.
                             This is the current subset. No
                             previous subset exists yet.
```

```
tr_query:  ! sl -inter dten >= 1/1/82

tr_query (select): 112 TR records selected.
                        Select from the current subset of
                        problems, the TRs  entered  on or
                        after Jan. 1, 1982. This is now the
                        new current subset. The previous
                        subset contains 4798 TRs.

tr_query: ! ted         Enter ted editor to edit selection
                        expression.
Entering ted.
! 1                     Print the expression.
dten >= 1/1/82
! s/$/ \c& type = suggestion/p
dten >= 1/1/82 & type = suggestion
                        Edit and print the expression.
! w                     Write the new selection.
! q                     Quit the editor.

tr_query: ! pse         print the selection expression.

dten >= 1/1/82 & type = suggestion
with controls -inter

tr_query:  ! sl -process_select  -union

tr_query (select): 130 TR records selected.
                        Select  from  the  database  the
                        suggestion  TRs entered on or after
                        Jan. 1, 1982 and merge the results
                        to the current subset creating a
                        new current subset of 130 TRs. The
                        previous subset now contains 112
                        TRs.
tr_query: ! pse

dten >= 1/1/82 & type = suggestion
with controls -database -union

tr_query: ! sl -prev

tr_query (select): 112 TR records selected.
                        Go back to the previous  subset of
                        112 TRs. This  becomes the current
                        subset. There is no previous subset.

tr_query: ! pse         print_selection_expression shows the
                        selection expression used to obtain
dten>= 1/1/82           this subset originally.
with controls  -inter
```

tr_query: ! quit Exit tr_query.

3. PRINT REQUESTS:
For a detailed description of specs, see SPECIFIERS below.

list, ls

ls {specs} {-control_args}

Prints a list of the TRs selected showing the TR number, area field (shown in parentheses), and as much of the summary field that will fit on one line. The default is to list all the selected TRs unless specs is specified.

Arguments:

-count, -ct
outputs the number of TRs without listing them.

Active Request: As an active request, [ls] returns the list of TR numbers selected, separated by a space.

print, pr, p

pr {specs} {-control_args}

prints the current (default) entry or the one specified by specs. The default TR format and content is defined by the set_output request.

Control arguments:

-format format_name, -fmt format_name
specifies the content and format of the TR report to be output as defined by format_name. The default is format_name "default". See FORMAT NAMES below for a list. Note this is incompatible with the -field_name and -control control_args.

-field_name tr_field1 tr_field2 ...tr_fieldN,
-fn tr_field1 tr_field2 ...tr_fieldN
specifies the TR fields to be output. The order of the tr_fields is significant. See TR FIELD NAMES below for a list. Note -field_name is incompatible with -fmt and -ctl. The format is one field per line containing the appropriate tr_field header and value. For example: "pr 1 -fn to tr ps" would appear as:

```
To:          Coren Batts
TR:          phx12058
```

Processing Stage: verified

-control ioa_str tr_fields, -ctl ioa_str tr_fields
specifies an output format for given tr_fields using ioa_$nnl constructs. See IOA_ CONSTRUCTS below for more information. Note that -ctl is incompatible with -fmt and -fn.

print_selection_expression, pse

Prints the selection expression for the current subset.

4. COPYING REQUESTS:
For a detailed description of specs, see SPECIFIERS below.

append, app

app {specs} path {-control_args}

Appends the specified TR reports of the current selection subset to the segment as designated by path. The default TR format and content is defined by the set_output request.

path
is the path of the segment for the TR numbers to be appended to. The suffix .trw is added if not already present. If the segment does not already exist, the user will be queried before creating it. The path must be specified before -field_name or -control unless -pathname is used.

Control Arguments:

-force, -fc
does not query the user before creating the segment if it doesn't already exist.

-pathname path, -pn path
specifies the pathname to be used.

-format format_name
-field_name tr_fields
-control ioa_str tr_fields
For a description of the above control arguments, see the print request.

preface, prf

prf {specs} path {-control_args}

Writes the specified TR reports of the current selection subset to the beginning of the designated segment. The default TR format and content is defined by the set_output request.

path
is the path of the segment for the TR numbers to be written to. The suffix .trw is added if not already present. If the segment does not already exist, the user will be queried before creating it. The path must be specified before -field_name or -control unless -pathname is used.

Control Arguments:

-force, -fc
-pathname path, -pn path
For a description of the above control arguments, see the append request above.

-format format_name
-field_name tr_fields
-control ioa_str tr_fields
For a description of the above control arguments, see the print request.

save, sv

sv path {-control_arg}

Writes the TR numbers of the current selection subset to a given segment, overwriting it if the segment already exists.

Arguments:

path
is the pathname of the segment for the TR numbers to be written to. The suffix .trsl is added if not already present. If the segment does not already exist, it will be created automatically.

Control Arguments:

-force, -fc
writes to an existing segment without querying the user first.

save_selection_expression, sse

sse path {-control_arg}

Writes the current selection expression to a given segment, overwriting if the segment already exists.

Arguments:

path
is the pathname of the segment for the select expression to be written to. The suffix .trsl is added if not already present. If the segment does not already exist, it will be created automatically.

Control Arguments:

-force, -fc
writes to an existing segment without querying the user first.

write, w

w {specs} path {-control_args}

Writes the specified TR reports of the current selection subset to the file as designated by path, overwriting if the segment already exists. The default TR format and content is defined by the set_output request.

path
is the path of the segment for the TR numbers to be written to. The suffix .trw is added if not already present. If the segment does not already exist, it will be created automatically. The path must be specified before -field_name or -control unless -pathname is used.

Control Arguments:

-force, -fc
writes to an existing segment without querying the user first.

-pathname path, -pn path
For a description of the -pn control argument, see the append request above.

-format format_name
-field_name tr_fields
-control ioa_str tr_fields
For a description of the above control arguments, see the print request.

5. EDIT REQUESTS:

edit, ed
allows the user to edit the current selection expression by invoking their default editor, as defined in the TR user registration database. For unregistered users, the default editor is qedx. The editor can be changed for this invocation of tr_query using the set_editor request.

set_editor {editor_name}, sed {editor_name}
sets the editor to be used in the edit request to editor_name for the current invocation of tr_query. The list of allowed editors include edm, qedx, ted, teco, emacs and tr_edit.

edm
emacs
qedx
teco
ted
Enters the given editor to edit the current selection expression.

tr_edit
is a user defined abbrev which invokes a private editor used to edit the current selection expression.

6. ADVANCED REQUESTS:

set_output, so

so {-control_args}

Sets the fields to be shown in the print, append, preface and write requests. The default is the TR record in the standard TR display format excluding the history field. The default is reset when set_output is invoked with no arguments.

Control arguments:

-format format_name
-field_name tr_fields
-control ioa_str tr_fields
For a description of the above control arguments, see the print request.

do request_line STR(s)
performs argument substitution in REQUEST_LINE using STR(s) and then executes the request line.

execute, e

e STR

passes the concatenation of STRs with intervening spaces to the Multics command processor. This request is different from ".." because it is first parsed as a tr_query request line. The tr_query request interpreter expands tr_query request functions, strips quotes, and performs iteration before the line is passed on to the command processor. For example, the request:

e string [list] prints the numbers of TRs selected.

The execute request function can be used to invoke a Multics active function from within tr_query. For example:
write [e date]"
writes the current TR report to a file whose path name is the current date, with the suffix trw.

.. command_line
passes the rest of the command line to the command processor, without processing by tr_query. The ".." must be the first two characters of the request line.

7. SUBSYSTEM REQUESTS.

abbrev, ab

ab {-control_arg}

When invoked with no arguments, turns the abbreviation processor on for the rest of the tr_query invocation. If it's already on, it has no effect.

Control arguments:

-off
turns abbrev processing off. A subsequent "ab" turns it on again.

The ab request acts on the user's default abbrev processor unless the -profile control was specified when tr_query was invoked. In this case, ab acts upon the processor specified by -profile.

answer

equivalent to the Multics answer command, provides preset answers to questions asked by another request.

if

equivalent to the Multics if command, it conditionally executes one of two request lines depending on the value of an active string. As an active request, returns one of two character strings to the subsystem request processor depending on the value of an active string.

quit, q
exits the tr_query request loop.

ready, rdy
prints the Multics ready message.

ready_off, rdf
turns off the Multics ready message.

ready_on, rdn
turns on the Multics ready message.

LIST OF REQUESTS: (in alphabetical order)

abbrev, ab
Turns abbrev on in a tr_query invocation.

answer
Provides preset answers to questions asked by another request.

append, app
Appends the specified TRs to a given segment.

do
Perform substitution into a request line before execution.

edit, ed
Enters an editor to edit the current selection expression.

edm
Enters edm to edit the current selection expression.

emacs
Enters emacs to edit the current selection expression.

execute, e
   Execute a Multics command line.

help, h
   Print information on selected help topics.

if
   Conditionally executes one of two request lines depending on the value of an active string.

list, ls
   Lists the current selected TRs.

list_field_values, lfv
   Lists the valid field values for given field names.

list_help, lh
   Print information on available help topics.

list_requests, lr
   List brief information about tr_query requests.

preface, prf
   Prefaces the specified TRs to a given segment.

print, pr, p
   Prints TRs from the current subset as specified.

print_selection_expression, pse
   Prints the selection expression.

qedx
   Enters qedx to edit the current selection expression.

quit, q
   Exits the tr_query request loop.

ready, rdy
   Prints Multics ready message.

ready_off, rdf
   Turns off the Multics ready message.

ready_on, rdn
   Turns on the Multics ready message.

save, sv
   Saves the specified TRs numbers in given a segment.

save_selection_expression, sse
   Saves the current selection expresion in a given segment.

select, sel, sl
Selects TR reports from the TR database.

set_editor, sed
Sets the editor to be invoked by the edit request.

set_output, so
Lets the user set his own default output format

teco
Enters teco to edit the current selection expression.

ted
Enters ted to edit the current selection expression.

tr_edit
Enters the editor specified by tr_edit to edit the selection expression.

write, w
Writes the specified TRs to a given segment.

?
Lists short and long names of trq requests.

.
Prints information about the current state of tr_query.

TR FIELD NAMES:

Field names have two functions: (1) specify the field to be selected upon in a selection expression and (2) to designate the exact fields one wants to see in the output requests using the set_output request or the print, write, append and preface requests and the -field_name control argument.

Field Names in alphabetical order
(with a selection expression example):

area, ar, problem_area
is the area of the system where the TR applies.
example: ar ~ hardcore

ar_sm
references the area and summary fields of the TR. The example: ar_sm ~ tr_query is equivalent to the selection expression: (ar ~ tr_query | sm ~ tr_query) but is much more efficient and faster.

ar_sm_tx
references the area, summary and text fields of the TR. The example: ar_sm_tx ~ tr_query is equivalent to the selection expression: (ar ~ tr_query | sm ~ tr_query | tx ~ tr_query) but is much more efficient and faster.

date_entered, dten
is the date/time when the TR was entered.

date_responded, dtrp
is the date/time when the TR was responded to.

date_resolved, dtro
is the date/time when the TR was resolved.

date_updated, dtu
is the date/time when the last transaction for this TR occured.

date_verified, dtvr
is the date/time when the TR was verified.

Examples of valid Date/Time forms:

Any format that is acceptable to convert_date_to_binary_ such as:
May_23_1980 "April 25, 1980" 05/25/79

"10 June 1978 MST 9:30 am" "08/01/80 1425.0 est Fri"

Also, the form 1980-08-20__14:27:32 is acceptable. When DATE alone is given, TIME defaults to the current time. Note that date/time forms with spaces or commas must be quoted, but underscores can be used instead.

distribution, dist
is a list of person_ids that the tr is to be distributed to. example: dist = Homan

distribution_site, dist_site
is the site names where the problem occured.
example: dist_site = Ford

elapsed_time_resolved, ero
is the time that has passed (in weeks) from the submission of the report to its resolution. example: ero = 5.0

elapsed_time_responded, erp
is the time that has passed (in weeks) from the submission of the report to its response. example: erp = 2.0

elapsed_time_verified, evr
is the time that has passed (in weeks) from the submission of the report to its verification. example: evr = 1.0

error_list_info, eli
is the error list info which contains the name of an existing error list that is associated with the TR and the corresponding number of the error list entry. Note: This is strictly used with output requests to print both the name and number and cannot be used with the select request.

error_list_name, elnm
is the name of an existing error list that is associated with the TR. Note: This can be used with both output requests to print the name and with the select request to select on the error list name. example: elnm = trq

error_list_number, elno
is the error list number associated with the error list entry. Note: This can be used with both output requests to print the number and with the select request to select on the error list number. example: elno = 16

exception, ex
is used internally for TR manipulation.

history, hist
keeps track of who has been assigned to process the TR in the past. Can be used to find resolved TRs for a given person. This is keyed on Person_ids. example: hist = Davidoff

linked_tr, ltr
a list of TR numbers that are related to the given TR. example: ltr = 6554

on_behalf_of, onbehalf, obh
identifies who the TR was entered for. example: obh = "J. Sanders"

origin, orig

identifies the general source of the report: internal (generated by Honeywell personnel) or external (generated by customer sites). example: orig = ex

priority, prty
is the urgency of the problem: critical, high or normal. For question/suggestion reports, this field is not defined. example: prty = high

processing_stage, stage, ps
is the current stage in the processing of the report. example: ps = resolved

release_first_in, rlfi
is the Multics release in which a problem is first known to occur. example: rlfi = MR8.0

release, rl
is the Multics release to which the TR report applies. example: rl = MR9.0

routed_to, to
are the Person_ids of people currently assigned to process the report. The to field applies only to non-resolved TRs. example: to = Greenburg

submitter, sub, from, fm
is the Person_id of the user or Site SA submitting the report. example: sub = GCroon

submitter_site, sub_site, site, from_site, fm_site
the site at which the TR report occured. example: site = CNO

tr_number, tr, number, no
is the number which identifies the report. example: tr=12000

state, st
is the current status of the report. Every state has a corressponding processing stage for each type of TR. example: st = error

summary, sm
is a brief description of the TR problem/question/suggestion. example: sm ~ tolts

test_path, tstp
is the test path(s) that correspond to the TR. example: tstp ~ fortran_bug.audit

text, tx
is where the full explanation of the TR problem/question /suggestion is expressed. example: tx ~ pl1

type, tp
is the kind of trouble report submitted. There are three:
problem, prob, prb, pb, p
suggestion, sugg, sg, s
question, ques, qs, q

Each type of TR has associated states and processing stages. This is explained further in the info file tr_states.info and can be accessed via the help facility at Multics command level or tr_query request level.

SELECTION EXPRESSION SYNTAX:

There are two basic kinds of selection <expression>s as defined by <basic_exp> in the BNF description below. They are <rel_exp>s and <range_exp>s. <range_exp>s are commonly used with date/time and the tr_number field names. The <field_name>s used in a <basic_exp> are described under TR FIELD NAMES above.

Selection Expression Grammar:

Symbols preceeded by an apostrophe (') denote them as part of the grammar and not part of the bnf syntax.

<expression> ::= <exp_2> | <expression> '| <exp_2>

<exp_2>::= <exp_1> | <exp_2> & <exp_1>

<exp_1> ::= <basic_exp> | <not_exp> | <parenthesized_exp>

<not_exp> ::= ^<parenthesized_exp>

<parenthesized_exp> ::= (<expression>)

<basic_exp> ::= <rel_exp> | <range_exp>

<rel_exp>::= <field_name> <rel_op> <value>
| <field_name> <rel_op> <&_list>
| <field_name> <rel_op> <or_list>

<range_exp> ::=
<value> <range_op> <field_name> <range_op> <value>

<range_op> ::= '< | '<= | > | >= | ^'< | ^>

<rel_op> ::= '< | '<= | > | >= | = | ^= | ^'< | ^> | ~ | ^~

<&_list>::= &(<value_list>)

<or_list>::= '|(<value_list>)

<value_list> ::= <value> | <value_list> <value>

<value> ::= <string>

<field_name> ::= <symbol>

Braces {} can be used interchangeably with parentheses (). This is useful when typing the selection expression on the request line because parentheses are interpreted as iteration by the request loop and braces are not.

The <rel_op>s "~" and "=" are defined as follows:

"A ~ BCde" means search field A for the containing string "bcde". Note that this is case insensitive.

"A = BCde" means match the entire field A to the value "BCde". This is not case insensitive.

<string> can be any non-null character string. If <value> contains blanks or quote characters then it must be enclosed in quotes and embedded quotes must be doubled quoted. For example:

```
"January 20, 1979" <= date_updated <= April_1_1979
area ~ "The word ""quote"" is quoted."
summary ~ |("volume dumper" mvp)
```

The <value_list> has two forms for convenience. The logical operator can appear before the left parenthesis or between each value specified. For example:

"tr = |(5000 6000 7000)" and "tr = (5000 | 6000 | 7000)"

are equilvalent in meaning. Note that mixing the and and or operators, eg. area = (a | b & c), is not permitted.

The <value_list> is useful when the TRs selected have different values for the same field name. For example, "type = |(suggestion question)" is quicker than "type= suggestion | type = question" and "summary= &(emacs ted)" is short for "summary= emacs & summary=ted". Use of the <value_list> is also more efficient in the selection process.

Evaluation of Logical Operators: Complex selection expressions can be created through the use of parentheses (), braces {}, the logical prefix operator "^", and the logical operators "|" (or) and "&" (and).

Order of Evaluation: This is determined by the precedence of the operators and by the parenthesization of subexpressions within an

<expression>. Precedence from highest to lowest is as follows:

relational operators	< ^< <= > ^> >= ^= = ~ ^~
logical and	&
logical or	¦

Operators of equal precedence are evaluated left-to-right. Parentheses can be used to override the order of precedence as parenthesized <expressions>s are evaluated first. Within parentheses, operators are evaluated according to their precedence. The logical not "^" is valid only before a parentheseized expression. For example:

^(site= GM) or site ^= GM select TRs where site is not GM.
^site = GM is invalid.

A selection expression can be broken across lines at any point and the <NL> is treated like a blank.

STRING SEARCHING:

All <field_name>s can be string searched. String searching occurs when <rel_op>s tilde (~) and not-tilde (^~) are used. The tilde can be expressed in words as meaning "contains". Both the <value> given to the right of the tilde and the value of each TR field (as defined by the <field_name>) are converted to lowercase and searching occurs using the PL/1 index function.

Clarification of string searching (~) and exact match (=):
Examples:
(a) routed_to = Ward (b) routed_to ~ ward

The routed_to field is keyed on Person_ids. Case (a) will select TRs with a routed_field of Ward only. Case (b) will convert the routed_to field to lowercase, thereby selecting TRs with Person_ids of Ward, MWard, WardD etc.

(a) summary = PL/1 (b) summary ~ PL/1

The summary field contains ascii data. Case (a) selects TRs with a summary field that matches the string PL/1 EXACTLY. Case (b) selects all TRs (after converting the summary field and the <value> to lowercase) having summary fields CONTAINING the string pl/1.

String searches are faster when keyed search selections are given in the same expression (instead of letting the selection default to the entire database). For example:
select area ~ pl/1 & type = problem instead of select area ~ pl/1

SELECTION EXPRESSION EXAMPLES:

(1) tr = ¦{3457 7194 3882}
selects the three TR reports associated with the numbers phx03457, phx07194 and phx03882.

(2) type = pb & to = Green
selects all problem reports open and routed to Mr. Green.

(3) priority = high & state ^= resolved
selects non-resolved trouble reports with a high priority.

(4) area ~ probe & {area ~ pl/1 ¦ area ~ cobol}
or
area~ probe & area ~ ¦{pl1 cobol}
selects TRs where both strings "probe" and "pl1" or both strings "probe" and "cobol" occur in the area (converted to lowercase) field.

(5) ar~probe & ar~pl/1 ¦ ar~cobol
has implied parens of: (area~probe & area~pl/1) ¦ area~cobol. This selects TRs whose area (converted to lowercase) has both probe" and "pl/"1" mentioned or has "cobol" mentioned.

(6) sub = Smith & state = ¦(iv v)
selects TRs submitted by Smith which are in the state of investigating or verified.

(7) sub=Gray & 09/10/80 > dten >= 08/01/80
selects TRs submitted by Gray between August 1, 1980 (current time is the default) and Sept. 10,1980 (current time).

(8) site= ¦{GM Ford} & area ~ FNP & pri=high &
June_1_1980_00:00 <= dtu < July_1_1980_m
selects high priority TRs changed between midnight June 1,1980 and midnight July 1,1980 where the site is GM or Ford and the area field (converted to lowercase) contains the string "fnp".

(9) state= entered ¦ ps =entered
selects TRs where either the state or the processing stage is entered.

(10) type = suggestion & (sub=Brown ¦ to =Brown)
selects suggestion TRs that have either been submitted by Brown or assigned to Brown.

(11) type = suggestion & sub=Brown ¦ to=Brown
has implied parens of:
(type = suggestion & sub=Brown) ¦ to=Brown

This selects all TRs assigned to Brown and the suggestion TRs that Brown submitted.

(12) tp=problem & dten> 08/01/80 & st ^= resolved &
(area~lister | sum ~ lister | tx~lister)
selects all unresolved problem TRs entered after August 1, 1980 with the string "lister" contained in at least one of the three fields summary, area or text.

(13) ar ~ "tr system" & dten> 07/01/80 & st = {iv | verified}
& routed_to = {Smith & Green}
selects TRs entered after July 1,1980 that have been assigned to both Smith and Green, still in the state of investigating or verified which contain the string "tr system" in the area field.

SPECIFIERS:

Specifiers refer to TRs in the subset. They are composed of TR numbers, local numbers, keywords, and the arithmetic operators '+' and '-'. Ranges are composed of two expressions separated by a colon(:) ie. '3:7'.

specifier numbers:
Both TR numbers and local numbers can be used as specifiers. The subset is re-numbered each time a selection occurs. A tr number is assumed when (1) it has a prefix 'phx' or 'tr' or (2) when the number is larger than the highest local number for the subset.

Arithmetic operators '+' and '-' are allowed with specifiers first, last, local numbers, etc. but not with tr numbers. For example, 'print current+3', 'list tr5500:last-5', 'pr 10+2'.

Keyword Specifiers:
first, f
returns the number of the first TR.
last, l
returns the number of the last TR.
previous, prev, p
returns the number of the last previous TR.
next, n
returns the number of the next TR.
current, c
returns the number of the current TR. This is initialized to 1 and changed by various requests.
all, a
references the range of all the TRs and is equilvalent to "first:last".

All the keywords except 'all' may be used in place of actual TR or local numbers.

Examples:

1	first TR in the subset
tr1	specifies TR phx00001
1:3	TRs 1 through 3
c:last	current TR through the last one
p-2	the third TR previous to the current one.

FORMAT NAMES:

The use of format names enable the user to output TR reports with the content and format already defined. The format names are:

brief, bf
specifies the TR number, state, type, stage, area and summary of the TR record.

default, df
specifies the TR record in the standard TR display format excluding the updated by, updated date, transaction, exception and history fields. This is the default.

header, he
specifies the TR record in the standard TR display format excluding the updated by, updated date, transaction, exception, history, test paths and text fields.

long, lg
specifies the entire TR record shown in the standard TR display format.

number, nb
specifies the TR number of the form phx01234.

status, st
specifies the TR record in the standard TR display format excluding the updated by, updated date, transaction, exception and text fields. Like "brief" with history fields included.

transaction, trans
specifies the TR number, state, stage and type along with the transaction header info. For example:

TR:	phx07602	State:	error
Type:	problem	Stage:	responded

[1] Problem By: Sibert, 1980-09-11 13:53:13
[2] Answer By: GJohnson, 12 September 1980 09:42 mst Fri
[3] Info By: Sibert, 12 September 1980 12:34 mst Fri

This page intentionally left blank.

[5] Answer By: Wallman, 28 October 1980 17:06 mst Tue

IOA_ CONSTRUCTS:

The use of ioa_STR allows one to format the output by using ioa_$nnl constructs. It is composed of two parts: the control string in quotes followed by the TR field names. "^a" should be used to denote the data type of each field. There are some limitations as follows:

(1) The history field is not a valid field to ioa_STR and the -format_name or -field_name controls should be used to display this field.

(2) The two combination fields, ar_sm and ar_sm_tx, are not valid output field designators to ioa_STR. The fields area, summary and text can be used to display these fields separately.

(3) For readability, the text field begins in column 1 and the summary field begins in column 16. Both are automatically broken into multiple lines of length 79. For example:

so -ctl "^a^/^a^/" tr summary
would appear as:

phx12058
Documention for tty_ in CC92-01 doesn't mention that star names are supported for the device argument in the attach description.

(4) A "^a" paired with an array TR field will cause output of all elements of the array separated by commas. For example:

(a) so -ctl "^a^-to ^a^/" tr to
would appear as:

phx12058 to Coren, Batts

(b) so -ctl "^a^-Err list: ^a^/" tr eli
would appear as:

phx09247 Err list: compose 724, compose 725

(c) so -ctl "^a is a ^a assigned to:^/^3-^a^/" tr tp to
would appear as:
phx12058 is a problem assigned to:
Coren, Batts

IOA_ CONSTRUCT EXAMPLES:
Note that the ioa_STR applies to each TR record.

(1) set_output -ctl "Number: ^a^3x State: ^a^/" tr state
followed by the request print 1:2
would appear as:

```
Number: phx09247    State: change_pending
Number: phx12058    State: verified
```

(2) so -ctl "^a^2x^a^2x^a^/" tr to site
followed by the request print 1 2
would appear as:

```
phx09247  Seaman  System_M
phx12058  Coren, Batts  System_M
```

(3) print all -ctl "^a- at stage ^a^/Assigned ^a^/" tr stage to
would appear as:

```
phx09247- at stage responded
Assigned Seaman
phx12058- at stage verified
Assigned Coren, Batts
```

HOW TO MAKE TR_QUERY WORK EFFICIENTLY.

First, it's best to make the subset as small as possible before doing any type of string searching, particularly before selecting on the area, summary, text, test_path and onbehalf_of fields.

For example, the subset can be narrowed by determining in the selection expression (when appropriate):

The type of TRs wanted
(problem, suggestion, question)
How far back to look
(date fields, in particular dten)
The processing stage of the TRs
(resolved or ^resolved, also entered, sub, etc.)
Who the TRs are/were assigned to
(to and history fields, also dist and site fields)

String selects in an and-clause (A & B) are the last criteria selected upon. "to = Soley & area ~ emacs" and "area ~ emacs & to = Soley" both get Soley's TRs first and then string searches the area for emacs. When looking for the same string in text fields, the use of ar_sm and ar_sm_tx is much more efficient and faster than using the area, text and summary fields separately.

For example, ar_sm_tx ~ fnp is equivalent to (ar~fnp ¦ sm~fnp ¦ tx~fnp).

For greater optimization, the most restricted keys of an and-clause (A & B) should be at the end of the and-clause. For example, type = problem & to = Braun. The to field is more restrictive than the type field. TRs routed_to Braun are selected and then the problems within that subset are selected. This is quicker than to = Braun & type = problem where 3000 some problem TRs are selected first and then from these, the problems assigned to Braun.

When selecting ranges of TR numbers or dates, form (a) is more efficient than its equivalent form (b).

(a) 10000 <= tr <= 11000
(b) tr <= 11000 & tr >= 10000

When selecting more than one value for a field, the and/or list form is quicker:

"to = ¦{Braun Soley}" vs. "to = Braun ¦ to = Soley",
"text ~ ¦{tr_query trq}" vs. "tx ~ tr_query ¦ tx ~ trq"

If it is really necessary to string search all problem TRs or the entire database at one time, then use the display_tr command, which is totally separate from tr_query and does not share trq's complicated selection expression overhead.