Measure Reference Manual

HP Part Number: 523324-013
Published: March 2010
Edition: J06.03 and subsequent J-series RVUs, H06.03 and subsequent H-series RVUs, G06.03 and subsequent G-series RVUs, and D40.00 and subsequent D-series RVUs.

Legal Notice

Table of Contents

List of Figures

List of Tables

About This Document

This manual describes the command and programmatic interfaces of the Measure performance monitor. This manual lets you look up specific information quickly.

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, G06.03 and all subsequent G-series RVUs, and D40.00 and all subsequent D-series RVUs, until otherwise indicated by its replacement publication.

Intended Audience

This manual is for system analysts and performance-management specialists who monitor performance on NonStop systems.

New and Changed Information

New and Changed Information for 523324–013 Revision

- Removed inaccurate reference to DEVICE-AST from "Average Service Time Counters (ZMS style Only)" (page 135).
- Removed inaccurate reference to REQUEST-AST from "Usage Notes for All DISC Entities" (page 196).
- Added description of no wait functionality to the FILE-BUSY-TIME counter in the "DDL Record for FILE Entities (ZMS Style)" (page 223).
- Added note to "PROCESSH" (page 291) to indicate that the ALLINTR option for PROCESSH sampling is disabled in H-series and J-series RVUs.
- Added information to "Usage Notes for All SQLSTMT Entities" (page 329) to indicate that you do not need to explicitly add the SQLPROC entity for the SQLSTMT entity.
- Changed the description of the run-unit field of the SQLSTMT^DESC structure in"MEASCONFIGURE" (page 358) to indicate that you can not use the asterisk (*) wild-card character to include all run-units. To include all run-units, the name of the procedure should be blank-filled.

New and Changed Information for H06.20/J06.09 RVUs (523324-012)

- Added information to plpt^flags indicating that you have to set the MEAS_PATH_SEL bit to 1 in a MEASREADACTIVE, MEAS_READACTIVE_ or MEAS_READACTIVE_MANY_ procedure call
- Changed the description of the MEASREAD (page 424) nomtime option.
- Made the following changes under the DISC entity's DDL Record Description Fields (page 185):
  - Deleted the C[n].BLOCK-SIZE field and its description.
  - Added Subfield, Description, and Counter Type information in the table under the C (page 188) field.
  - Added Subfield, Description, and Counter Type information in the table under the CN (page 195) field.
- Under "MEASGETVERSION" (page 398), added three new optional parameters and their definitions: *sysidbuf*, *sysidbuflen*, and *sysidlen*.
- Changed Cause, Effect, and Recovery information for error code 3230 in Appendix B: Error Codes (page 471).

- Made the following changes in description of the "PROCESS" (page 272) entity:
  - Added the "IPU-NUM" (page 290) and "IPU-NUM-PREV" (page 290) fields to the "DDL Record for PROCESS Entities (ZMS Style)" (page 276)
  - Documented the subsystem versions for the PROCESS entity in "Usage Notes for H-Series and J-Series PROCESS Entities" (page 291).
- Updated the Measure product versions to `H05` and `J03`.

New and Changed Information for H06.18/J06.07 RVUs (523324-011)

- Changed the recovery information for error message 3204 to indicate that, if MEASREADACTIVE encountered this error and requires a buffer larger than 32 KB, you should use the MEAS_READACTIVE_ Procedure.
- Changed the recovery information for error message 3240 to indicate translations are processed through the SQL/MX services, rather than the OSS file-system name server.
- Under Entity Specification Syntax for LINE Entities (page 236), changed the syntax specification for the LINE entity.
- Under Entity Specification Syntax for NETLINE Entities (page 244), changed the syntax specification for the NETLINE entity.
- Under Entity Specification Syntax for TERMINAL Entities (page 333), changed the syntax specification for the TERMINAL entity.
- Under MEASCONFIGURE, added statement to config^name stating that the asterisk in the first byte that indicates all configuration names can be set with `config^name^s := "*"`.
- Under the DDL Record Description Fields (page 160) of the CPU entity, added statement to the COMP-TRAPS field defining what a compatibility trap is: a data misalignment event, an unexpected transition to or from accelerated or unaccelerated code, or a relative segment 2 or 3 problem. See the *EPTRACE Manual* for further details
- Under the DDL Record Description Fields (page 278) of the PROCESS entity, added statement to the COMP-TRAPS field defining what a compatibility trap is: a data misalignment event, an unexpected transition to or from accelerated or unaccelerated code, or a relative segment 2 or 3 problem. See the *EPTRACE Manual* for further details
- Under the SQL-STMT-RECOMPILES (page 317) DDL record description field of the SQLPROC entity, indicated that the SQL-STMT-RECOMPILES counter is incremented when a similarity check fails and automatic recompilation happens. Similarity checks and automatic recompilation are explained in the *SQL/MX Programming Manual for C and COBOL*.
- Changed ERR^MISSINGSQLJOURNAL to ERR^MISSING^SQLJOURNAL in the error-code table under MEASLISTGNAME (page 412).
- Under Measure Support for Open System Services (OSS) (page 144), changed the FORMAT-VERSION (page 143) field to read: (ZMS Style only) The Measure product version of the program that formatted the entity report or structured record.
- Changed Usage Notes (page 437) under MEAS_READACTIVE_MANY_ (page 434) to read: For best performance, use the ZMS `entity-template-version` literal from the MEASDDLS file for the current RVU. Using an earlier template version or requesting legacy style records increases performance cost.
- Under MEAS_ADJUSTZMSRECORD_ (page 356), changed the *`in^record`* description to indicate that the length of *`in^record`* is derived from the information encoded in the template version of the header record pointed to by *`in^record`*.
- Changed Cause, Effect, and Recovery information for error message 3114.
- Changed Recovery information for error code 3257.
- Changed Cause and Recovery information for error code 3258.
- Under MEASINFO (page 403), indicated that in Measure G09 and later PVUs, MEASINFO has a parameter for retrieving settings that reports the configuration of the journal segment functions.

- Under MEASINFO (page 403), changed several of the settings.
- Under MEASREADCONF (page 437), indicated that in Measure G09 and later PVUs, MEASREADCONF has a parameter for retrieving settings that reports the configuration of the journal segment functions.
- Under MEASSTATUS (page 446), indicated that in Measure G09 and later PVUs, MEASSTATUS has a parameter for retrieving settings that reports the configuration of the journal segment functions.
- Under the SERVERNET (page 302) entity, added Usage Notes for CLIMs (page 311).
- Under DDL Header Fields (ZMS Style) (page 142) in the Common Entity Header Fields (page 141) section of Chapter 3: Entities and Counters (page 133), changed record-template-version to template-version. Changed the corresponding field description title to TEMPLATE-VERSION.
- Under DDL Header Fields (ZMS Style) (page 142), added a note that you should always use TEMPLATE-VERSION to get the external records corresponding to a format that you can handle. Explained how to do this under the TEMPLATE-VERSION (page 143) field.
- Under the sections, LIST *entity-type* (page 68), MEASREAD_DIFF_ (page 426), and MEASWRITE_DIFF_ (page 448), referred to the TEMPLATE-VERSION (page 143) field for details on how to obtain external records corresponding to a format that you can handle.

New and Changed Information for H06.17/J06.06 RVUs (523324-010)

Measure Structured File Limits

- Under START MEASSUBSYS, Usage Notes (page 118), changed text to indicate that the number of CIDs supported in a processor is now increased to 512,000 for each CPU for Measure H04, J02, and later PVUs.

  Changed list of ADD DEFINE commands to include =_SET_MAX_CIDS_TO_512000 option.

  Indicated that each additional 32,000 CIDs requires an additional 3.5 MB of memory for CIDs and 6.5 megabytes for counter space, a total of 10 megabytes for each CPU.
- Under STATUS MEASSUBSYS, Usage Note (page 124), changed text to indicate that the number of CIDs supported in a processor is now increased to 512,000 for each CPU for H-series and J-series RVUs.
- Under MEASMONSTATUS (page 419), changed the settings table to show 6 = 512,000 maximum CIDs.
- Under MEASMONSTATUS, Usage Notes (page 420), indicated that CID sizes of 512,000 are supported.
- Under Usage Notes for the LIST *entity-type* (page 68), LISTALL *entity-type* (page 90), and LISTACTIVE *entity-type* (page 85) commands, indicated that Measure H04 and J02 and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size, if the report output file is a structured file.
- Under MEASWRITE_DIFF_, Usage Notes (page 452), indicated that Measure H04 and J02 and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size.

Added AF_UNIX Support

- Under DDL Record for OSSCPU Entities (ZMS Style) (page 258), added eight new LCL counters and field descrptions for them:
  - LS-sends
  - LS-recvs
  - LS-queues
  - LS-send-bytes
  - LS-recv-bytes

  - LS-queue-bytes
  - LS-awakes
  - LS-selects
- Under the ADD PLOT (page 48) command, added Examples (page 50) using new counters of the OSSCPU entity.
- Added information and a sample LIST FILE display to OSS Naming Conventions (page 217).
- Changed the description of LS-MESSAGES to indicate additional types of OSS support.
- Changed examples for these OSS file opens:
  - OSS Opens of FIFOs (page 233)
  - OSS Opens of Pipes (page 233)
  - OSS Opens of AF_UNIX Sockets (page 233)
  - OSS Opens of AF_UNIX Sockets Using socketpair() (page 234)
  - OSS Opens of AF_UNIX Stream Sockets Using socket() or accept() (page 234)
- Added note to the LS-SENDS (page 264) field of OSSCPU (page 256) stating:

  Processes can be spawned in other CPUs where the descendent inherits the file descriptors (opened files) of the parent. A master socket is one created via a call to socket(), socketpair() or accept(). A satellite socket is created when a process that has created a socket spawns a process in another CPU.

  Referred to this note in other fields that discuss satellite sockets.

Miscellaneous Changes

- Removed statement from DISC (page 180) entity, SWAPS counter, “memory manager swaps two pages at a time.”
- Under the DDL Record Description Fields (page 185) of the DISC (page 180) entity's ZMS style DDl record, changed definitions for these fields:
  - STARTING-FREE-SPACE
  - STARTING-FREE-BLOCKS
  - ENDING-FREE-SPACE
  - ENDING-FREE-BLOCKS
- Added statement to LISTALL Usage Notes (page 96):

  Effective with Measure H04, J02, and later PVUs for the CPU and PROCESS entities, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the NATIVE-BUSY-TIME, ACCEL-BUSY-TIME and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.
- Under the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields of the CPU (page 156) and PROCESS (page 272) entities, added statement:

  Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.
- Under the LIST entity DDL Record for EXTNAMES file (page 79), corrected FILE-NAME-MID to MID.

New and Changed Information for H06.16/J06.05 RVUs (523324-009)

- Added the *errDetail* parameter to the MEASCONFIGURE (page 358) callable procedure. The *errDetail* parameter may only be used with Measure H03-AFV and above, and J01-AFW and above.
- Added a note to the Usage Notes (page 123) of the START MEASUREMENT (page 119) command stating that you should consider stopping all measurements before using the TACL SETTIME command or the SETSYSTEMCLOCK procedure in order to preserve measurement interval accuracy.
- Flagged several counters legacy style only. For the CONTROLLER entity, these counters are (see DDL Record for CONTROLLER Entities (page 155):
  - IO-QLEN-MAX

  For the PROCESS entity, these counters are (see DDL Record Description Fields (page 278):
  - PRES-PAGES-MAX
  - EXT-SEGS-MAX
  - RECV-QLEN-MAX
  - MAX-LCBS-INUSE
  - MAX-MQCS-INUSE
  - UCL-MAX
- Changed the CAPACITY (page 190) field description of the DDL Record for DISC Entities (ZMS Style) (page 184) to read: "Disk volume capacity in bytes. It can be used in the IF and BY clauses."
- Added USERDEF information to the LIST *entity-type* (page 68) BY option and IF option.
- Deleted the phrase, "or the literal NAME to indicate all counters" from the PLOT counter option of the ADD PLOT (page 48) command.
- Added the ZMS-style definition for STARTING-FREE-CIDS under the CPU (page 156) entity type.
- Changed the *value* options under LIST *entity-type* (page 68) and LISTALL *entity-type* (page 90) to read: "a number in the range 0 through 2147482.999. From Measure G09 and later PVUs the range is 0 through 999999999999."

New and Changed Information for H06.15/J06.04 RVUs (523324-008)

Added Telco CLIM (CLMO)

- Under SERVERNET DDL Record Description Fields, NODE-CLASS-S (page 306), added the new CLMO node class for Telco CLIMs.

  Indicated that CLIM is a wildcard for all CLIM node classes.
- Under Usage Notes for ServerNet IPC and RIPC (page 311), added notes to support node class CLMO.

  Indicated that CLIM is a wildcard for all CLIM node classes.
- Under SERVERNET Examples of Configuring Measurements for CLIMs (page 313), added the example, ADD SERVERNET * (CLMO) with description.
- Under MEASCONFIGURE (page 358) node^class, added `node^class^clim^open = "CLMO"`.

Miscellaneous Changes

Added a paragraph to the *templateversion* argument of these Measure callable procedures:

- MEAS_ADJUSTZMSRECORD_
- MEASREAD
- MEASREAD_DIFF_

- MEAS_READACTIVE_
- MEASREADACTIVE
- MEAS_READACTIVE_MANY_
- MEASWRITE_DIFF_

If it is passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Returned external records, in that case, may not match the counter record definitions with which the application was compiled.

New and Changed Information for H06.14/J06.03 RVUs (523324-007) and J06.02 (523324-06)

Changes for the NSMA (NonStop Multicore Architecture) Product in Measure J01 and Later PVUs

- Under ADD PLOT (page 48), added IPU-specific counters.
- Under STATUS MEASSUBSYS, added an example that shows the total number of running MEASIPs reported by this command.
- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style), subsection, ID Fields DDL Definition, added a new IPUS field.
- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, Counter Fields DDL Definition, added the IPU field and its three new IPU-specific counters:

  IPU-BUSY-TIME
  IPU-QTIME
  IPU-DISPATCHES

- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, DDL Record Description Fields, added the IPU-BUSY-TIME field description:

  The time during which this IPU was busy.

- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, DDL Record Description Fields, added the IPU-QTIME field description:

  The time (in microseconds) that processes spent on the ready list. Same as CPU-QTIME but for an IPU.

- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, DDL Record Description Fields, added the IPU-DISPATCHES field description:

  The number of dispatches for that IPU.

- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, DDL Record Description Fields, changed the CPU-BUSY-TIME field description to add:

  For NSMA systems in Measure J01 and later PVUs, CPU-BUSY-TIME is the aggregate of the BUSY-TIMEs of the individual IPUs that comprise the logical CPU. With REPORT RATE ON, the maximum value of this field is always 100%. With REPORT RATE OFF, this is the raw aggregate value of the BUSY-TIMEs of all its IPUs.

- Under the CPU entity, section DDL Record for CPU Entities (ZMS Style) (page 159), subsection, DDL Record Description Fields, changed the PROCESSH-SAMPLES field description to add:

  In Measure J01 and later PVUs, PROCESSH-SAMPLES reports the aggregate value of all the number of samples on all the IPUs and hence is typically equal to n*sampling frequency on that CPU.

- Under the PROCESS entity, section DDL Record for PROCESS Entities (ZMS Style) (page 276), subsection, ID Fields DDL Definition, added a new IPUS field.
- Under the PROCESS entity, section DDL Record for PROCESS Entities (ZMS Style) (page 276), subsection, Counter Fields DDL Definition, added a new IPU-SWITCHES field.

- Under the PROCESS entity, section DDL Record for PROCESS Entities (ZMS Style) (page 276), subsection, DDL Record Description fields, added the IPU-SWITCHES field description:

  An accumulating counter that indicates the total number of times that a process switched IPUs while executing. On a non-NSMA system, this is always 0.

- Under Appendix C: Subsystem Files (page 491), changed the definition of the Measure Interrupt Process (MEASIP) to the following:

  MEASIP is the Measure Interrupt Process. It does the sampling for a PROCESSH measurement. The MEASMON process creates one or more MEASIP processes that run in each CPU of the local system when the Measure subsystem starts. This process is not present in RVUs prior to H-series. MEASIP is stored in $SYSTEM.SYS*nn*.

- Added note after Figure 1-1 to indicate the differences for an NSMA system.

Changes for the CLIM (Cluster I/O module) Product in Measure H03, J01, and Later PVUs

- Under the ADD *entity-type* (page 43), DELETE *entity-type* (page 52), LIST *entity-type* (page 68), LISTACTIVE *entity-type* (page 85), and LISTALL *entity-type* (page 90) commands, added usage notes indicating that the commands allow SERVERNET, DEVICE, and DISC entity specifications using CLIMs. Examples are under the DEVICE and DISC entities.
- Under DEVICE (page 171), added syntax and descriptions, including the *sac*, *lun* and *path* options, for CLIM-attached devices. Added the *path* option for FCSA storage devices. Included examples.
- Under DDL Record for DEVICE Entities (ZMS Style) (page 174), updated the DDL record and added the new field descriptions.
- Under DISC (page 180), added syntax and descriptions, including the *sac*, *lun* and *path* options, for CLIM-attached devices. Added the *path* option for FCSA storage devices. Included examples.
- Under DDL Record for DISC Entities (ZMS Style) (page 184), updated the DDL record and added the new field descriptions.
- Under SERVERNET (page 302), added the *port* option, its description, and its values for both HP Integrity NonStop BladeSystems and other systems (the values differ according to system).
- Under SERVERNET (page 302), added the *fiber* option and its description. This option is used on NonStop BladeSystems only.
- Under SERVERNET (page 302), added the CLIM node class, description, and examples. CLIM is a wildcard for the CLMI (IP CLIM) or CLMS (Storage CLIM).
- Under DDL Record for SERVERNET Entities (ZMS Style) (page 304), updated the DDL record and added the new field descriptions.
- Under error code 3402, changed the Cause information to read: “The entity could not be accessed because of the current state. The CPU, group, module, slot number, SCSI ID, lun, or path entered for the device may not be the correct number. Find out the correct number from correct number from SCF and retry the command.”
- Under MEASCONFIGURE (page 358), SVNET^DESC, added the PF port-fiber structure for CLIM and its two subfields, *port* and *fiber*.
- Under MEASCONFIGURE (page 358), added the DEVICE^CLIM^DESC record, which contains the CLIM PLPT fields, with its *plpt^flags*, *path*, *lun*, and *target-id* subfields.
- Added error message MEAS 3122 under Appendix A: Error Messages (page 455).
- Under Usage Notes for H-Series and J-Series PROCESSH Entities (page 300), indicated that the sampling rate is per IPU for J-series RVUs.
- Under MEASGETVERSION (page 398), added the Itanium processors to the processor^type designation.

Miscellaneous Changes

- Under error code 3071, changed the Recovery information to read:

  Check the spelling and verify, using the appropriate report, that the name entered is a valid item name. Legacy style report items that are no longer present in the equivalent ZMS style report can only be plotted in the legacy style interface. And ZMS style report items that are no longer present in the equivalent legacy style report can only be plotted in the ZMS style interface.
- Changed the RESET PLOT and SET PLOT commands. The RESET command accepts attributes, but no values. To set a plot to a value other than the default, refer to the SET PLOT command.
- Changed the RESET REPORT and SET REPORT commands. The RESET command accepts attributes, but no values. To set a plot to a value other than the default, refer to the SET REPORT command.
- The MEASSQL_ procedures were published in the SOFTDOC in the Measure H01 PVU, and renamed in the Measure H02 PVU:

  MEASSQL_MAP_INIT is now MEAS_SQL_MAP_INIT_

  MEASSQL_MAP_STOP_ is now MEAS_SQL_MAP_STOP_

  MEASSQLNAME_COMPARE_ is now MEAS_SQLNAME_COMPARE_

  MEASSQLNAME_RESOLVE_ is now MEAS_SQLNAME_RESOLVE_

  MEASSQLNAME_SCAN_ is now MEAS_SQLNAME_SCAN_
- Added note to The Three Steps of Measurement (page 34) stating: The MEASCOM listings shown in the Manual are only examples and can be changed without notice. They cannot be used for parsing of output. For parsing, use the SET REPORT FORMAT STRUCTURED command.
- Under the DELETE *entity-type* command, added Examples (page 53) to describe how to include and exclude subsets of entities in a measurement. These examples are referred to in the Usage Notes of the ADD *entity-type* command, as well.
- Changed the fields, STARTING-FREE-SPACE, ENDING-FREE-SPACE, STARTING-FREE-BLOCKS, and ENDING-FREE-BLOCKS under the DDL Record for DISC Entities (ZMS Style) (page 184).
- Added the DEVICE command to LISTACTIVE Entity Specification Special Cases (page 87).
- Under LISTACTIVE *entity-type* (page 85), indicated that an asterisk (*) cannot be used for the DEVICE or DISK entities; a specific device or disk must be named.

New and Changed Information for H06.12 RVU (523324-05)

Changes for ANSI SQL Names

- Added general information under Handling of ANSI SQL Names.
- Added overview information for callable procedures under Specifying ANSI SQL Names (page 353).
- Under the ADD *entity-type* (page 43) command, added a usage note and examples to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Under the DELETE *entity-type* (page 52) command, added a usage note and examples to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Under the ENV (page 56) command, changed syntax, usage note, and examples to support ANSI SQL name-related enhancements.
- Under the INFO MEASUREMENT (page 64) command, added a usage note and example to support ANSI SQL names.

- Under the LIST *entity-type* (page 68), added a usage note to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Under the LISTALL *entity-type* (page 90), added a usage note to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Under the LISTACTIVE *entity-type* (page 85) command, added description and a usage note to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Under the LISTGNAME (page 97) command, added description and an example to support ANSI SQL names as entity specifiers or as displayable entity attributes.
- Added new SQLCATALOG (page 114) command with syntax, description, and examples to support ANSI SQL names.
- Added new SQLSCHEMA (page 114) command with syntax, description, and examples to support ANSI SQL names.
- Under the STATUS MEASSUBSYS (page 123) command, updated the example data for the SQL journal and MEASIP.
- Under SQL Journal Segment (page 147), added a paragraph stating: "ANSI SQL object names can be fully or partially qualified. If partially qualified, the omitted catalog and schema fields are resolved using MEASCOM session environment values (see the SQLCATALOG and SQLSCHEMA commands)."
- Under the LIST EXTNAMES (page 78) command, added DDL record description fields for the EXTNAMES File to show OSS pathname and ANSI SQL name (EXTNAME) external structured records.
- Under DISKFILE (page 207), Entity Specification Syntax, changed syntax, description, fields, and examples to support ANSI SQL object names and SQL/MX partition names.
- Under DISCOPEN (page 198), Entity Specification Syntax, changed syntax, description, fields, and examples to support ANSI SQL object names and SQL/MX partition names.
- Under FILE (page 216), Entity Specification Syntax, changed syntax, fields, description, and example to support ANSI SQL object names.
- Under the SQLSTMT (page 319) entity, added and changed syntax, descriptions, DDL fields, and examples to support ANSI SQL object names. Deleted the field, *disk fname-set*, which is not in use.
- Under MEASCONFIGURE (page 358), added DISKFILE entity descriptor field, DISKFILE^ANSI^DESC, to specify an ANSI SQL object.
- Under MEASCONFIGURE (page 358), added FILE and DISCOPEN entity descriptor field, FILE^OPEN^ANSI^DESC, to specify an ANSI SQL object.
- Under MEASCONFIGURE (page 358), changed the SQLSTMT^DESC entity to link to SQLSTMT^ANSI^DESC for information on ANSI SQL names.
- Under MEASCONFIGURE (page 358), added SQLSTMT entity descriptor field, SQLSTMT^ANSI^DESC, to specify an ANSI SQL object.
- Added new callable procedure, MEAS_GETDESCINFO_ (page 400), for translating an ANSI SQL name to its corresponding entity descriptor components.
- Added new callable procedure, MEAS_SQL_MAP_INIT_ (page 441), to initiate the SQL/MX mapping session.
- Added new callable procedure, MEAS_SQL_MAP_STOP_ (page 442) to stop the SQL/MX mapping session.
- Added new callable procedure, MEAS_SQLNAME_COMPARE_ (page 442), to compare two fully qualified ANSI SQL names in external format.
- Added new callable procedure, MEAS_SQLNAME_RESOLVE_ (page 443), to combine ANSI SQL name parts to create a fully qualified name in normalized external format.
- Added new callable procedure, MEAS_SQLNAME_SCAN_ (page 445), to parse a fully qualified, possibly wildcarded, ANSI SQL name in external format. This routine verifies the syntax of an ANSI SQL name.

- Under MEASCONFIGURE (page 358), added syntax and description information.
- Under MEASINFO (page 403), added description information.
- Under MEASLISTENAME (page 408), added description, usage note, and example information.
- Under MEASLISTEXTNAMES (page 411), changed description information.
- Under MEASLISTGNAME (page 412), added syntax, description, usage note, and example information.
- Under MEASMONSTATUS (page 419), changed description information.
- Under MEASOPEN (page 421), changed syntax and usage note information to reflect SQL/MX.
- Under MEASREAD (page 424), changed description information.
- Under MEASREADACTIVE (page 430)_, added description and usage note information.
- Under MEAS_READACTIVE_ (page 433), added description and usage note information.
- Under MEAS_READACTIVE_MANY_ (page 434), added description and usage note information.
- Under MEASREADCONF (page 437), changed description information and added usage note.
- Under MEASREAD_DIFF_ (page 426), changed description information and added usage note.
- Under MEASSTATUS (page 446), changed description information.
- Under MEASWRITE_DIFF_ (page 448), added description and usage note information.
- Added new error messages in Appendix A: Error Messages (page 455): 3025, 3118, and 3121. Changed error message 3117.
- Added new error codes in Appendix B: Error Codes (page 471): 3239 (ERR^BADFORMATSQLNAME), 3295 (ERR^SQL^API^INTERNAL), 3296 (ERR^SQLMX^MAP^PROCESS), 3297 (ERR^SQLMX^MAP^DUPCONNECTION), and 3298 (ERR^SQLMX^MAP^INIT). Changed description information in error codes 3238, 3240, 3242, and 3292.

Changes for Measure Limits Removal

- Under INFO MEASUREMENT (page 64), added a usage note that states: "In Measure H02 and later PVUs, the output produced by the MEASCOM INFO MEASUREMENT command displays whether counter data records have been suppressed for the measurement."
- Under INFO MEASUREMENT (page 64), added an example that displays "Counter data records suppressed."
- Under START MEASUREMENT (page 119), added a usage note that states: "In Measure H02 and later PVUs, the MEASCOM START MEASUREMENT command allows the user to select the measurement data file size, suppress counter data records in the measurement data file, or both."
- Under START MEASUREMENT (page 119), indicated that this command is enhanced to allow options to be specified in any order.
- Under START MEASUREMENT (page 119), added syntax and description for the parameters, FILESIZE and NOCOUNTERS.
- Under START MEASUREMENT (page 119), changed the data-file option to state: "If the measurement data file does not exist, Measure will create it as an unstructured file with a file code of 175, primary and secondary extent sizes of 2048 pages each, and a maximum number of extents of 256. This results in a default file size (capacity) of 1024 MB."
- Under STATUS MEASUREMENT (page 125), added a usage note that states: "In Measure H02 and later PVUs, the output produced by the MEASCOM STATUS MEASUREMENT command is enhanced to display statistics for very large data files and to display whether counter data records have been suppressed for the measurement."

- Under STATUS MEASUREMENT (page 125), added description and an example that displays "Counter Suppression OFF."
- Under MEASINFO (page 403), added settings.9 information to the *settings* option.
- Under MEASOPEN (page 421), added a usage note that states: "In Measure H02 and later PVUs, the MEASOPEN callable procedure allows the caller to select the measurement data file size, suppress counter data records in the measurement data file, or both."
- Under MEASOPEN (page 421), added options.9 information to the *settings* option.
- Under MEASOPEN (page 421), added the new *filesize* option and its description.
- Under MEASREADCONF (page 437), added settings.9 information to the settings option.
- Under MEASSTATUS (page 446), added settings.9 information to the settings option.
- In Appendix A: Error Messages (page 455), added new error messages, MEAS 3026 and 3027.
- In Table C-1: PIN Information for the Measure Product (page 492), indicated that MEASFH can run at high PIN and defaults to high PIN as of Measure H02 and later PVUs.

Miscellaneous Corrections

- In Measure G12 or later G-series PVUs, it is no longer necessary to move a G-series data file to a system running an H-series RVU in order to split the file. See Appendix D: Measure Data File Tool (MEASFT) (page 495).
- Changed the values under the Usage Note (page 498) in Appendix D: Measure Data File Tool (MEASFT) (page 495) to read 1024 MB (instead of 128 MB) and 1,073,741,824 bytes (instead of 134,217,728 bytes).
- Under the ABORT-TRANS PROCESS entity field, changed the description to read: "Number of times the process invokes the TMF ABORTTRANSACTION procedure, thus causing a TMF transaction to be rolled back."
- Under ADD MEASUREMENT (page 46) Usage Notes (All RVUs), changed second sentence to read: "This disk space is either on the same disk as the MEASCOM swap volume or on the volume specified for swap files through the SWAPVOL command."
- Under ENV (page 56), changed the description of *env-param* SWAPVOL to read: "Displays the swap volume name if it is different from the MEASCOM swap volume."
- Under INFO *entity-type* (page 62), deleted *entity-spec* option.
- Under SWAPVOL (page 128) $*volume*, indicated that if the volume name is omitted, the swap volume uses the default volume. In the Usage note, stated that the default volume is the MEASCOM swap volume.
- Under Usage Notes for H-Series and J-Series PROCESSH Entities (page 300), added an item that states: "If the *process-spec* is either a *process-name* or a *cpu,pin* combination, a *code-file-spec* needs to be specified to configure a PROCESSH measurement. If an object filename is specified as *process-spec*, the *code-file-spec* is optional. For LIST commands, it is not necessary to specify a *code-file-spec*."
- Added notes about the usage of the descriptors to MEASCONFIGURE, SQLSTMT^DESC, SQLSTMT^OSS^DESC, and SQLSTMT^ANSI^DESC. Under SQLSTMT^DESC, indicated that *run-unit* and *stmt^index* are "Not used for ANSI SQL."
- Under MEASINFO (page 403), changed setting to settings under the settings parameter list.
- Under MEASLISTCONFIG (page 407), added usage note to say: "If MEASLISTCONFIG is called at the same time a ServerNet device is dynamically added or deleted (e.g., with an SCF ADD or SCF DELETE command), the returned configuration data could contain duplicate entries or have missing entries." Also, corrected error code number to 3022 for WARN^NO^MORE^DATA.
- Under MEASOPEN (page 421), changed swapvol to say: "If *swapvol* is not specified, swap files are created on the swap volume of the calling process."
- Under ENV (page 56), changed SWAPVOL to say: "Displays the swap volume name if it is different from the MEASCOM swap volume."

- In Appendix A: Error Messages (page 455) under MEAS 3098, changed 528 to 2100 characters.
- In Appendix A: Error Messages (page 455) under MEAS 3100, changed 528 to 2100 characters and change *nnn* to *nnnn*.
- In Appendix B: Error Codes (page 471), added new error code 3320 for G09 and later RVUs.
- In Appendix B: Error Codes (page 471), changed error code 3256 (ERR^ISEGOVERFLOW) Cause and Recovery information to: "Cause: The MEASFH process received an error while attempting to sort a data file index segment." "Recovery: Close the file and retry the operation."
- In Appendix B: Error Codes (page 471), changed error code 3292 (ERR^CANNOTMAKEJOURNAL) to reword the Cause, Effect, and Recovery text.

Document Organization

This document is organized as follows:

Chapter 1: Introduction to Measure
: Provides an overview of the Measure performance monitor

Chapter 2: MEASCOM Commands
: Describes the Measure command interface

Chapter 3: Entities and Counters
: Describes how to identify each system resource to be measured

Chapter 4: Measure Callable Procedures
: Describes the Measure programmatic interface

Appendix A: Error Messages
: Describes messages you might see while using the command interface

Appendix B: Error Codes
: Describes codes that might be returned from the programmatic interface

Appendix C: Subsystem Files
: Describes the files that make up the Measure software

Appendix D: Measure Data File Tool (MEASFT)
: Describes the utility you can use to split a measurement data file into multiple smaller files

Notation Conventions

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS
: Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

  MAXATTACH

Italic Letters
: Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

  file-name

Computer Type
: Computer type letters indicate:

- C and Open System Services (OSS) keywords, commands, and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

  Use the `cextdecs.h` header file.

- Text displayed by the computer. For example:

```
Last Logon: 14 May 2006, 08:02:23
```

- A listing of computer code. For example

```
if (listen(sock, 1) < 0)
{
perror("Listen Error");
exit(-1);
}
```

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

[ ] Brackets

Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]

K [ X | D ] address
```

{ } Braces

A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]…

 - ] {0|1|2|3|4|5|6|7|8|9}…
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char…"
```

Punctuation

Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]…
```

!i and !o

In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o

In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i

In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length          !i:i
                            , filename2:length ) ;      !i:i
```

!o:i

In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                       !i
                        , [ filename:maxlen ] ) ;       !o:i
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

Nonitalic Text

Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

Italic Text

Italic text indicates variable items whose values are displayed or returned. For example:

```
p-register

process-name
```

[ ] Brackets

Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
proc-name trapped [ in SQL | in SQL file system ]
```

{ } Braces

A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
obj-type obj-name state changed to state, caused by
{ Object | Operator | Service }

process-name State changed from old-objstate to objstate
```

```
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign

A percent sign precedes a number that is not in decimal notation. The % notation precedes an octal number. The %B notation precedes a binary number. The %H notation precedes a hexadecimal number. For example:

```
%005400

%B101111

%H2F

P=% p-register E=% e-register
```

Notation for Management Programming Interfaces

This list summarizes the notation conventions used in the boxed descriptions of programmatic commands, event messages, and error lists in this manual.

UPPERCASE LETTERS

Uppercase letters indicate names from definition files. Type these names exactly as shown. For example:

```
ZCOM-TKN-SUBJ-SERV
```

lowercase letters

Words in lowercase letters are words that are part of the notation, including Data Definition Language (DDL) keywords. For example:

```
token-type
```

!r

The !r notation following a token or field name indicates that the token or field is required. For example:

```
ZCOM-TKN-OBJNAME      token-type ZSPI-TYP-STRING.          !r
```

!o

The !o notation following a token or field name indicates that the token or field is optional. For example:

```
ZSPI-TKN-MANAGER      token-type ZSPI-TYP-FNAME32.         !o
```

Related Information

Refer to the *Measure Users Guide* for basic information on using Measure. Refer to the *Operator Messages Manual* for a list of Measure operator messages.

Publishing History

Part Number	Product Version	Publication Date
523324-001	Measure D45 Measure G10	August 2002
523324-002	Measure D45 Measure G11	April 2004
523324-003	Measure D45 Measure G12	December 2004
523324-004	Measure D45 Measure G12 Measure H01	February 2007
523324-005	Measure H01 Measure H02	November 2007
523324-006	Measure H03 Measure J01	March 2008
523324-007	Measure H03 Measure J01	May 2008
523324-008	Measure H04 Measure J02	August 2008
523324-009	Measure H04 Measure J02	November 2008
523324-010	Measure H04 Measure J02	February 2009
523324-011	Measure H04 Measure J02	May 2009
523324-012	Measure H05 Measure J03	February 2010

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to docsfeedback@hp.com.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

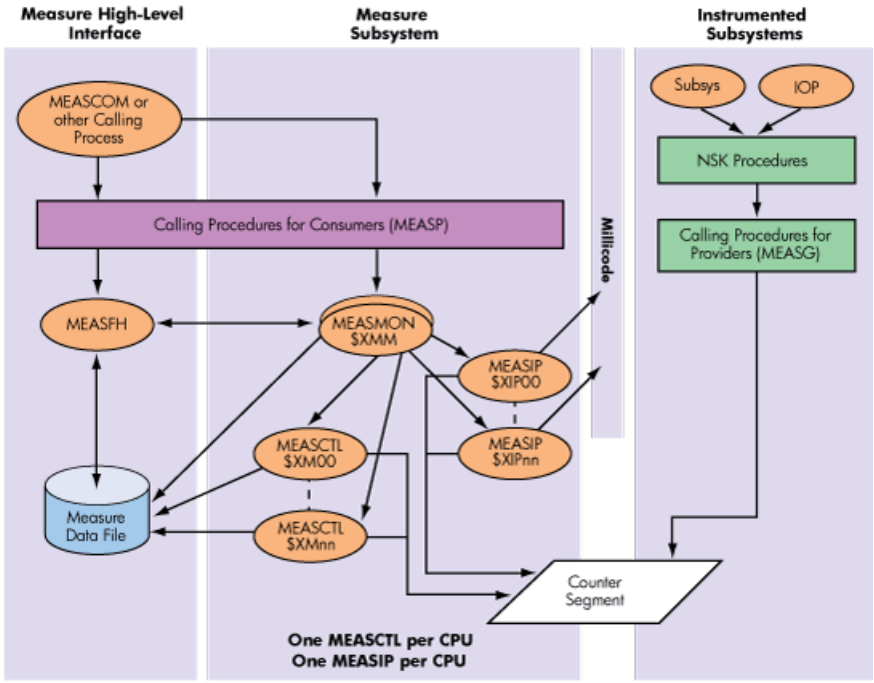
1 Introduction to Measure

The Measure performance monitor is a tool that collects performance statistics about system resources. It lets you gather data from system components, network components, and your own business applications. You can use the data to balance and tune your system, detect bottlenecks, balance workloads, and do capacity planning. Multiple users can run Measure sessions at the same time, and each user can configure and take measurements independently.

Operational Overview

Figure 1-1 shows the major components of the Measure performance monitor: the Measure subsystem, the user interface, and the subsystem instrumentation.

Figure 1-1 Operational Overview of the Measure Performance Monitor



NOTE: For NSMA systems, Figure 1-1 would show one MEASIP process per IPU, with MEASIP process names of the form "$XP*nna*," where *nn* is the CPU number (00-15) and *a* is a letter denoting the IPU number, starting with "A" for IPU 0.

The Measure Subsystem

The Measure subsystem must be running before any measurement can be made. Starting the subsystem requires a super-group user (a member of user group 255). The subsystem has minimal impact on performance. It can be started and allowed to run continuously. In some installations, an operator starts the subsystem as part of the system-load operation.

The User Interfaces

The Measure performance monitor provides two interfaces:

- The command interface is a set of commands you can enter at the terminal or input from a file. Each command calls a procedure, and each procedure performs a function such as configuring a measurement, starting the measurement, displaying collected data, and so

on. For a description of the command interface, see Chapter 2: MEASCOM Commands (page 37) and Chapter 3: Entities and Counters (page 133).

- The programmatic interface is a set of callable procedures. By calling these procedures from your applications, you can access all Measure functions. For example, you can write an application that configures a measurement, starts the measurement, stops it, and writes the data to a file for analysis. For a description of the programmatic interface, see Chapter 4: Measure Callable Procedures (page 349).

A sample measurement that shows the command interface appears later in this section. But first, you need to know two basic terms: entity and counter.

Entities and Counters

An entity is the system resource to be measured. The Measure performance monitor recognizes the types of entities in Table 3-1: Measure Entity Types (page 133).

In this manual, a class of entities (such as processes or disks) is called an entity type. The identifier for a specific individual entity in a class is called an entity specification. Thus, FILE is an entity type, and \NY.$SALES.Q1.JAN is an entity specification.

Each entity type is associated with a predefined set of counters. A counter is a programmatic structure that collects a specific type of performance data. For example, the CPU entity type has a SWAPS counter, which counts the number of swaps that occur in each measured CPU.

The system software associated with each supported entity type includes built-in calls to Measure procedures. When a measurable event occurs, the controlling software calls a procedure that increments the appropriate counter.

You can add custom instrumentation to your measurements by defining your own counters and using Measure procedure calls in your programs to increment those counters. For example, you can create a counter that is incremented each time a program executes a particular loop. Processes that carry out custom measurements are identified by the entity type USERDEF, for user-defined.

The Three Steps of Measurement

The three steps of measurement are:

1. Configure the measurement.
2. Take the measurement.
3. Examine the measurement.

The next three subsections describe a sample measurement session using the command interface. In this example, the commands you would enter are shown in boldface type. The Measure screen displays are shown in regular type.

A measurement session that uses the programmatic interface instead of the command interface follows the same basic operations, but the operations are implemented by procedure calls within a program.

NOTE: The MEASCOM listings shown in the Manual are only examples and can be changed without notice. They cannot be used for parsing of output. For parsing, use the SET REPORT FORMAT STRUCTURED command.

Step 1. Configure the Measurement

The measurement configuration determines which entities are measured.

The first three lines of this sample measurement are:

```
21> MEASCOM
1+ ADD CPU 5
2+ ADD CPU 6
```

The first line starts MEASCOM, the Measure command interface. The plus sign (+) is the MEASCOM command prompt. The next two lines configure the measurement, instructing Measure to take measurements on CPUs 5 and 6. In these commands, CPU is the entity type, and 5 and 6 are entity specifications.

Step 2. Take the Measurement

To start the sample measurement:

```
3+ START MEASUREMENT MYDATA
```

The file name MYDATA identifies the output file—that is, the file in which the Measure performance monitor is to save the measurement data. Each time you start a measurement, you must identify a new or existing measurement file.

This measurement uses many defaults. You can specify options such as a starting time for the measurement, a duration, and a measurement interval. The measurement interval causes Measure to write counters to the data file at a specified interval rather than just at the beginning and end of the measurement. For example, you might have counter values recorded at 30-minute intervals. When you examine the measurement, you can see results for successive time periods, as well as the totals.

Step 3. Examine the Measurement

To stop the sample measurement and display the results of the measurement:

```
4+ STOP MEASUREMENT MYDATA
5+ LIST CPU *

Cpu  5 Cyclone      Initial Lock Pgs 2048      Mem Pages  8192
Memory MB 16        PCBs    483                Page Size  2048
Local System \TSII From 17 Jul 1993, 15:06:38 For 30 Minutes

Cpu-Busy-Time        34.83 % Swaps                0.34
Cpu-Qtime             0.39 # Cpu-Qlen-Max           58 #
Mem-Qtime             0.02 # Mem-Qlen-Max           13 #
Dispatches           83.27   Intr-Busy-Time       1.45 %
Process-Ovhd          0.02 % Send-Busy-Time       0.61 %
Disc-IOs              9.56   Cache-Hits          17.42
Transactions                 Response-Time
Page-Requests                Page-Scans
Ending-Free-Mem       4092   Ending-UCME
Ending-UDS             500   Ending-SDS           1500
Ending-UCL             100   Ending-SCL           2000

++

Cpu  6 Cyclone      Initial Lock Pgs 2048      Mem Pages  8192
Memory MB 16        PCBs    483                Page Size  2048
Local System \TSII From 17 Jul 1993, 15:06:38 For 30 Minutes

Cpu-Busy-Time        16.40 % Swaps                0.58
Cpu-Qtime             0.22 # Cpu-Qlen-Max           58 #
Mem-Qtime             0.03  Mem-Qlen-Max           13 #
Dispatches           35.14   Intr-Busy-Time       0.61 %
Process-Ovhd          0.05 % Send-Busy-Time       0.17 %
Disc-IOs              5.34   Cache-Hits          10.24
Transactions                 Response-Time
Page-Requests                Page-Scans
Ending-Free-Mem       4092   Ending-UCME
Ending-UDS             500   Ending-SDS           1500
Ending-UCL             100   Ending-SCL           2000
```

The LIST command displays the results of the measurement on the screen. Each measured CPU has its own report. The counter values appear as percentages or totals.

Other output options exist as well. If you specified a measurement interval, you can examine the counter values taken at each interval, in addition to the final values. You can display uninterpreted values—that is, values that are not converted to percentages—and you can choose between a full report and an abbreviated report.

For most entity types, you can also view counter values while the measurement is in progress.

When a measurement is complete, you can display one or more counter values as a plot. The plot provides a graphical view that makes the values easier to analyze. To identify a counter to be plotted and create the plot:

```
6+ ADD PLOT CPU-BUSY-TIME
7+ LIST PLOT

  0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+ ...
A ************************|
B ***********|
  0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+ ...

      Min value = 16.036              Max value = 34.827

      A: CPU-BUSY-TIME                Cpu 5
      B: CPU-BUSY-TIME                Cpu 6
```

The resulting plot shows the CPU-BUSY-TIME counter values from the two measured CPUs. This plot is in bar-graph format, with each counter value represented by a horizontal row of asterisks. You can also use a two-axis format in which one axis represents counter values and the other axis represents time intervals. For either type of plot, you can change the orientation (horizontal or vertical), the scale of values, and the time window of the plot.

2 MEASCOM Commands

This chapter explains how to use the Measure command interface, MEASCOM, and describes each MEASCOM command. For more information, see General Information About Using MEASCOM (page 39).

Summary of MEASCOM Commands

Table 2-1 MEASCOM Commands

Command	Function	Page
Subsystem Control Commands		
START MEASSUBSYS	Start the Measure subsystem.	117
STATUS MEASSUBSYS	Display status of the Measure subsystem.	123
STOP MEASSUBSYS	Stop the Measure subsystem.	126
Measurement Commands		
ADD *entity-type*	Select an entity to measure.	43
ADD COUNTER	Select a user-defined counter.	44
DELETE *entity-type*	Delete an entity from those selected to measure.	52
DELETE COUNTER	Delete a user-defined counter.	54
INFO *entity-type*	Display entities selected to measure.	62
INFO COUNTER	Display information about user-defined counters.	63
SQLCATALOG	Specify a default catalog for expanding partially qualified ANSI SQL names.	114
SQLSCHEMA	Specify a default schema for expanding partially qualified ANSI SQL names.	114
START MEASUREMENT	Start a measurement.	119
STATUS MEASUREMENT	Display information about a measurement.	125
STOP MEASUREMENT	Stop a measurement.	127
Display Commands		
ADD MEASUREMENT	Select a measurement data file to examine.	46
DELETE MEASUREMENT	Delete a measurement data file from those selected to examine.	55
INFO MEASUREMENT	Display information about the measurement data files.	64
SET REPORT	Set the report format options.	108
RESET REPORT	Reset the report format options to default values.	103
SHOW REPORT	Display the current report format options.	116
LIST *entity-type*	Display a report on selected entities.	68
LIST EXTNAMES	Create or append to a key-sequenced file of records for mapping Measure OSS PATHID values to OSS pathnames, or Guardian file names or MIDs to ANSI SQL names and Guardian file names.	78

Table 2-1 MEASCOM Commands *(continued)*

Command	Function	Page
LIST OSSNAMES	Create or append to a key-sequenced file of records that map Measure OSS PATHID values to OSS pathnames, or Guardian file names or Measure MID values to their OSS file pathname equivalents.	80
LISTALL *entity-type*	Display a report about selected entities by interval records.	90
LISTACTIVE *entity-type*	Display a report on an active entity.	85
LISTENAME	Translate a Guardian file name and creation version serial number (CRVSN) to its corresponding external name (ANSI SQL name or OSS pathname).	96
LISTGNAME	Translate an OSS file pathname to its corresponding Guardian file name and creation version serial number (CRVSN).	97
LISTPNAME	Translate a Guardian file name and creation version serial number (CRVSN) to its OSS file pathname equivalent.	98
OSSPATH	Declare a default directory for use in expanding OSS file pathnames.	100
PAGESIZE	Set the number of output lines to be displayed before the double prompt (++).	101
Plot Commands		
ADD PLOT	Select a counter to plot.	48
DELETE PLOT	Delete a counter from those selected to plot.	55
INFO PLOT	Display information about the counters selected to plot.	67
SET PLOT	Set the plot format options.	104
RESET PLOT	Reset the plot format options to default values.	102
SHOW PLOT	Display the current plot format options.	116
LIST PLOT	Display a plot of selected counters.	80
Basic Commands		
ASSUME	Select a command object.	51
COMMENTS	Selectively display or suppress comment messages.	52
ENV	Display the environment parameters.	56
EXIT	End the MEASCOM session.	58
FC	Change and execute a command.	59
HELP	Display command syntax and error messages.	59
HISTORY	List previously entered commands.	61
LOG	Log commands and responses to a file.	99
OBEY	Execute a command (OBEY) file.	100
OUT	Redirect command responses to a file.	101
RUN	Run another process without exiting Measure.	104
SETPROMPT	Display environmental information in the MEASCOM prompt.	114

Table 2-1 MEASCOM Commands *(continued)*

Command	Function	Page
SWAPVOL	Specify the swap file volume.	128
SYSTEM	Select a default system.	128
TIME	Display the current date and time from a specified system.	129
VOLUME	Select a default volume.	129
WARNINGS	Selectively display or suppress warning messages.	130
!	Reexecute a command.	130

General Information About Using MEASCOM

Syntax Conventions for MEASCOM Commands

Multiple Commands on a Line

A MEASCOM command line can contain more than one command. Use a semicolon (;) to separate multiple commands.

Multiline Commands

To continue a command to the next line, end the line with an ampersand (&). MEASCOM displays a continuation prompt (also an ampersand) on the next line. Text you enter on that next line is treated as part of the preceding command. Each line can contain up to 132 characters. You can continue a command over several lines, up to a total of 1100 characters.

Comments Within a Command

To embed comments in a command, put a blank, two hyphens, and a blank (--) before and after the comment. (Blanks are recommended but not required. They prevent problems because hyphens are valid in user-defined names.) The end of a line also ends any comment ahead of it.

Command Prompts and Display

MEASCOM displays a plus sign (+) prompt when it is ready to accept a new command or an ampersand (&) when it is ready for you to continue a command from the preceding line. The prompts distinguish between lines that you enter and lines that MEASCOM generates (lines without prompts).

The LIST commands generate several reports in succession, pausing between each to let you examine the report. A double prompt (++) indicates that another report follows. To display the next report, press the Return key. To skip any remaining reports and return to the input prompt, press Ctrl-Y (the BREAK key sequence, which is also available from the BREAK key on some keyboards).

Some Measure online help displays use the double prompt (++) to indicate that more information is available. Again, press Return to display the next screen, or press Ctrl-Y to return to command entry.

In Measure G09 and later PVUs, you have some control over the scrolling of Measure entity reports on your screen. The PAGESIZE command on page 101 lets you declare the number of lines to display before another prompt is issued. The presence of OSS file pathnames in the displays makes the report formats variable in length. To prevent information from scrolling from a display before it can be read, use PAGESIZE to directly control the output length.

Disk File Names

To designate a disk file, you can use any of these forms:

```
filename
subvolume.filename
$volume.subvolume.filename
$volume.#filenum
```

You can use an asterisk as a wildcard to replace an element of the file name. For example:

- $SR8.QUAL.* refers to all files in the subvolume QUAL on the volume $SR8.
- $*.QUAL.STATUS refers to files named STATUS in subvolume QUAL on any volume.
- $SR8.*.* refers to all subvolumes and all files on volume $SR8.
- $*.*.* refers to all files in all subvolumes on all volumes.

You cannot use an asterisk to represent only part of a file-name element. For example, it would be invalid to specify $SR* as the volume name.

MEASCOM uses the file-system procedures to parse file names. You can also use logical DEFINE and device names in place of file names. For example, to configure measurement of a FILE entity whose name is defined by =_myfile:

```
1+ ADD FILE =_myfile
```

To configure measurement for the DISC volume corresponding to LDEV 14:

```
2+ ADD DISC $14
```

If the file system cannot convert the logical file name to a physical file name, or if the physical name does not correspond to the specified entity type, an error is returned.

NOTE: Logical device numbers are specific to a given system. If a data file is moved from one system to another, any existing logical device number for that file becomes invalid.

On systems running G-series or later RVUs, logical device numbers are determined dynamically at system startup. Therefore, the logical number for a given device is likely to change as other devices are added to or deleted from the system. For this reason, do not use logical device numbers in command (OBEY) files or other noninteractive files on systems running G-series or later RVUs.

Abbreviations in Commands

When you enter MEASCOM commands interactively at the + prompt, you can abbreviate the keywords for commands, objects, attributes, and counter names. For example, instead of entering STATUS MEASUREMENT, you can enter STAT MEASU.

- MEASCOM matches abbreviations word by word and gives priority to exact matches. For example, LIST is always recognized as the LIST command, not as an abbreviation of LISTALL.
- MEASCOM recognizes DISK as an alternate spelling for the keyword DISC, so you cannot use DISK as an abbreviation for DISKFILE.
- For commands, objects, and attributes, MEASCOM compares each abbreviation against all other commands, objects, and attributes. Each abbreviation you use must be unique. For example, RE is not accepted as an abbreviation for the RESET command because it also matches the REPORT object.
- For counter names, MEASCOM compares abbreviations only against other counter names for the specified entity type. For example, the DISC entity has counters named REQUEST-QTIME and REQUEST-QLEN-MAX. You could abbreviate these names as REQUEST-QT and REQUEST-QL. Hyphens are treated as part of the name, not as spaces. You cannot abbreviate REQUEST-QTIME as REQ-QT.
- As new keywords are added to MEASCOM, you might need to modify the abbreviations you use.
- You cannot use abbreviations in noninteractive command entry (that is, when you execute a MEASCOM command from the TACL prompt).

- You cannot use abbreviations in a command (OBEY) file or in an input file used during noninteractive command entry. Because abbreviations might change from RVU to RVU, this restriction prevents command files from becoming obsolete.
- You cannot abbreviate the topic in a HELP command. For example, you must enter HELP ADD TERMINAL, not HELP ADD TER.

Running a MEASCOM Session

The Measure subsystem must be running before any measurement can be made through the command interface or the programmatic interface. Many installations start the subsystem as part of the system-load operation, then let it run continuously.

Starting the Measure Subsystem

Only a super-group user (255,*n*) can start the subsystem.

To start the subsystem using MEASCOM, at the TACL prompt:

```
> MEASCOM START MEASSUBSYS
```

To start the subsystem programmatically, use the MEASMONCONTROL procedure. For a description, seeMEASMONCONTROL (page 418).

Starting and Stopping MEASCOM

You can use MEASCOM interactively or noninteractively. These examples of each usage assume that the Measure subsystem is already running:

- To use MEASCOM interactively, type the command MEASCOM at your command interpreter prompt. The system starts a MEASCOM process, the process prompts for input using a plus sign (+), then you enter commands interactively. At the end of the session, type EXIT or press Ctrl-Y to return to your operating-system command interpreter. For example:

```
34> MEASCOM
MEASURE - T9086D30 - (31OCT94) - \HATI
1+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  1
  $DATA1.PERF.MAY04

Number of Active MEASCTL Processes = 9
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7, 8
2+ EXIT
35>
```

- To use MEASCOM noninteractively, type MEASCOM followed by one or more Measure commands. A MEASCOM process starts, executes the specified commands, then returns control to your command interpreter. For example:

```
34> MEASCOM STATUS MEASSUBSYS
MEASURE - T9086D30 - (31OCT94) - \HATI
Number of Active (or Configured) Measurements =  1
  $DATA1.PERF.MAY04

Number of Active MEASCTL Processes = 9
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7, 8
35>
```

  As an alternative to entering commands after the keyword MEASCOM, you can specify an input file that contains Measure commands. For example:

```
35> MEASCOM /IN $MEAS.CMDS.MCONFIG /
```

  MCONFIG is a file containing these commands:

```
ADD CPU *
ADD DISC *
START MEASUREMENT $DATA.PERF.MAY04
```

For the full syntax of the MEASCOM command, see MEASCOM (page 42).

Creating a Custom Startup File

You can create a file of MEASCOM commands, similar to a command (OBEY) file, to execute each time MEASCOM is invoked. The file must be an edit file, named MEASCSTM, that contains only MEASCOM commands.

MEASCOM searches for the MEASCSTM file on the current subvolume. If it does not find the file, MEASCOM searches your default subvolume. This lets you set up different MEASCSTM files for different measurements and analysis environments.

This example shows a MEASCSTM file:

```
COMMENTS SUPPRESS 2000
WARNINGS SUPPRESS 3013
SETPROMPT VOLUME
SET REPORT RATE OFF
SET REPORT LOADID TEST
```

MEASCOM

This command starts a measurement session by starting the Measure command interface (MEASCOM).

Syntax

```
MEASCOM [ / [ IN filename ] [ , OUT listfile ] / ]
   [ command [ ; command ] ... ]
```

filename

is an existing disk file, a device other than a disk, or a process where MEASCOM is to read commands. If omitted, *filename* defaults to the current input file for your command interpreter, usually the terminal.

listfile

is a disk file, a device other than a disk, or a process to which MEASCOM is to write output. If omitted, *listfile* defaults to the current output file for your command interpreter, usually the terminal.

MEASCOM writes command output to *listfile*:

- If *listfile* is an existing disk file, MEASCOM appends output to the file.
- If *listfile* is a tape file, MEASCOM writes two consecutive file marks after writing to the file and before closing it.
- If *listfile* is a line printer or a process, MEASCOM writes a page eject after opening the file and before writing to it.
- If *listfile* does not exist, MEASCOM creates it as a new disk file.

MEASCOM writes no more than 80 characters per line to a terminal. When plotting data, MEASCOM writes more than 80 characters if the output device can handle the wider line.

command

is a Measure command. If omitted, MEASCOM reads commands from *filename*. If commands are specified, MEASCOM executes them and then returns control to the operating system command interpreter.

Usage Note

If multiple calls are needed while attempting to retrieve data from MEASFH, MEASCOM checks for BREAK before each call.

Example

To start the Measure command interface, check the status of the Measure subsystem, and stop the command interface:

```
34> MEASCOM
MEASURE - T9086D30 - (31OCT94) - \HATI
1+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  1
  $DATA1.PERF.MAY04

Number of Active MEASCTL Processes = 9
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7, 8
2+ EXIT
35>
```

ADD *entity-type*

This command designates which entity type (for example, CPU) to measure.

To measure several entity types simultaneously, enter an ADD *entity-type* command for each entity type before starting the measurement.

Syntax

```
ADD [ entity-type ] entity-spec [ , entity-spec ] ...
```

entity-type

is one of:

```
CLUSTER      DISCOPEN  OPDISK     SERVERNET  TMF
CONTROLLER   DISKFILE  OSSCPU     SQLPROC    USERDEF
CPU          FILE      OSSNS      SQLSTMT
DEVICE       LINE      PROCESS    SYSTEM
DISC         NETLINE   PROCESSH   TERMINAL
```

The *entity-type* keyword is optional if you specified it as the default object by using the ASSUME command.

entity-spec

identifies the entity to be measured. The identifiers you use are specific to each entity type. For identifier syntax, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).

Related Commands

Command	Function	Page
DELETE	Deletes an entity from the configuration	52
INFO	Displays the entities in the configuration	62
START MEASUREMENT	Starts a measurement after defining the configuration	119

Usage Note

- In Measure H01 and later PVUs, the ADD command allows DISCOPEN, DISKFILE, and FILE entity specifications using ANSI SQL names. For the specification syntax, see DISCOPEN (page 198), DISKFILE (page 207), or FILE (page 216).
- In Measure H03, J01, and later PVUs, the ADD command allows SERVERNET, DEVICE, and DISC entity specifications using CLIMs. For specification syntax and examples, refer to the SERVERNET (page 302), DEVICE (page 171) and DISC (page 180) entities.
- For a description of how to include and exclude subsets of entities in a measurement, refer to the DELETE *entity-type* command under Examples (page 53).

Examples

- To define a configuration that measures all CPUs, all disks, and all processes:

```
+ ADD CPU *
+ ADD DISC *
+ ADD PROCESS *
```

- To measure both logical (FILE) and physical (DISCOPEN) I/O operations performed on file NWREG.ACCOUNTS when that file is opened by process 6,234:

```
+ ADD FILE $*.NWREG.ACCOUNTS (6,234)
+ ADD DISCOPEN $*.NWREG.ACCOUNTS (6,234)
```

- To add all processes running a specific object file to a measurement and collect procedure (code-range) execution data:

```
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.TCPIP)
```

- To add measurement of specific code ranges and provide recognizable names via an EDIT (code 101) file, either ranges or range offsets can be specified. This example shows use of offsets to define code ranges:

```
OBJECT $a.b.prog1
CSS^6100^LEVEL3                    130206
FIRST^LOCATION                    +000000
PROCESS^EVENT                     +000401
PROCESS^EVENT^1                   +001401
PROCESS^EVENT^2                   +002401
PROCESS^EVENT^3                   +003401
PROCESS^EVENT^4                   +004401
PROCESS^EVENT^5                   +005401
L4^PROTOCOL                       +006577
CMI^PROTOCOL                      +006743
```

- This example, on H-series, shows the use of system DLLs INITDLL and MCPDLL in place of the symbol file TSYSCLR used on G-series:

```
Add Processh $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.TSYSDP2)
Add Processh $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.INITDLL)
Add Processh $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.MCPDLL)

Add Processh $SYSTEM.SYS01.TSYSDP2 (11, *) ($SYSTEM.SYS01.TSYSDP2)
Add Processh $SYSTEM.SYS01.TSYSDP2 (11, *) ($SYSTEM.SYS01.INITDLL)
Add Processh $SYSTEM.SYS01.TSYSDP2 (11, *) ($SYSTEM.SYS01.MCPDLL)
Add Processh Alltime 1 ($SYSTEM.SYS00.INITDLL)
Add Processh Alltime 1 ($SYSTEM.SYS00.MCPDLL)
Add Processh Alltime 1 (SL $SYSTEM.SYS00.TSL)
Add Processh Alltime 8 ($SYSTEM.SYS00.INITDLL)
Add Processh Alltime 8 ($SYSTEM.SYS00.MCPDLL)
Add Processh Alltime 8 (SL $SYSTEM.SYS00.TSL)
```

- To specify entities using ANSI SQL names:

```
+ add file 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
+ add discopen 'SCHEMA CATALOG_12.SCHEMA_34'
+ add diskfile 'CATALOG CATALOG_12'
```

NOTE: If an added table is being dropped and later recreated during the same measurement, only the first instance of that table is measured.

ADD COUNTER

This command adds a user-defined counter from a specified process to the current configuration.

Before configuring a user-defined counter, you must configure the process associated with that counter. To configure the process, use the ADD USERDEF command (the ADD *entity-type* command with USERDEF as the entity type).

For information about counter types, see Interpreting Counter Values (page 134).

Syntax

```
ADD [ COUNTER ] counter-name, PROCESS process-spec, counter-type [ , ARRAY size  ]
```

COUNTER

is optional if you specified COUNTER as the command object by using the ASSUME command.

counter-name

is a counter name. Counter names can contain 1 to 16 letters, numbers, and hyphens. The first character must be a letter.

process-spec

is the name of a process that maintains the counter. This process specification must match exactly the process specification in the ADD USERDEF command.

You can use a wild-card template, such as \MAIN.$ACCTS.BILLING.*, as a process specification. However, all processes indicated by the template must modify the same counters.

counter-type

is the type of counter to create. In D-series and G-series RVUs, *counter-type* must be one of:

ACCUM	Accumulating counter (length = 3 words)
FACCUM	64-bit accumulating counter (length = 5 words)
BUSY	Busy counter (length = 6 words)
QUEUE	Queue counter (length = 7 words)

In H-series RVUs *counter-type* must be one of:

ACCUM	Accumulating counter (length = 2 words)
FACCUM	64-bit accumulating counter (length = 4 words)
BUSY	Busy counter (length = 4 words)
QUEUE	Queue counter (length = 8 words)
QBUSY	Queue busy counter (length = 8 words)
TCELLBUSY	Busy counter, maintained with timer cells (length = 4 words)
TCELLQUEUE	Queue counter, maintained with timer cells (length = 8 words)
TCELLQBUSY	Queue busy counter, maintained with timer cells (length = 8 words)

size

is the length of the counter array (1 through 127). Array indexes start at 0, so a size of 1 creates a two-element array, a size of 2 creates a three-element array, and so on.

By default, a counter name is associated with a single counter value (array length of 0). You can also associate a counter name with an array of counter values.

Related Commands

Command	Function	Page
ADD USERDEF	Adds to the configuration a process that modifies one or more user-defined counters	43
DELETE COUNTER	Deletes a user-defined counter from the configuration	54
INFO COUNTER	Displays the user-defined counters in the configuration	63

Usage Notes

- To place a user-defined counter in an application, modify the source code to call the MEASCOUNTERBUMPINIT and MEASCOUNTERBUMP procedures at appropriate times. (For descriptions of these procedures, see MEASCOUNTERBUMP (page 396) and MEASCOUNTERBUMPINIT (page 397).) To collect information from a user-defined counter:
  1. Add the processes that modify the counter to the configuration by using the ADD USERDEF command.
  2. Add the counter to the configuration by using the ADD COUNTER command. Identify each process that modifies the counter by specifying an *entity-spec* exactly as specified in the ADD USERDEF command.
- The record for user-defined counters can be up to 1000 words long. Each process can maintain up to 85 named user-defined counters, and each counter can be an array of up to 128 counter values, as long as the 1000-word limit is not exceeded. Each entity in an array is considered a counter and counts against the 1000-word limit. If you have long counter arrays, a counter record overflow is possible.

  Each USERDEF entity is limited to 256 total array elements. To calculate the number of array elements, add the number of elements in each array (ARRAY *size*) plus 1 for each array defined. For example:

  ```
  ADD COUNTER COUNTER1, PROCESS MEASTEST, ACCUM, ARRAY 20
  ADD COUNTER COUNTER2, PROCESS MEASTEST, BUSY,  ARRAY 20
  ```

  The total number of array elements in this example is 42 (20 + 20 + 1 + 1). If the total number of array elements exceeds 256, error 3251 (configuration error) is returned.
- More than one process can maintain a counter. However, you can measure only one set of user-defined counters for a process. That is, if you start a measurement using a set of user-defined counters in a process, you cannot start another measurement using a different set of user-defined counters in that same process until the first measurement stops.

Example

This example adds two user-defined counters to the current configuration: TRANSACTIONS, an accumulating counter, and WAITING, a busy counter. Both counters are modified by the BILLING process.

```
+ ADD USERDEF BILLING
+ ADD COUNTER TRANSACTIONS, PROCESS BILLING, ACCUM
+ ADD COUNTER WAITING, PROCESS BILLING, BUSY
```

ADD MEASUREMENT

This command makes a measurement data file accessible to MEASCOM, so you can generate reports or plots from the data file. You must issue ADD MEASUREMENT before using the LIST, LISTALL, or LISTACTIVE commands to display data.

ADD MEASUREMENT makes a data file accessible by creating a file-handling process (MEASFH) for the file. The MEASFH process performs all I/O for that measurement data file. MEASFH is

created the first time you issue ADD MEASUREMENT for a file and remains active until you delete access to the data file or end the Measure session.

Although you can access only one data file at a time, you need not delete access to a data file before making a different file accessible. For example, you can issue an ADD MEASUREMENT command for FILE1, display data from FILE1, then issue an ADD MEASUREMENT command for FILE2. FILE2 becomes the currently accessible data file, and access to FILE1 is suspended (but not deleted). This feature lets you move between data files without stopping and starting the associated MEASFH processes.

Syntax

```
ADD [ MEASUREMENT ] data-file [ , MEASFH measfh ]
```

MEASUREMENT

is optional if you specified MEASUREMENT as the command object by using the ASSUME command.

data-file

is a disk file (local or remote) containing measurement data. You can specify a currently active data file.

MEASFH *measfh*

specifies a MEASFH process other than the default for your system. *measfh* is an object file name. Specify an alternate MEASFH process when you:

- Need to process a measurement data file created by an earlier product version of the Measure subsystem. MEASFH and measurement data files are release dependent. You must use D40 MEASFH with a D40 data file, and so on.
- Specify a data file on a remote system. Although you are not required to use the MEASFH on the remote system, it is much faster to have the remote MEASFH perform the I/O to the remote data file than to perform the I/O over the network.

MEASFH object files are typically named $SYSTEM.SYS*nn*.MEASFH. To find the subvolume name for your system, use the STATUS * command. SYS*nn* is the subvolume that contains the OSIMAGE file.

NOTE: To examine data files from D-series or previous RVUs on systems running G-series RVUs, you must use a D*nn* product version of MEASCOM as well as the appropriate MEASFH. For maximum compatibility with a variety of data files, always use the most recent D*nn* product version of MEASCOM.

Usage Notes (All RVUs)

- MEASFH requires disk space three times the size of the data file for creating file indexes and external counter records. This disk space is either on the same disk as the MEASCOM swap volume or on the volume specified for swap files through the SWAPVOL command.
- Your user ID must have read access to the data file.
- If you try to access too many measurements during the same session, SEGMENT_ALLOCATE_ returns error 5 (invalid segment size). For a complete list of error numbers and their definition, see the *Guardian Procedure Calls Reference Manual*.
- The ADD MEASUREMENT command processes the MEASFH clause the first time ADD MEASUREMENT is issued for a given data file. To use a different MEASFH process on the same data file during a session, delete the current MEASFH. Use the DELETE MEASUREMENT command, reissue the ADD MEASUREMENT command, and specify a new MEASFH.

Usage Notes (G-Series and Later RVUs)

- Prior to Measure G05, a limit of 127.5 MB existed on the size of the measurement data file that could be opened for analysis. With Measure G05 and later PVUs, the Measure data file size is 890 MB. However, to create data files larger than the default size of 127.5 MB, you must allocate the data file prior to measurement. You might also need to adjust the extent size or max extents values for the data file.

  Prior to starting MEASCOM for analysis of large data files, check for sufficient disk space (for example, twice the size of the data file) to accommodate swap files. Do not use $SYSTEM for disk swap space. Swapping might interfere with system performance and memory availability to other applications. If insufficient swap space exists on the $VOLUME where MEASCOM was started, the MEASCOM segment (SEGMENT_allocate error 21) might overflow and cause a trap.

- On systems running G-series RVUs, some data files are smaller than the files for comparable measurements on systems running D-series RVUs due to a change in how MEASCTL writes data records when there is a collection interval. The difference is most noticeable in large measurements, especially those that measure all FILE and DISCOPEN entities.

Usage Notes (H-Series and J-Series RVUs)

In H-series and J-series RVUs starting with H06.07, you can use the MEASFT utility to divide the measurement data file according to split criteria such as processor number or entity type. Measurement configuration and OSS or ANSI journal data are included in both files.

You can also use MEASFT to split an existing G-series data file. In this case, you must first move the G-series data file onto a system running an H-series RVU. After running MEASFT to split the data file, either move the new data files back to the G-series RVU for analysis or run the appropriate G-series version of MEASFH on the system where you ran MEASFT, using the MEASFH parameter to specify the object file.

For information about the MEASFT utility, see Appendix D: Measure Data File Tool (MEASFT) (page 495)).

Related Commands

Command	Function	Page
DELETE MEASUREMENT	Closes a data file and deletes its current MEASFH process	55
INFO MEASUREMENT	Determines which data files are accessible and which file is used for reports and plots (current data file)	64

Examples

- To make two measurement data files accessible to MEASCOM (the data file $DATA.MEAS.MAY04 is the current data file):

```
+ ADD MEASUREMENT $DATA.MEAS.MAY03
+ ADD MEASUREMENT $DATA.MEAS.MAY04
```

  If you subsequently type this command, MAY03 becomes the current data file:

```
+ ADD MEASUREMENT $DATA.MEAS.MAY03
```

- To specify a remote measurement data file as the current data file:

```
+ ADD MEASUREMENT \NY.$METS.MEAS.MAY06, &
& MEASFH \NY.$SYSTEM.SYS14.MEASFH
```

ADD PLOT

This command adds a counter to the current plot definition.

If you already started a plot definition, adding a counter from a second data file causes MEASCOM to display a warning message, delete the current plot definition, then start a new plot definition for the new data file.

Syntax

For all entity types except PROCESSH and USERDEF, the syntax is:

```
ADD [ PLOT ] counter [ ( record-number ) ]
```

PLOT
: is optional if you specified PLOT as the command object by using the ASSUME command.

counter
: is a counter name chosen from the last report generated by the LIST command. If the report displayed multiple entities, ADD PLOT adds the specified counter from each entity to the plot definition. You can plot as many as 26 counters in one plot.

record-number
: is the ordinal number of a single record to be plotted (1 through 200).

For the PROCESSH entity type, the syntax is:

```
ADD [ PLOT ] counter [ ( code-space ) ]
```

PLOT
: is optional if you specified PLOT as the command object by using the ASSUME command.

counter
: is one of:

procedure-name	Adds an individual procedure to the plot.
CODE-RANGE	Adds all code samples. Measurements for accelerated, unaccelerated, and native mode object-code execution are combined in one plot item. When you use CODE-RANGE, depending on what is running at the time, some of the counters might not exist.
ACCEL-BUSY-SAMPLES	Adds accelerated code samples.
TNS-BUSY-SAMPLES	Adds TNS code samples.
TNSR-BUSY-SAMPLES	Adds TNS/R native code samples. (G-series only)
TNSR-BUSY-TIME	Adds TNS/R native code space. (G-series only)
NATIVE-BUSY-SAMPLES	Adds native code samples. (G-series or H-series)
NATIVE-BUSY-TIME	Adds native code space. (G-series or H-series)
IPU*n*-BUSY-TIME	Adds IPU busy-time code space. (J-series), where *n* is the IPU number
IPU*n*-QTIME	Adds IPU queue-time code space. (J-series), where *n* is the IPU number
IPU*n*-DISPATCHES	Adds IPU dispatches. (J-series), where *n* is the IPU number

code-space
: is a code-space specification. For accelerated and TNS code:

Space	Specification	Accelerated	TNS
User code	UC[.*n*], where *n* is in the range:	0-31	0-15
User library	UL[.*n*], where *n* is in the range:	0-31	0-15

Space	Specification	Accelerated	TNS
System code	SC[.*n*], where *n* is in the range:	0-31	0
System library	SL[.*n*], where *n* is in the range:	0-31	0-31

n defaults to all code spaces of the specified type.

Some additional restrictions apply to the number of code spaces you can specify. For more information, see PROCESSH on page 3-209.

For TNS/R native code:

Space	Specifications
User code	UCR
User library	ULR
System code	SCR
System library	SLR

For H-series RVUs, no *code-space* specification applies. Any value you specify will be accepted but ignored.

For the USERDEF entity type, the syntax is:

```
ADD [ PLOT ] counter [ ( array-index ) ]
```

PLOT
is optional if you specified PLOT as the command object by using the ASSUME command.

counter
is the name of a user-defined counter.

array-index
is the index number of an item in an array (0 through 127).

Related Commands

Command	Function	Page
DELETE PLOT	Deletes a counter from the plot definition	55
INFO PLOT	Displays the plot definition	67
LIST PLOT	Displays the plot	80

Usage Note

You cannot plot data displayed by the LISTACTIVE command.

Examples

- To add two counters to the plot definition:

```
+ LIST CPU 1
        .
        .   (CPU report appears here)
        .
+ ADD PLOT CPU-BUSY-TIME
+ LIST FILE $DATA1.ACCT.BOOKS
        .
        .   (FILE report appears here)
```

```
      .
+ ADD PLOT FILE-BUSY-TIME
```

- This example adds the FILE-BUSY-TIME counter from each file displayed by the LIST command. Counters are added to the plot in the same order the LIST command displays them. In this example, if more than 26 files are displayed, the counters from the 26 files that have the largest FILE-BUSY-TIME values are added to the plot definition:

```
+ LIST FILE *, BY FILE-BUSY-TIME
      .
      .   (FILE reports appear here)
      .
+ ADD PLOT FILE-BUSY-TIME
```

- This example adds the ACCEL-BUSY-TIME, TNS-BUSY-SAMPLES, and TNSR-BUSY-TIME counters (by using CODE-RANGE, which is a sum of the counters) from the object file $DATA.APPL.UPDKEYS. When you generate a plot with TIME-BASE OFF, the three counters are plotted on one line. For an example, see LIST PLOT (page 80).

```
+ LIST PROCESSH $DATA.APPL.UPDKEYS
      .
      .   (PROCESSH report appears here)
      .
+ ADD PLOT CODE-RANGE
```

- This example lists the OSSCPU records by ascending order of the C0-LS-SENDS counter, where C0 is the CPU number of the other CPU involved in the socket operation.

```
LIST OSSCPU *, BY C0-LS-SENDS ( ASCENDING )
```

  This example lists the OSSCPU record for CPU 1 and then will show a plot of the values for the C6-LS-AWAKES counter, where C6 is the CPU number of the other CPU involved in the socket operation.

```
LIST OSSCPU 1
ADD PLOT C6-LS-AWAKES
LIST PLOT
```

  Refer to the DDL Record for OSSCPU Entities (ZMS Style) (page 258) for a list of counters that can be used by the ADD PLOT command and the IF and BY clauses.

ASSUME

This command sets the default command object. Most MEASCOM commands consist of an action and an object. The object name is optional. If it is omitted, MEASCOM uses the default command object specified by the most recent ASSUME command. Initially, the default command object is MEASUREMENT.

Syntax

```
ASSUME object
```

object

is one of:

COUNTER MEASSUBSYS MEASUREMENT PLOT REPORT *entity-type*

entity-type

is one of:

```
CLUSTER      DISCOPEN    OPDISK      SERVERNET     TMF
CONTROLLER   DISKFILE    OSSCPU      SQLPROC       USERDEF
CPU          FILE        OSSNS       SQLSTMT
DEVICE       LINE        PROCESS     SYSTEM
DISC         NETLINE     PROCESSH    TERMINAL
```

Example

To use the ASSUME command to shorten command lines:

```
+ ADD $DATA.MEAS.MAY04   -- MEASUREMENT is default object
+ ASSUME REPORT
+ SET FORMAT BRIEF
+ SET TOTALS INCLUDE
+ ASSUME CPU
+ LIST 2
    .
    .   (report for CPU 2 appears here)
    .
+ LIST 4
    .
    .   (report for CPU 4 appears here)
    .
+ LIST 6
    .
    .   (report for CPU 6 appears here)
    .
```

COMMENTS

This command specifies which comment messages are displayed. Comments are identified by the word COMMENT and a number in the range 2000 through 2999.

Syntax

```
COMMENTS { DISPLAY  { ALL | comm-num [, comm-num ] ... } }
         { SUPPRESS { ALL | comm-num [, comm-num ] ... } }
```

DISPLAY ALL

causes all comments to be displayed except those specified by one or more succeeding COMMENTS SUPPRESS commands. COMMENTS DISPLAY ALL is the default.

SUPPRESS ALL

causes all comments to be suppressed except those specified by one or more succeeding COMMENTS DISPLAY commands.

comm-num

is the number of a comment (2000 through 2999).

Usage Notes

- Measure includes only one comment, numbered 2000.
- To suppress warnings (identified with the word WARNING), use the WARNINGS command on page 130.

Example

To suppress comment 2000:

```
+ COMMENTS SUPPRESS 2000
```

DELETE *entity-type*

This command deletes entities of the specified type from the current configuration.

Use the DELETE *entity-type* command in combination with the ADD *entity-type** command. DELETE *entity-type* lets you exclude a specific resource from a measurement.

Syntax

```
DELETE [ entity-type ] entity-spec [ , entity-spec ] ...
```

entity-type

is one of:

```
CLUSTER      DISCOPEN   OPDISK      SERVERNET    TMF
CONTROLLER   DISKFILE   OSSCPU      SQLPROC      USERDEF
CPU          FILE       OSSNS       SQLSTMT
DEVICE       LINE       PROCESS     SYSTEM
DISC         NETLINE    PROCESSH    TERMINAL
```

The *entity-type* keyword is optional if you specified it as the command object by using the ASSUME command.

entity-spec

identifies the entity to be measured. The identifiers you use are specific to each entity type. For identifier syntax, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).

Usage Notes

- For the PROCESSH entity, you can delete the entire entity from a measurement, but you cannot delete an individual code-space specifier.
- In Measure H01 and later PVUs, the DELETE command allows DISCOPEN, DISKFILE, and FILE entity specifications using ANSI SQL names. For the specification syntax, see DISCOPEN (page 198), DISKFILE (page 207), or FILE (page 216).
- In Measure H03, J01, and later PVUs, the DELETE command allows SERVERNET, DEVICE, and DISC entity specifications using CLIMs. For specification syntax and examples, refer to the SERVERNET (page 302), DEVICE (page 171) and DISC (page 180) entities.

Related Commands

Command	Function	Page
ADD *entity-type*	Adds an entity to the configuration	43
INFO *entity-type*	Displays the configuration	62

Examples

- To define a configuration that measures all CPUs except CPU 6:

```
+ ADD CPU *
+ DELETE CPU 6
+ INFO CPU
Add cpu *
Delete cpu 6
```

- To measure the TRANSACTIONS counter in all programs of the ACCT subvolume except ACCT.BILLING:

```
+ ADD USERDEF ACCT.*
+ DELETE USERDEF ACCT.BILLING
+ ADD COUNTER TRANSACTIONS, PROCESS ACCT.*, TYPE ACCUM
```

- To specify entities using ANSI SQL names:

```
+ delete file 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
+ delete discopen 'TABLE CATALOG_12.SCHEMA_34.TABLE_56 PARTITION PART_78'
+ delete diskfile 'SCHEMA SCHEMA_34'
```

- To exclude a subset of entities from being measured, first use the ADD command to add the set, then use the DELETE command to delete the subset. For example, if you want to measure all CPUs except CPU 0, enter the following:

```
+ ADD CPU *
+ DELETE CPU 0
+ START MTEST
```

- To measure all files on all discs except $SYSTEM, and only measure files in subvolume SYS00 of disc $SYSTEM, add each individual disc except $SYSTEM, then add the $SYSTEM.SYS00 subvolume:

```
+ ADD FILE $DISC1.*.*
+ ADD FILE $DISC2.*.*
+ ADD FILE $DISC3.*.*
+ ADD FILE $DISC4.*.*
+ ADD FILE $DISC5.*.*
+ ADD FILE $SYSTEM.SYS00.*
```

NOTE: You cannot use a wildcard to specify all files, then delete the $SYSTEM volume and add back the SYS00 subvolume of the $SYSTEM volume, for it is not possible to add a subset of an item that has been deleted. Accordingly, the following example will **NOT** work:

```
+ ADD FILE $*
+ DELETE FILE $SYSTEM.*.*
+ ADD FILE $SYSTEM.SYS00.*
```

If there are too many discs in the system to add separately, you can measure all files by using `+ ADD FILE $*` and then use the LIST command to view the files you want.

DELETE COUNTER

This command deletes one or more user-defined counters from the current configuration.

Syntax

```
DELETE [ COUNTER ] counter-name [ , counter-name ] ...
```

COUNTER

is optional if you specified COUNTER as the command object by using the ASSUME command.

counter-name

is the name of the counter to be deleted. To delete all user-defined counters from the current configuration, use an asterisk (*).

CAUTION: DELETE COUNTER deletes all configured counters of the specified name even if they are modified by different processes.

Related Commands

Command	Function	Page
ADD COUNTER	Adds a user-defined counter to the configuration	44
INFO COUNTER	Displays the user-defined counters in the configuration	63

Example

To delete the user-defined counter TRANSACTIONS (both TRANSACTIONS counters are deleted even though they are modified by two different processes):

```
+ INFO COUNTER *
Add counter TRANSACTIONS, process $PERF.QUOTAS.BILLING, accum
Add counter WAITING, process $PERF.QUOTAS.BILLING, busy
Add counter TRANSACTIONS, process $PERF.QUOTAS.INCOMING, accum
+ DELETE COUNTER TRANSACTIONS
+ INFO COUNTER *
Add counter WAITING, process $PERF.QUOTAS.BILLING, busy
```

DELETE MEASUREMENT

This command deletes one or more measurement data files from the set of those accessible by MEASCOM.

Syntax

```
DELETE [ MEASUREMENT ] data-file [ , data-file ] ...
```

MEASUREMENT

is optional if you specified MEASUREMENT as the command object by using the ASSUME command.

data-file

is the name of the measurement data file to be deleted. To indicate all accessible data files, use an asterisk (*).

Related Command

Command	Function	Page
ADD MEASUREMENT	Makes a measurement data file accessible	46

Usage Note

Use this command to discontinue access to data files you are not using. Deleting a measurement saves system resources by stopping the file-handling process (MEASFH) associated with the measurement and releasing the temporary structures maintained for the data file.

Example

To make two data files accessible by MEASCOM and subsequently make $DATA.MEAS.MAY03 inaccessible:

```
+ ADD MEASUREMENT $DATA.MEAS.MAY03
+ ADD MEASUREMENT $DATA.MEAS.MAY04
                  .
                  .
                  .
+ DELETE MEASUREMENT $DATA.MEAS.MAY03
```

DELETE PLOT

This command deletes counters from the current plot description.

Syntax

```
DELETE [ PLOT ] counter [ ( char ) ] [, counter [ ( char ) ]    ... ]
```

PLOT

is optional if you specified PLOT as the command object by using the ASSUME command.

counter

For all entity types except PROCESSH and USERDEF, COUNTER is the name of the counter to delete. Use an asterisk (*) to indicate all counters in the plot description.

For the PROCESSH entity type, *counter* is a procedure name. To delete all PROCESSH procedures, specify *counter* as CODE-RANGE.

For the USERDEF entity type, *counter* is the user-defined counter name. To delete all user-defined counters, specify *counter* as NAME.

char

is the character previously designated for plotting the counter. Use an asterisk (*) to indicate all counters named *counter*.

You must specify *char* if the current plot definition contains multiple counters that have the same name.

Related Commands

Command	Function	Page
ADD PLOT	Adds a counter to the plot description	48
INFO PLOT	Displays the plot description	67
LIST PLOT	Displays the plot	80

Example

To delete one of two CPU-BUSY-TIME counters (the warning message appears when two or more counters have the same name):

```
+ INFO PLOT *
Add measurement $SPOOL.QUOTAS.DATAGO
--A-- List Cpu 6, &
        To 20 Aug 1994, 14:29:08
      Add plot CPU-BUSY-TIME
--B-- List Cpu 7, &
        To 20 Aug 1994, 14:29:08
      Add plot CPU-BUSY-TIME
+ DELETE PLOT CPU-BUSY-TIME
MEAS 3057 Ambiguous DELETE command;
   specify which occurrence to delete.
+ DELETE PLOT CPU-BUSY-TIME (A)
+ INFO PLOT *
Add measurement $SPOOL.QUOTAS.DATAGO
--B-- List Cpu 7, &
        To 20 Aug 1994, 14:29:08
      Add plot CPU-BUSY-TIME
```

ENV

This command displays the value of one or all MEASCOM session environmental parameters.

In Measure G09 and later PVUs, the ENV command also displays the current settings for OSSPATH and PAGESIZE.

In Measure H01 and later PVUs, the ENV command displays the current setting for the SQLCATALOG (page 114) and SQLSCHEMA (page 114) commands.

Syntax

```
ENV [ / OUT filename / ] [ env-param ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an edit file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the ENV command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file. For information about the log file, see LOG (page 99).

env-param

is one of:

ASSUME	Displays the default command object
COMMENTS	Displays the current settings for the COMMENTS command
LOG	Displays the name of the default log file
OSSPATH	Displays the current default OSSPATH
OUT	Displays the name of the default output file
PAGESIZE	Displays the current PAGESIZE setting
SETPROMPT	Displays the current prompt text
SQLCATALOG	Displays the current default catalog value used to resolve partially qualified ANSI SQL names
SQLSCHEMA	Displays the current default schema value used to resolve partially qualified ANSI SQL names
SWAPVOL	Displays the swap volume name if it is different from the MEASCOM swap volume.
SYSTEM	Displays the name of the default system
VOLUME	Displays the name of the default volume and subvolume
WARNINGS	Displays the current settings for the WARNINGS command

If *env-param* is omitted, all environment parameter values are displayed.

Usage Note

MEASCOM displays the environment information in the form of the commands you would use to set the environmental parameters. This method lets you use /OUT *filename*/ to create a command (OBEY) file. You can use the command file to restore a given set of environmental parameters.

Examples

- To display environment parameters (if explicitly set, the OUT, LOG, and SWAPVOL parameters would be included in this display):

```
+ ENV
  System    \BUYER
  Volume     $DATA.MEAS
  Assume     MEASUREMENT
  Setprompt Assume
  Comments  Display All
  Warnings  Display All
```

- In Measure G09 and later PVUs, to display the current default path for OSSPATH and the current setting for PAGESIZE, respectively:

```
+ ENV
  System    \BUYER
  Volume     $DATA.MEAS
  OSSpath    "/This/is/the/path/"
  Assume     MEASUREMENT
  Setprompt  Assume
  Pagesize   24
  Comments   Display All
  Warnings   Display All

+ ENV OSSPATH
  OSSpath    "/This/is/the/path"
```

- In Measure H01 and later PVUs, to display environment parameters, including ANSI SQL names:

```
+ ENV
  System      \BUYER
  Volume      $DATA.MEAS
  OSSpath     "/This/is/the/path/"
  SQLCATALOG  'CATALOG_12'
  SQLSCHEMA   'SCHEMA_34'
  Assume      MEASUREMENT
  Setprompt   Assume
  Pagesize    24
  Comments    Display All
  Warnings    Display All
```

For example, if an ANSI SQL table name is specified as:

```
'TABLE table_56'
```

MEASCOM will prepend the default catalog and schema, if set, and the fully qualified name will be (with the values above):

```
'TABLE CATALOG_12.SCHEMA_34.table_56'
```

EXIT

This command ends a MEASCOM session.

Syntax

```
EXIT
```

Usage Notes

- MEASCOM returns a completion code when it stops or abends. The completion codes are:

Code	Meaning
0	Normal voluntary termination. No warnings or errors were issued.
1	Normal voluntary termination. Warnings were issued.
2	Abnormal voluntary termination. Error messages were issued.
3	Abnormal premature termination. MEASCOM encountered an error condition that caused it to terminate prematurely.

If the code is 0, no message is displayed, but the TACL prompt reappears. If the code is 1, 2, or 3, an explanatory message appears.

- If you specify an IN file when invoking MEASCOM, reaching the end of the file is equivalent to the EXIT command.

Example

This example shows the EXIT display for an abnormal voluntary termination:

```
46+ EXIT
CPU time 0:00:02.430
2: Process terminated with fatal errors or diagnostics
```

FC

This command operates like the TACL FC command. FC retrieves a command from the history buffer, displays it so you can modify it, then executes the modified command. For a complete description of FC, see the *TACL Reference Manual*.

Syntax

```
FC [ number  ]
   [ -number ]
   [ text    ]
```

number

is the number of the line in the history buffer that contains the command to retrieve.

-number

is the number of history-buffer lines to subtract from the current line to arrive at the command to be retrieved.

text

is a text string. The most recently entered command that begins with the text string is retrieved.

If no option is specified, the last command entered is retrieved.

Related Command

Command	Function	Page
!	Immediately reexecutes a previous command without modifying it	130

Examples

- To retrieve the command on history buffer line 15:

  ```
  21+ FC 15
  ```

- To retrieve the command three lines before the current line:

  ```
  24+ FC -3
  ```

- To retrieve the most recently entered command that begins with the text string SE (for example, a SET REPORT FORMAT BRIEF command):

  ```
  25+ FC SE
  ```

- To retrieve the last command entered:

  ```
  28+ FC
  ```

HELP

The HELP command displays quick-reference information about commands, objects, entities, and error messages.

Syntax

```
HELP [ / OUT filename / ] [ command-name        ]
                         [ command-name object ]
                         [ command-name ENTITY ]
```

```
[ command-name entity ]
[ object              ]
[ entity              ]
[ counter             ]
[ entity COUNTERS     ]
[ error-number        ]
[ ABBREVIATIONS       ]
[ ALL                 ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an edit file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the HELP command executes, MEASCOM resumes writing to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file.

command-name

provides a brief description of the specified command. (Example: HELP ADD.)

command-name object

shows the complete syntax of the specified command. (Example: HELP ADD MEASUREMENT.)

command-name ENTITY

provides general syntax information for commands in which the object is an entity type. (Example: HELP ADD ENTITY.)

command-name entity

provides detailed syntax information for the specified entity type. (Example: HELP ADD CPU.)

object

describes the attributes of the specified object, including values for each attribute. (Example: HELP REPORT.)

entity

provides a brief description of the specified entity type. (Example: HELP CONTROLLER.)

counter

provides a brief description of the specified counter and identifies the entities that use a counter of this name. (Example: HELP RESPONSE-TIME.)

entity COUNTERS

lists counter values for the specified entity. (Example: HELP CPU COUNTERS.)

error-number

provides cause, result, and recovery information about the specified Measure error message. (Example: HELP 3100.)

ABBREVIATIONS

lists all Measure keywords (commands, objects, and entities) in alphabetical order. Use this list to determine the number of characters required to abbreviate a keyword uniquely.

ALL

lists all Measure keywords (commands, objects, and attributes) by category and provides a guide to more specific help topics. Issuing HELP with no parameters is the same as issuing HELP ALL.

Usage Notes

- You cannot abbreviate the help topic. For example, you must enter HELP START MEASUREMENT, not HELP START MEASU.
- A double prompt (++) signals that more information is available for a topic. Press Return (or any other key) to see the additional information. Press Ctrl-Y or type BREAK at the double prompt to return to command entry.

Examples

- To get information about the SET command:

```
+ HELP SET
The SET command can operate on the PLOT and REPORT objects.
SET's function is to change the value of the attribute
specified to the value entered.  Each time MEASCOM is
started, the PLOT and REPORT objects are initialized to
default values.  These values can be changed by using the SET
command, or can be temporarily changed by setting the
attribute on the LIST (LISTALL, and LISTACTIVE) command.  In
this case the value is changed only for the duration of the
command.

For specific command syntax perform:
  HELP SET PLOT
  HELP SET REPORT
```

- To get the syntax of the SET PLOT command:

```
+ HELP SET PLOT
SET [ PLOT ] <plot attribute> <value> [, <plot attribute>
<value> ]...

See HELP PLOT for <plot attribute> and <value>.
```

- To identify the entities that use a READY-TIME counter and get an explanation of the meaning of the counter:

```
+ HELP READY-TIME
CPU Report:
  Time processes were executing or waiting on the ready list.
  Counter type: Queue

Process Report:
  Time this process was executing or waiting on the ready list.
  Counter type: Busy
```

- To display information about Measure error 3014:

```
+ HELP 3014
Cause:    A three-digit sequence-id is added to LOADID to
          designate each interval of the measurement, when
          "LISTALL" is used. It is suppressed if 3 characters
          are not available for use.
Result:   Sequencing is suppressed.
Recovery: Use 5 or fewer characters for LOADID.
```

HISTORY

This command lists one or more of the most recently entered commands.

Syntax

```
HISTORY [ number ]
```

number

specifies the number of history-buffer lines to display. If you do not specify a number, the last 10 lines in the history buffer appear. If the buffer contains fewer than the specified number of lines, HISTORY lists the existing lines.

Related Commands

Use the HISTORY command with:

Command	Function	Page
FC	Displays, modifies, and executes a previously entered command	59
!	Reexecutes a previously entered command	130

Example

This example shows both the HISTORY command and the FC command. The HISTORY command lists the last five lines in the history buffer (lines 28 through 32). The FC command redisplays line 31 from the history buffer. After using FC, you can either reexecute the command or edit the command, and then reexecute it.

```
32+ HISTORY 5

28+ LIST PROCESS (0,*), RATE OFF
29+ INFO MEASUREMENT *
30+ LIST DISC $SYSTEM
31+ LIST CPU *, BY CPU-BUSY-TIME
32+ HISTORY 5

33+ FC 31
33+ LIST CPU *, BY CPU-BUSY-TIME
33..
```

INFO *entity-type*

This command displays all entities of the specified type in the current configuration.

Syntax

```
INFO [ / OUT filename / ] [ entity-type ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the INFO command executes, MEASCOM resumes writing to the current OUT file (typically, the terminal). The OUT option does not affect log file contents.

entity-type

is one of:

```
CLUSTER      DISCOPEN    OPDISK      SERVERNET     TMF
CONTROLLER   DISKFILE    OSSCPU      SQLPROC       USERDEF
CPU          FILE        OSSNS       SQLSTMT
DEVICE       LINE        PROCESS     SYSTEM
DISC         NETLINE     PROCESSH    TERMINAL
```

The *entity-type* keyword is optional if you specified it as the command object by using the ASSUME command.

Related Commands

Command	Function	Page
ADD *entity-type*	Adds an entity to the configuration	43
DELETE *entity-type*	Deletes an entity from the configuration	52

Usage Notes

- If the INFO display shows an entity as both added and deleted, the delete operation takes precedence.
- MEASCOM displays the information as a list of the commands used to define the configuration. This lets you use /OUT *filename*/ to create a command (OBEY) file, with which you can later restore the current measurement configuration.
- The output of the INFO PROCESSH command is a list of the commands used to define the current PROCESSH measurement configuration. /OUT filename/ can be used to create a command (OBEY) file that stores a given PROCESSH configuration.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).
- In Measure H01 and later PVUs, the INFO *entity-type* command shows DISCOPEN, DISKFILE, and FILE entity specification using ANSI SQL names if the ANSI SQL names were used for measurement configuration. For the specification syntax, seeDISCOPEN (page 198) , DISKFILE (page 207), or FILE (page 216).

Examples

- To display all CPU entities in the current configuration:

```
+ INFO CPU
Add cpu *
Delete cpu 6
```

- This example is on G-series. Notice that it does not display code-space identifiers:

```
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (UCR $SYSTEM.SYS00.TCPIP)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (ULR $SYSTEM.SYS00.ZCRESRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (ULR $SYSTEM.SYS00.ZINETSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (ULR $SYSTEM.SYS00.ZLANCSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (ULR $SYSTEM.SYS00.ZCRTLSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (SCR $SYSTEM.SYS00.TSYSCLR)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP (SLR $SYSTEM.SYS00.TSYSCLR)
+ INFO PROCESSH
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.TCPIP)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZCRESRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZINETSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZLANCSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZCRTLSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.TSYSCLR)
+
```

INFO COUNTER

This command displays, for one or more user-defined counters, the counter name and type, and the names of any processes that modify the counter.

Syntax

```
INFO [ / OUT filename / ] [ COUNTER ] counter-name
   [ , counter-name ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* exists, MEASCOM opens the file and appends command output to it.

After the INFO command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of a log file.

COUNTER

is optional if you have specified COUNTER as the command object by using the ASSUME command.

counter-name

is a counter name or an asterisk (*) to indicate all counters.

Related Commands

Command	Function	Page
ADD COUNTER	Adds a user-defined counter to the configuration	44
DELETE COUNTER	Deletes a user-defined counter from the configuration	54

Usage Notes

- MEASCOM displays the information as a list of the commands used to define the counters. This lets you use /OUT *filename*/ to create a command (OBEY) file where you can later restore the current set of counters.

  When you save user-defined counter information, you might also want to save the current USERDEF configuration information. To save the user-defined configuration information, use INFO USERDEF with an OUT parameter.

- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).

Example

To display all user-defined counters in the current configuration:

```
+ INFO COUNTER *
Add counter TRANSACTIONS, process $PERF.QUOTAS.BILLING, accum
Add counter WAITING, process $PERF.QUOTAS.BILLING, busy
Add counter TRANSACTIONS, process $PERF.QUOTAS.INCOMING, accum
```

INFO MEASUREMENT

This command displays this information for one or more of the measurement data files currently accessible by MEASCOM:

- File name, displayed as an ADD MEASUREMENT command.
- Start time, stop time, and the collection interval of the measurement. The stop time appears for a currently active measurement only if you specified a stop time when you started the measurement.
- Information about system data space usage during the measurement. For each entity type, this includes:
  - The entity type name
  - The maximum number of entities of that type under concurrent measurement
  - The maximum number of words in system data space allocated to entities of that type

These values do not appear for a currently active data file.

- The measurement configuration, displayed as comments containing ADD and DELETE commands.
- The OSS journal segment in the data file and the current status of the OSS journal segment in the MEASCOM session (Measure G09 and later PVUs only).
- The ANSI SQL journal segment in the data file and the current status of the ANSI SQL journal segment in the MEASCOM session (Measure G11 and later PVUs only).

Syntax

```
INFO [ / OUT filename / ] [ MEASUREMENT ] data-file
     [ , data-file ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the INFO command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file.

MEASUREMENT

is optional if you specified it as the command object using the ASSUME command.

data-file

is the name of an accessible measurement data file. Use an asterisk (*) to indicate all accessible data files.

Related Command

Command	Function	Page
ADD MEASUREMENT	Makes a data file accessible by MEASCOM	46

Usage Notes

- In the display, the name of the current data file (the file MEASCOM uses when generating reports) is followed by the comment “Current Data File.” To make a different file the current data file, use the ADD MEASUREMENT command.
- The output of the INFO MEASUREMENT command is a list of the commands used to define the configuration. You can use /OUT *filename*/ to create a command (OBEY) file, which you can then edit to create a new configuration file.
- In Measure G09 and later PVUs, the OSS journal segment status appears if Measure attempted to construct a journal segment in the data file. (Measure attempts to construct a journal segment if you requested one either explicitly, in the START MEASUREMENT command, or by making journal segment construction the default, as described in the usage notes for START MEASSUBSYS (page 117).) OSS journal segment states can be:
  - OSS journal segment attached
  - OSS journal segment under construction
  - OSS journal segment error: nnnn

  Journal segment construction occurs after the measurement is stopped. When construction is complete, current openers of the file are not notified. MEASCOM sessions that have a data file open can use the OSS file-system name server to resolve OSS file pathname references. To access the journal segment, issue a DELETE MEASUREMENT command and then repeat the original ADD MEASUREMENT command.

- In Measure G11 and later PVUs, the ANSI SQL journal segment status appears if journal segment construction was attempted. (Measure attempts to construct a journal segment if you requested one either explicitly, in the START MEASUREMENT command, or by making journal segment construction the default, as described in the usage notes for START MEASSUBSYS (page 117).) ANSI SQL name segment states can be:
  - SQL journal segment attached
  - SQL journal segment under construction
  - SQL journal segment error: nnnn

  Journal segment construction occurs after the measurement is stopped. When construction is completed, current openers of the file are not notified. MEASCOM sessions that have a data file open can use ANSI SQL to resolve ANSI SQL name references. To access the journal segment, issue a DELETE MEASUREMENT command and then repeat the original ADD MEASUREMENT command.
- If OSS file pathnames are used in measurement specification, the specified pathname must be translatable at the time the INFO MEASUREMENT report is formatted. If not, a formatting error occurs. This translation can come from the system under measurement or an attached OSS file pathname. The final example highlights this behavior.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).
- In Measure H01 and later PVUs, the INFO MEASUREMENT command shows DISCOPEN, DISKFILE, and FILE entity specifications using ANSI SQL names if the ANSI SQL names were used for measurement configuration. For the specification syntax, see DISCOPEN (page 198) , DISKFILE (page 207), or FILE (page 216).
- In Measure H02 and later PVUs, the output produced by the MEASCOM INFO MEASUREMENT command displays whether counter data records have been suppressed for the measurement.

Examples

- This example displays information about all currently accessible data files. The current file is an active data file. Therefore, INFO MEASUREMENT cannot display the stop time of the measurement or the contents of the file.

```
+ INFO MEASUREMENT *
Add measurement $DATA.CAPTURE.CPUPROC - Current Data File --
Data collected from system \MEASURE, Measure release version G11.2
From 29 Aug 2003,  07:00:00
-- Add cpu *
-- Add process *
Add measurement $DATA.CAPTURE.NEWPH
Data collected from system \MEASURE, Measure release version G11.2
From 28 Aug 2003,  20:15:00,  To 28 Aug 2003,  20:30:00
Process                525 Entities          115500 Words
Processh               525 Entities          435317 Words
-- Add Processh $SYSTEM.SYS00.TSYSDP2 ($SYSTEM.SYS00.TSYSDP2)
-- Add Processh $SYSTEM.SYS00.MEASCOM ($SYSTEM.SYS00.MEASCOM)
+
```

- For an OSS measurement data file, this example shows the active measurement:

```
+ INFO MEASUREMENT *
Add measurement $SPOOL.MEASDATA.MDATA
Data collected from system \DEV, MEASURE release version G09
From 17 Nov 2000,11:19:18
OSS journal segment attached
Cpu                      6 Entities            1992 Words
Process                909 Entities          156348 Words
-- Delete Process system-processes
```

```
-- Add CPU *
-- Add Process "/bin"
```

- This example shows an INFO MEASUREMENT report with an OSS file pathname translation error (indicated by "*OSSPath*"):

```
+ INFO MEASUREMENT *
Add measurement $SPOOL.MEASDATA.MDATA -- Current Data File --
Data collected from system \DEV, MEASURE release version G09.
From 17 Nov 2000, 11:19:15,  To 17 Nov 2000, 11:19:18
Cpu                          6 Entities            1992 Words
Process                    909 Entities          156348 Words
-- Add Cpu *
-- Add Process "*OSSPath*"
```

- This example displays information about all currently accessible data files. Because the current file is an active data file, INFO MEASUREMENT cannot display the stop time of the measurement or the contents of the file.

```
+ INFO MEASUREMENT *

Add measurement $DATA.CAPTURE.CPUPROC - Current Data File --
Data collected from system \MEASURE, Measure release version G11

From 29 Aug 2003,  07:00:00
-- Add cpu *
-- Add process *

Add measurement $DATA.CAPTURE.NEWPH
Data collected from system \MEASURE, Measure release version G11

From 28 Aug 2003,  20:15:00,  To 28 Aug 2003,  20:30:00
Process                  525 Entities          115500 Words
Processh                 525 Entities          435317 Words
-- Add Processh $SYSTEM.SYS00.TSYSDP2 ($SYSTEM.SYS00.TSYSDP2)
-- Add Processh $SYSTEM.SYS00.MEASCOM ($SYSTEM.SYS00.MEASCOM)
+
```

- This example shows an INFO MEASUREMENT report that includes ANSI SQL names:

```
+ info measurement $guest.measure.demo
Add measurement $GUEST.MEASURE.DEMO -- Current Data File --
Data collected from system \DEMO, MEASURE release version H01.
From 26 Feb 2003, 15:59:54
-- Add File 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
-- Add Discopen 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
-- Add Diskfile 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
```

- This example shows an INFO MEASUREMENT report where counter data record suppression was specified for the measurement. Measure H02 and later PVUs enable this suppression feature.

```
+ INFO MEASUREMENT MEASXX
Add measurement $PERF.GREG.MEASXX -- Current Data File --
Data collected from system \YOSPRD, MEASURE release version H02.
From 16 Aug 2007, 13:52:32
Counter data records suppressed
-- Add Cpu 0
```

INFO PLOT

This command displays the current plot definition.

Syntax

```
INFO [/OUT filename/] [ PLOT ] counter [ (char) ]
    [ , counter [ (char) ] ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the INFO command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file.

PLOT

is optional if you specified PLOT as the command object by using the ASSUME command.

counter

for all entity types except PROCESSH and USERDEF, COUNTER is a counter name. Use an asterisk (*) to indicate all counters.

- For the PROCESSH entity type, *counter* is a procedure name. To display all procedures in the plot, specify *counter* as CODE-RANGE.
- For the USERDEF entity type, *counter* is the user-defined counter name. To display all user-defined counters in the plot, specify *counter* as NAME.

char

is the plot character of the plotted counter to display or an asterisk (*) to indicate all counters named *counter*. If omitted, the default is the asterisk (*).

Usage Note

The output of INFO PLOT is a list of the commands used to add the counters to the plot definition. You can use /OUT *filename*/ to create a command (OBEY) file that stores a given plot definition.

Related Commands

Command	Function	Page
ADD PLOT	Adds a counter to the plot definition	48
DELETE PLOT	Deletes a counter from the definition	55
LIST PLOT	Displays the plot	80

Example

To display the current plot definition:

```
+ INFO PLOT *

Add measurement $SPOOL.QUOTAS.DATAGO
--A-- List Cpu 6, &
        To 20 Aug 1994, 14:29:08
      Add plot CPU-BUSY-TIME
--B-- List Cpu 7, &
        To 20 Aug 1994, 14:29:08
      Add plot CPU-BUSY-TIME
```

LIST *entity-type*

This command reads data from the current data file (the file most recently specified in an ADD MEASUREMENT command) and displays a report for each entity included in the *entity-spec* parameter.

A double prompt (++) signals that another report follows the one on the screen. Press Return (or any other key) to view the next report. Press Ctrl-Y or type BREAK at the double prompt to ignore subsequent reports:

- If you entered multiple commands on the previous command line, BREAK causes the current command to be interrupted and the next command to be executed.
- If you did not enter multiple commands, BREAK interrupts the current command, and the MEASCOM prompt returns.

Syntax

```
LIST [ / OUT filename / ] [ entity-type ] entity-spec
    [ , list-option ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the LIST command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file.

entity-type

is one of:

CLUSTER	DISCOPEN	OPDISK	SERVERNET	TMF
CONTROLLER	DISKFILE	OSSCPU	SQLPROC	USERDEF
CPU	FILE	OSSNS	SQLSTMT	
DEVICE	LINE	PROCESS	SYSTEM	
DISC	NETLINE	PROCESSH	TERMINAL	

The *entity-type* keyword is optional if you specified it as the command object by using the ASSUME command.

For special considerations for the FILE and PROCESS entities, see Usage Notes (page 74).

entity-spec

identifies the entity to be measured. The identifiers you use are specific to each entity type. For identifier syntax, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).

list-option

is one of:

BY *item-name* [(ASCENDING) | (DESCENDING)]

sorts the report in ascending or descending order.

item-name is one of:

{ *counter* }
{ *identification-item* }

where

counter is a counter name. By default, counter items are sorted in descending order.

Special considerations for the PROCESSH, DISC, CPU, and USERDEF entity types include:

- For PROCESSH, *counter* is a procedure name.
- For legacy style DISC cache counters, *counter* must be preceded by C0-, C1-, C2-, or C3- to specify the size of the cache blocks to be considered. For ZMS style DISC cache counters, *counter* must be preceded by C*n*- where *n* is a value from 0 to 7.
- For CPU, if a counter is not specified, the CPUs are listed in numeric order.
- For USERDEF, the valid counter names are *name*, *value*, *type*, and *index*.

identification-item is an identification item such as CPU-NUM or PROCESS-NAME. You can use any valid identification item for the specified entity. By default, identification items are sorted in ascending order.

CR-NAME-LEN { SHORT | LONG }

controls the displayed length of procedure (code-range) names. The default is SHORT.

SHORT

truncates procedure (code-range) names (if necessary) at 32 characters. Mangled procedure names are not demangled prior to display.

LONG

procedure (code-range) names both in their SHORT form and, if longer than 32 characters, in their entirety on a subsequent line. Mangled procedure names are not demangled for display.

CR-NAME-FORM { STANDARD | DEMANGLED | BOTH }

controls whether demangled procedure (code-range) names are displayed (if applicable). The default is STANDARD.

STANDARD

displays procedure (code-range) names in the form specified in the code file (mangled) or EDIT file (demangled).

DEMANGLED

demangles procedure (code-range) names, if necessary, prior to display.

BOTH

displays procedure (code-range) names in both STANDARD and DEMANGLED forms.

CR-NAME-QUAL { UNQUALIFIED | QUALIFIED }

controls whether procedure (code-range) names are displayed with object file name qualifiers, if available, in the Code-Range Name column on the line immediately following the traditional line of code-range output. The default is UNQUALIFIED.

UNQUALIFIED

displays procedure names in traditional form with no qualifiers.

QUALIFIED

displays procedure names with the Guardian or OSS object file name of the associated code. This can be useful in differentiating between like-named procedures in different object files. Guardian file names have their associated CRVSN appended to the end of the name.

DOTS { ON } { OFF }

specifies whether report displays include connecting dots between labels and numeric values. Valid only if STYLE is ZMS. The default is OFF.

ON

displays dots to connect labels to formatted numbers.

OFF

does not display dots to connect labels to formatted numbers.

FOR *duration*

specifies the duration of the report window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement ends or when measurement of the entity ends, whichever is earlier.

duration

is a time interval in one of these formats:

n SECOND[S] *n* MINUTE[S] *n* HOUR[S]

where *n* is an integer in the range 1 through 9999.

FORMAT { BRIEF } { NORMAL } { STRUCTURED }

sets the format of the report. The default is NORMAL.

BRIEF

displays an abbreviated report that contains the most commonly used counters for each entity type.

For information about which counters are included in brief-format reports, see the DDL record for the specified entity type in Chapter 3: Entities and Counters (page 133). Counters included in brief-format reports are in **boldface type** in each DDL record.

NORMAL

displays all counters.

STRUCTURED

writes the report to a structured file for subsequent examination using Enform. A single record consists of all counters for a single entity. (For DDL record formats, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).)

Reports for entities of the same entity type are written to the same file, and the file is named for the entity type. If the file already exists, MEASCOM appends the data to the file. If the file does not exist, MEASCOM creates it. (Structured files are closed when you modify the FORMAT attribute.)

FROM [[*start-date*,] *start-time-of-day*]

specifies the start of the report window. If you omit FROM, the start time of the measurement or the start time of the measurement of the entity is used, whichever is later.

start-date

is the date on which the report window starts, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 to 31.

mmm is the first three letters of the month; for example, JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *date*, the start date of the measurement is used.

start-time-of-day

is the time the report window starts, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *time-of-day*, the start time of the measurement is used.

IF { *item-name operation value* }

determines which data records are included in the report. Only records that meet the specified condition are included.

item-name

is one of:

{ *counter* }
{ *identification-item* }

where

counter is a counter name. Special considerations for these entity types include:

- For the PROCESSH entity type, *counter* is a procedure name.
- For the cache counters of the DISC entity type, *counter* must be preceded by C0-, C1-, C2-, or C3- to specify the size of the cache blocks to be considered.

identification-item is a numeric entity identification item of INT or INT(32) type. To determine which identification items can be used, see the MEASDDLS file or the descriptions of each entity descriptor in MEASCONFIGURE (page 358).

Character string items such as PROCESS-NAME and fixed-length items such as FROM-TIMESTAMP cannot be used in the IF clause.

An exception to the rules above is the USERDEF entity, for which a character string can be used if the counter equals *name*. For example:

```
+ list userdef *, if name = MYCOUNTER
```

operation

is one of:

> (greater than)	= (equal to)
< (less than)	<> (not equal to)

value

is a number in the range 0 through 2147482.999. From Measure G09 and later PVUs the range is 0 through 999999999999.

LOADID *loadid*

specifies the name to be placed in the *loadid* field of the records generated by this command. The LOADID option applies only to structured reports.

loadid

is an alphanumeric string, 1 through 8 characters long. The string can contain letters, numbers, carets (^), hyphens (-), and underscores (_). The first character must be a letter.

RATE { OFF | ON }

determines how counter values are displayed. The RATE attribute has no effect on structured files. The default is ON.

OFF

displays uninterpreted counter values.

ON

displays interpreted counter values (counts per second and percent busy).

STYLE { LEGACY } { ZMS }

sets whether displays or structured data are formatted using the ZMS style interface or using the legacy interface compatible with pre-G11 Measure PVUs. In G-series RVUs, the default is LEGACY; in H-series RVUs, the default is ZMS.

LEGACY

displays and structured records use the legacy style.

ZMS

displays and structured records use the ZMS style.

TO [*end-date*,] *end-time-of-day*

specifies the end of the report window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement ends or when measurement of the entity ends, whichever is earlier.

end-date

is the date on which the report window ends, in the same format as the *start-date*.

If you omit *end-date*, the end date of the measurement is used.

end-time-of-day

is the end time of the report window, in the same format as the *start-time-of-day*.

If you omit *end-time-of-day*, the end time of the measurement is used.

TOLERANCE { ON | OFF }

is the tolerance to be applied in deciding which measurements are included in the report:

- TOLERANCE OFF applies the FROM and TO limits exactly as specified.
- TOLERANCE ON (the default) interprets the FROM and TO limits as a range bounded by plus or minus one-half of the measurement interval. For example, if the specified FROM time is 8:00 and the measurement interval is 30 minutes, the actual FROM time can be as early as 7:45. Similarly, if the specified TO time is 10:00, the actual TO time can be as late as 10:15.

TOTALS { INCLUDE | ONLY | SUPPRESS }

determines whether TOTALS are displayed. The default is SUPPRESS.

INCLUDE

indicates that both the per-process and aggregated totals are displayed. When aggregated totals are displayed, the set of totals is enclosed in brackets ([…]), and the number of processes that were executing the code being totaled precedes the individual code-range results, also enclosed in brackets (that is, [Totals across 3 processes]).

ONLY

For all entities except PROCESSH and USERDEF, displays only the final TOTALS report. If only one entity report is generated, the TOTALS report is not displayed.

For the PROCESSH entity, only aggregated PROCESSH data is displayed.

NOTE: PROCESSH data can be collected on either a per-code-file basis or a per-procedure basis, depending on whether a code-file-spec was supplied when the PROCESSH measurement was configured (via the ADD PROCESSH command). Thus, the displayed totals can cover an entire code-file or each of a set of code ranges within the code-file.

SUPPRESS

displays only the entity reports. (The TOTALS attribute is ignored if a report contains only measurements for one entity.)

For PROCESSH, SUPPRESS indicates only per-process data is displayed (aggregated data is not displayed).

SUPPRESS has no effect on USERDEF reports.

ZERO-REPORTS { INCLUDE | SUPPRESS }

determines whether records containing all zero values are displayed. The default is SUPPRESS.

INCLUDE

displays entity reports even if all counter values are zero.

SUPPRESS

does not display entity reports if all counter values are zero.

ZERO-VALUES { INCLUDE | SUPPRESS }

determines whether values less than 0.005 are displayed as zeros or blanks. The default is SUPPRESS.

INCLUDE

displays zeros if counter values are less than 0.005.

SUPPRESS

displays blanks if counter values are less than 0.005.

The ZERO-VALUES attribute has no effect on structured files.

Related Commands

Command	Function	Page
SET REPORT	Specifies the report format	108
LISTACTIVE	Displays an active measurement	85

Usage Notes

- The FILE and PROCESS entities display the names of open files. A file can include resources whose names do not conform to file name syntax, such as a process. To display reports for these entities, specify the CPU, PIN, and file number of the resource rather than its file name. For an example, see Examples (page 75).
- If you use the FROM, FOR, or TO clauses to specify a report window to view only a portion of the measurement data, LIST generates the report:
  - If the data was collected with no collection interval, only entities that started and stopped within the specified report window are included in the report.
  - If the data was collected with a collection interval, for each entity you specify, MEASCOM reads the record written nearest to the beginning of the report window and the record nearest to the end of the report window. MEASCOM then generates the report from the differences between the two records. Both records must be within the report window or within one collection interval of the start or end time of the window.
- You cannot generate a sorted report (BY *item-name*) if the report output file is a structured file. The FORMAT option of the REPORT object determines the output format.
- In G-series RVUs, IF clauses cannot refer to fields that appear only in ZMS style records and reports, even in the ZMS style report mode.
- In H-series RVUs, IF clauses cannot refer to fields that appear only in legacy style records and reports, even in the legacy style report mode.
- The list options CR-NAME-LEN, CR-NAME-FORM, and CR-NAME-QUAL are valid for the PROCESSH entity only.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).
- In Measure H01 and later PVUs, the LIST command allows DISCOPEN, DISKFILE, FILE, and SQLSTMT entity specifications using ANSI SQL names. For the specification syntax, see DISCOPEN (page 198), DISKFILE (page 207), FILE (page 216), or SQLSTMT (page 319).

- In Measure H03, J01, and later PVUs, the LIST command allows SERVERNET, DEVICE, and DISC entity specifications using CLIMs. For specification syntax and examples, refer to the SERVERNET (page 302), DEVICE (page 171) and DISC (page 180) entities.
- Measure H04, J02, and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size, if the report output file is a structured file.
- To obtain external records corresponding to a format that you can handle, see TEMPLATE-VERSION (page 143).

Examples

- To display a CPU report, using the NORMAL report format (NORMAL means all counters are displayed):

```
+ LIST CPU 5
Cpu  5 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    491               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        21.46 % Swaps                 0.21
Cpu-Qtime             0.27 # Cpu-Qlen-Max            62 #
Mem-Qtime             0.01 # Mem-Qlen-Max            26 #
Dispatches           31.80   Intr-Busy-Time        0.49 %
Process-Ovhd          0.02 % Send-Busy-Time        0.12 %
Disc-IOs              2.00   Cache-Hits            4.39
Transactions                 Response-Time
Page-Requests                Page-Scans
Ending-Free-Mem       4092   Ending-UCME
Ending-UDS             500   Ending-SDS            1500
Ending-UCL             100   Ending-SCL            2000
```

- To display all CPU reports, using the BRIEF format (BRIEF causes a Measure-defined subset of the counters to be displayed):

```
+ LIST CPU *
Cpu  5 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    491               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        21.46% Swaps                  0.21
Ending-Free-Mem       4092  Ending-UCME
++
Cpu  6 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time       18.42% Swaps                 0.27
Ending-Free-Mem       4092 Ending-UCME              0
++
Cpu  7 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time           5% Swaps                  0.19
Ending-Free-Mem       4092 Ending-UCME              0
```

- To display the same CPU reports in ascending order according to the SWAPS counter:

```
+ LIST CPU *, BY SWAPS (ASCENDING)
Cpu  7 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time           5% Swaps                  0.19
Ending-Free-Mem       4092 Ending-UCME              0
++
```

```
Cpu  5 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    491               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        21.46% Swaps                 0.21
Ending-Free-Mem       4092  Ending-UCME             0
++
Cpu  6 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        18.42% Swaps                 0.27
Ending-Free-Mem       4092  Ending-UCME             0
++
```

- To display only CPU reports in which the CPU-BUSY-TIME value is greater than 5:

```
+ LIST CPU *, IF CPU-BUSY-TIME > 5
Cpu  6 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        18.42% Swaps                 0.27
Ending-Free-Mem       4092  Ending-UCME             0
++
Cpu  5 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    491               Page Size  2048
Local System \BUYER From 22 Oct 1994, 12:20:47 For 118 Secs

Cpu-Busy-Time        21.46% Swaps                 0.21
Ending-Free-Mem       4092  Ending-UCME             0
```

- To display two reports for CPU 1, one of which averages CPU activity over the entire measurement window, and the second of which averages CPU activity over a smaller report window:

```
+ LIST CPU 1
Cpu  1 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    256               Page Size  2048
Local System \BUYER From 5 Nov 1994, 10:29:24 For 3.5 Hours

Cpu-Busy-Time        36.29% Swaps                 0.42
Ending-Free-Mem       4092  Ending-UCME             0

+ LIST CPU 1, FROM 12:30, TO 12:45
Cpu  1 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    256               Page Size  2048
Local System \BUYER From 5 Nov 1994, 12:30:24 For 15 Minutes

Cpu-Busy-Time        15.05% Swaps                 0.05
Ending-Free-Mem       4092  Ending-UCME             0
```

- If you are examining a measurement data file from a remote system, you must use complete specifications when referencing files within the data file. To display a FILE report:

```
+ LIST FILE *
File Open $NEW01.MEAS.QCPU4A
Device Type  3
Opener Process: CPU  3  PIN   37    File Number   1
Local System \BILLS From 24 Oct 1994, 10:01:08 For 0.1 Seconds

Reads                      45.46
Writes
++
File Open \BILLS.03,037
Device Type  0
Opener Process: CPU  3  PIN   39    File Number   2
```

```
Local System \BILLS From 24 Oct 1994, 10:01:08 For 0.2 Seconds

Reads
Writes                        86.28
++
```

- The local system for the measurement was \BILLS. Assuming that you are examining the file on \ACCT, to display the first file of the previous example:

  ```
  + LIST FILE \BILLS.$NEW01.MEAS.QCPU4A
  ```

  To display the second file, you must use the CPU, PIN, and file number because the file name (as listed by MEASCOM) points to a different system:

  ```
  + LIST FILE \BILLS.$*.*.* (3,39,2)
  ```

- To display collected PROCESSH samples in BRIEF format:

  ```
  + RESET REPORT *
  + SET REPORT FORMAT BRIEF
  + LIST PROCESSH *, RATE OFF, IF CODE-RANGE > 0
  Process  1,47                  Pri 168        Device Name $SYSTEM
  Program  $SYSTEM.SYS46.FUP
  Userid   255,255   Creatorid  255,255   Ancestor 1,37 ($Z02M)
  Local System \RAMBLER From  10 Sep 2004, 13:22:08   For  22.2 Seconds
  Total samples =         69 #
  Code-Map  UC.0          Samples          1 #      1.45 % of total
  Code file $SYSTEM.SYS46.FUP:134451329318623
  Code-Map  UC.1          Samples          2 #      2.90 % of total
  Code file $SYSTEM.SYS46.FUP:134451329318623
  Code-Map  NATIVE        Samples         65 #     94.20 % of total
  Code file $SYSTEM.SYS46.OSIMAGE:134451333766514
  Code-Map  NATIVE        Samples          1 #      1.45 % of total
  Code file $SYSTEM.SYS46.OSIMAGE:134451333766514
  ```

- To display collected PROCESSH samples in the default format (with CR-NAME-LEN SHORT, CR-NAME-FORM STANDARD, and CR-NAME-QUAL UNQUALIFIED):

  ```
  + list processh 1,47,by code-range,if code-range > 0
   Process  1,47                  Pri 168        Device Name $SYSTEM
   Program  $SYSTEM.SYS46.FUP
   Userid   255,255   Creatorid  255,255   Ancestor 1,37 ($Z02M)
   Local System \RAMBLER From  10 Sep 2004, 13:22:08   For  22.2 Seconds
   Total samples =         69 #
   Code-Map  UC.0          Samples          1 #      1.45 % of total
   Code file $SYSTEM.SYS46.FUP:134451329318623
                                                        Percent      Percent
   Code-Range Name               Accel        TNS    of Code-Map    of Total
  ------------------------------ ----------- ----------- ----------- ----------
  FILE^HAS^FORMAT1                           1 #               100 %      1.45
   Code-Map  UC.1          Samples          2 #      2.90 % of total

   Code file $SYSTEM.SYS46.FUP:134451329318623
                                                        Percent      Percent
  Code-Range Name               Accel        TNS    of Code-Map    of Total
  ------------------------------ ----------- ----------- ----------- ----------
   START^LARGE^READ                         1 #                50 %      1.45
   READ^BLOCK                               1 #                50 %      1.45
   Code-Map  NATIVE        Samples         65 #      94.20 % of total

   Code file $SYSTEM.SYS46.OSIMAGE:134451333766514
                                                        Percent      Percent
  Code-Range Name                       NATIVE       of Code-Map    of Total
   ------------------------------ ----------- ----------- -----------
   dbio_xsums_validate_                    31 #     47.69 %     44.93 %
   dbio_xsums_generate_                     6 #      9.23 %      8.70 %
   tser__transfer                           3 #      4.62 %      4.35 %
   WAIT                                     3 #      4.62 %      4.35 %
   tip_avtrelease_flags                     2 #      3.08 %      2.90 %
   tip_avtsetup_data                        2 #      3.08 %      2.90 %
  ```

```
LOCKMEMORY                                2 #       3.08 %       2.90 %
searchInitIndex__45NSKOrderedArr          1 #       1.54 %       1.45 %
```

- To display collected PROCESSH samples with demangled names (with CR-NAME-LEN LONG, CR-NAME-FORM DEMANGLED and CR-NAME-QUAL UNQUALIFIED). Code range name "searchInitIndex__45NSKOrderedArr" becomes "NSKOrderedArrayOf<NSKM_MOPSTEntry, unsigned long>::searchInitIndex(unsigned long, unsigned int) const" which is more familiar to the programmer:

```
+ list processh 1,47,by code-range,if code-range > 0
 Process  1,47                 Pri 168        Device Name $SYSTEM
 Program  $SYSTEM.SYS46.FUP
 Userid   255,255   Creatorid  255,255   Ancestor 1,37 ($Z02M)
 Local System \RAMBLER From  10 Sep 2004, 13:22:08   For  22.2 Seconds

 Total samples =         69 #

 Code-Map  UC.0          Samples          1 #      1.45 % of total
 Code file $SYSTEM.SYS46.FUP:134451329318623

                                                        Percent      Percent
 Code-Range Name                Accel        TNS     of Code-Map    of Total
------------------------------ ---------- ---------- ---------- ----------
 FILE^HAS^FORMAT1                        1 #               100 %      1.45
 FILE^HAS^FORMAT1
 Code-Map  UC.1          Samples          2 #      2.90 % of total
 Code file $SYSTEM.SYS46.FUP:134451329318623
                                                        Percent      Percent
  Code-Range Name               Accel        TNS     of Code-Map    of Total
------------------------------ ---------- ---------- ---------- ----------
 START^LARGE^READ                        1 #                50 %      1.45
 START^LARGE^READ
 READ^BLOCK                              1 #                50 %      1.45
 READ^BLOCK

 Code-Map  NATIVE        Samples         65 #     94.20 % of total
 Code file $SYSTEM.SYS46.OSIMAGE:134451333766514

                                                 Percent      Percent
                                                 Percent      Percent
     Code-Range Name               NATIVE     of Code-Map    of Total
-------------------------------- ---------- ---------- ----------
dbio_xsums_validate_                     31 #     47.69 %     44.93 %
dbio_xsums_validate_()
dbio_xsums_generate_                      6 #      9.23 %      8.70 %
dbio_xsums_generate_()
tser__transfer                            3 #      4.62 %      4.35 %
tser__transfer()
WAIT                                      3 #      4.62 %      4.35 %
WAIT()
tip_avtrelease_flags                      2 #      3.08 %      2.90 %
tip_avtrelease_flags()
tip_avtsetup_data                         2 #      3.08 %      2.90 %
tip_avtsetup_data()
LOCKMEMORY                                2 #      3.08 %      2.90 %
LOCKMEMORY()
searchInitIndex__45NSKOrderedArr          1 #      1.54 %      1.45 %
NSKOrderedArrayOf<NSKM_MOPSTEntry
, unsigned long>::searchInitIndex(unsigned long
, unsigned int) const
```

LIST EXTNAMES

In Measure G11 and later PVUs, this command creates or appends to a key-sequenced file of records for mapping:

- Measure OSS PATHID values to OSS pathnames and Guardian file names
- MIDs to ANSI SQL names and Guardian file names

The file EXTNAMES contains EXTNAME name records and is assumed to be located in the current default subvolume.

This command has no options or qualifiers. It can be issued only within the context of the REPORT FORMAT STRUCTURED option.

Syntax

```
SET REPORT FORMAT STRUCTURED

LIST EXTNAMES
```

DDL Record for EXTNAMES file

The DDL record for EXTNAMES, defined in MEASDDLS, is:

```
RECORD extname. FILE is "extnames" KEY-SEQUENCED.
 02 MID.
    03 PATHID                  type character 24.
    03 CRVSN                   type character 6.
 02 FILE-SYSTEM-NAME           type character 8.
 02 FILE-NAME.
    03 VOLUME                  type character 8.
    03 SUBVOL                  type character 8.
    03 FILENAME                type character 8.
 02 NAME-TYPE                  type binary 16 unsigned.
 02 FULL-NAME-LEN              type binary 16 unsigned.
 02 FULL-NAME                  type character 1100.
end
```

DDL Record Description Fields

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

FILE-NAME-MID

A Measure identifier for an OSS file or SQL/MX file. For OSS and SQL/MX files, the FILE-NAME-MID can be used as a key to the EXTNAMES file to retrieve the corresponding Guardian file name and external format OSS pathname or ANSI SQL name. FILE-NAME-MID has two subfields: PATHID and CRVSN.

- PATHID is an internal format representation of an OSS file or SQL/MX file.
- CRVSN is a creation version serial number that identifies a unique instance of an OSS or SQL/MX file.

FILE-SYSTEM-NAME

The EXPAND system name of a file described by *file-name* if the file is located on a node other than that measured in the data file. Otherwise, this field contains spaces.

FILE-NAME

The Guardian local-internal-format file name for the OSS file or SQL/MX file. Also known as the *gname* of the file.

NAME-TYPE

Name types are: 0 - not used, 1 - OSS, 2 - ANSI SQL.

FULL-NAME-LEN

The length, in bytes, of the external-format OSS pathname or ANSI SQL name stored in the FULL-NAME field. The maximum value is 1100 in Measure G11 and later PVUs.

FULL-NAME

The left-justified external-format OSS pathname or ANSI SQL name. The field is defined as 1100 characters, but the length of the actual name is reported in FULL-NAME-LEN. Remaining characters are undefined.

Example

To create an EXTNAMES structured output file:

```
+ SET REPORT FORMAT STRUCTURED
+ LIST EXTNAMES
```

LIST OSSNAMES

In Measure G09 and later PVUs, this command creates or appends Guardian file names or Measure MID values to their OSS file pathnames equivalent. The key-sequenced file is named OSSNAMES and is assumed to be in the current default subvolume.

The LIST OSSNAMES command has no options or qualifiers. The command can only be issued following the REPORT FORMAT STRUCTURED command.

In Measure G11 and later PVUs, LIST OSSNAMES is still available but is superseded by LIST EXTNAMES (page 78), which supports both OSS file pathnames and ANSI SQL names.

Syntax

```
LIST OSSNAMES
```

Usage Notes

This warning or error message might be displayed:

```
3114 OSS journal segment is required in this context and is not available for access.
```

DDL Record for OSSNAMES file

The DDL record for OSSNAMES, defined in MEASDDLS, is:

```
RECORD OSSNAMES.  FILE is "OSSNAMES" KEY-SEQUENCED.
 02 MID.
    03 PATHID                  type character 24.
    03 CRVSN                   type character 6.
 02 file-system-name           type character 8.
 02 file-name.
    03 volume                  type character 8.
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 pathname-len               type binary 16 unsigned.
 02 pathname                   type character 1024.
*    key is MID duplicates not allowed
end
```

Example

To create an OSSNAMES structured output file:

```
+ SET REPORT FORMAT STRUCTURED
+ LIST OSSNAMES
```

LIST PLOT

This command displays the currently designated plot, as specified by one or more ADD PLOT commands.

In Measure G11 and later PVUs, ANSI SQL names appear in plots.

Syntax

```
LIST [ / OUT filename / ] [ PLOT ]
   [ , FROM [ start-date, ] start-time-of-day ]
   [ , FOR duration ]
   [ , TO [ stop-date, ] stop-time-of-day ]
   [ , INTERVAL interval ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* exists, MEASCOM opens the file and appends command output to the end of it.

After the LIST PLOT command executes, MEASCOM redirects its output to the current OUT file (typically, the terminal). This option does not affect the contents of the log file.

PLOT

is optional if you specified PLOT as the command object by using the ASSUME command.

FROM [*start-date*,] *start-time-of-day*

specifies the start of the plot window. If you omit FROM, the start time of the measurement or of the entity measurement is used, whichever is later.

start-date

is the date on which the plot window starts, in one of these formats:

{ [d]d mmm[yyyy] }
{ mmm [d]d[yyyy] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *start-date*, the start date of the measurement or of the measurement of the entity is used, whichever is later.

start-time-of-day

is the time the plot window starts, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *start-time-of-day*, the start time of the measurement is used.

FOR *duration*

specifies the duration of the plot window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement or the entity measurement ends, whichever is earlier.

duration

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

INTERVAL *interval*

specifies the time interval for the time line in the plot.

interval

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999 and a multiple of the collection interval specified when the data was collected.

If you omit INTERVAL, MEASCOM uses the collection interval specified when the data was collected.

SCALE-FROM *number*

sets the lower boundary of the data-value axis.

number

is a number in the range 0 through 999999999999.999. The default is 0.

SCALE-TO *number*

sets the upper boundary of the data-value axis.

number is a number in the range 0 through 999999999999.999. The default is 100.

TIME-BASE { OFF | ON }

displays a bar graph or a two-axis plot of counter values. If a time interval was specified when the data was collected, the default is TIME-BASE ON. Otherwise, the default is TIME-BASE OFF.

OFF

displays a bar graph of average counter values.

ON

displays a plot of counter values over time.

TO [*end-date* ,] *end-time-of-day*

specifies the end of the plot window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement or the entity measurement ends, whichever is earlier.

end-date

is the end date of the data used for the plot, in the same format as *start-date*.

end-time-of-day

is the end time of the data used for the plot, in the same format as *start-time-of-day*.

VERT-BASE { OFF | ON }

controls the display of the time axis. The default is ON.

OFF

for a two-axis plot, displays time on the horizontal axis. For a bar graph, displays the bars vertically (base on the horizontal axis).

ON

for a two-axis plot, displays time on the vertical axis. For a bar graph, displays the bars horizontally (base on the vertical axis).

WIDE-ITEM { OFF | ON }

sets the density of the plot. The default is OFF.

OFF

for a two-axis plot, displays a single plot character for each counter value at each time interval. For a bar graph, displays bars one character wide.

ON

for a two-axis plot, fills in the area between the time axis and the lowest counter value. For a bar graph, displays bars two characters wide to six characters wide depending on the number of counter values on the graph.

Related Commands

Command	Function	Page
ADD PLOT	Adds a counter to the current plot description	48
DELETE PLOT	Deletes a counter from the current plot description	55
INFO PLOT	Displays the current plot description	67
SET PLOT	Specifies the plot format	67

Examples

- To display a plot in the default format, assuming that the measurement data was collected using a collection interval of 4 seconds:

```
+ LIST CPU *
      .
      .   (reports for CPUs appear here)
      .
+ ADD PLOT CPU-BUSY-TIME
+ LIST PLOT

         0:::::+:::20.0:::::+:::40.0:::::+:::60.0 ...
14:28:19 -                   B          A
14:28:23 - B       A
14:28:27 - B      A
14:28:31 -B                        A
14:28:35 - B        A
14:28:39 -B   A
14:28:43 -B              A
14:28:47 - B               A
14:28:51 -B   A
14:28:55 -B   A
14:28:59 -B   A
14:29:03 -B   A
14:29:07 -          B                          A
         0:::::+:::20.0:::::+:::40.0:::::+:::60.0 ...

      Min. value =   2.000                Max value =  88.000

      A: CPU-BUSY-TIME                 Cpu 6
      B: CPU-BUSY-TIME                 Cpu 7
```

- To display a plot in the default format, assuming that the measurement data was collected without a collection interval:

```
+ LIST CPU *
      .
      .   (reports for CPUs appear here)
      .
+ ADD PLOT CPU-BUSY-TIME
+ LIST PLOT

         0:::::+:::20.0:::::+:::40.0:::::+:::60.0 ...
```

```
 A ************|
 B ***|
   0:::::+:::20.0:::::+:::40.0:::::+:::60.0 ...

Min. value =   2.000                Max value =  88.000

A: CPU-BUSY-TIME                  Cpu 6
B: CPU-BUSY-TIME                  Cpu 7
```

- This example displays a PROCESSH plot that shows how many samples of accelerated code and TNS code were measured. The asterisk (*) represents accelerated code, and the plus sign (+) represents TNS code. The plot also shows the results for TNS code plotted separately.

```
+ LIST PROCESSH $SYSTEM.SYS02.MEASCOM (UC.0)
      .
      .   (report for PROCESSH entity appears here)
      .
+ ADD PLOT CODE-RANGE
+ ADD PLOT TNS-BUSY-SAMPLES
+ SET PLOT TIME-BASE OFF

+ LIST PLOT

     0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+:::80.0:::+::::100
    A ***************************+++++++++++++++++++|                    -
    B ********************************++|                               -
    C ************************************************************+++++++++-
      ++++++++++++++++++++++|                                            -
    D +++++++++++++++|                                                   -
    E +                                                                  -
    F ++++++++++++++++++++++++++++++++++|                                -
     0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+:::80.0:::+::::100

      Min. value =   2.000               Max value =  88.000

    A: FORMAT          ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
    B: PRINT^T RESULT  ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
    C: FIND^T          ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
    D: FORMAT          ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
        (TNS)
    E: PRINT^T RESULT  ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
        (TNS)
    F: FIND^T          ProcessH $SYSTEM.SYSO2.MEASCOM (7,37)
        (TNS)
```

- To display a PROCESSH plot that shows how many samples of TNS/R native code were measured (the pound symbol (#) represents TNS/R native code):

```
+ LIST PROCESSH $SYSTEM.SYS02.TSYSDP2 (UCR)
      .
      .   (report for PROCESSH entity appears here)
      .

+ ADD PLOT TNSR-BUSY-TIME
+ ADD PLOT TIME-BASE OFF
+ LIST PLOT

     0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+:::80.0:::+::::100
    A ##########################################|                       -
    B ###################################|                              -
    C #|                                                                 -
     0:::::+:::20.0:::::+:::40.0:::::+:::60.0:::::+:::80.0:::+::::100

      Min. value =   2.000               Max value =  65.000

    A: AC1^CKPT^GET^BLOCK            ProcessH $SYSTEM.SYSO2.TSYSDP2 (7,20)
       (TNSR)
    B: AC1^RECMOVE^DATA              ProcessH $SYSTEM.SYSO2.TSYSDP2 (7,20)
       (TNSR)
    C: AC1^REDCHAIN^PURGE            ProcessH $SYSTEM.SYSO2.TSYSDP2 (7,20)
       (TNSR)
```

LISTACTIVE *entity-type*

This command assembles data from the active measurement and displays a report for each entity included in the *entity-spec* parameter.

A double prompt (++) at the bottom of the screen signals that another report follows the one currently on the screen. Press Return (or any other key) to view the next report. Press Ctrl-Y or type BREAK at the double prompt to ignore subsequent reports:

- If you entered multiple commands on the previous command line, BREAK interrupts the current command and executes the next command.
- If you did not enter multiple commands, BREAK interrupts the current command and returns the MEASCOM prompt.

The value in an active counter record reflects the activity of the counter from the time it was initialized:

- For a CPU entity, the system initializes the active counter record when the CPU is loaded.
- For an entity of any other type, the Measure subsystem initializes the active counter record when the measurement starts.
- If multiple measurements include the same entity, all measurements use the same active counter record. The first measurement started initializes the counter record.

Syntax

```
LISTACTIVE [ / OUT filename / ] [ entity-type ]
   entity-spec [ , list-option ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the LISTACTIVE command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal). This option does not affect the log file contents.

entity-type

is one of:

```
CLUSTER      DISKFILE   OSSCPU      SQLPROC    USERDEF
CONTROLLER   FILE       OSSNS       SQLSTMT
CPU          LINE       PROCESS     SYSTEM
DEVICE       NETLINE    PROCESSH    TERMINAL
DISC         OPDISK     SERVERNET   TMF
```

The *entity-type* keyword is optional if you specified it as the command object by using the ASSUME command.

LISTACTIVE has special entity specification syntax for the FILE, LINE, NETLINE, PROCESS, SQLPROC, USERDEF, and TERMINAL entities. For details, see LISTACTIVE Entity Specification Special Cases (page 87).

NOTE: For a LISTACTIVE command, an asterisk (*) cannot be used for the DEVICE or DISK entities; a specific device or disk must be named.

entity-spec

identifies the entity to be measured. The identifiers you use are specific to each entity type. For identifier syntax, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).

NOTE: Because LISTACTIVE uses entity-spec to access the entity control block directly (which in turn is used to access the counter record in system counter space), *entity-spec* must specify only one entity and must indicate the processor in which it resides.

Thus, DISKFILE accepts ANSI SQL partition names but not ANSI SQL object names.

list-option

is one of:

DOTS { ON } { OFF }

sets whether report displays include connecting dots between labels and numeric values. Valid only if STYLE is ZMS. The default is OFF.

ON

displays use dots to connect labels to formatted numbers.

OFF

displays do not use dots to connect labels to formatted numbers.

FOR *duration*

for each field in the record, displays the difference between the immediate value and the value after *duration*. If FOR is omitted, the field values reflect activity since the measurement began.

duration is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

FORMAT { BRIEF } { NORMAL } { STRUCTURED }

sets the format of the report. The default is NORMAL.

BRIEF

displays an abbreviated report that contains the most commonly used counters for each entity type.

For information on which counters are included in brief-format reports, see the DDL record for the specified entity type in Chapter 3: Entities and Counters (page 133). Counters included in brief-format reports are in **boldface type** in each DDL record.

NORMAL

displays all counters.

STRUCTURED

writes the report to a structured file for subsequent examination using Enform. A single record consists of all counters for a single entity. (For DDL record format, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).)

Reports for entities of the same entity type are written to the same file, and the file is named for the entity type. If the file already exists, MEASCOM appends the data to the file. If the file does not exist, MEASCOM creates it. (Structured files are closed when you modify the FORMAT attribute.)

LOADID *loadid*

specifies the name to be placed in the *loadid* field of the records generated by this command. The LOADID option applies only to structured reports.

loadid is an alphanumeric string, 1 through 8 characters long. The string can contain letters, numbers, carets (^), hyphens (-), and underscores (_). The first character must be a letter.

RATE { OFF | ON }

determines how counter values are displayed. The RATE attribute has no effect on structured files. The default is ON.

OFF

displays uninterpreted counter values.

ON

displays interpreted counter values (counts per second and percent busy).

STYLE { LEGACY } { ZMS }

sets whether displays or structured data are formatted using the ZMS style interface or using the legacy interface compatible with G10 and earlier Measure PVUs. In G-series RVUs, the default is LEGACY; in H-series RVUs, the default is ZMS.

LEGACY

displays and structured records use the legacy style.

ZMS

displays and structured records use the ZMS style.

ZERO-REPORTS { INCLUDE | SUPPRESS }

determines whether records containing all zero values are displayed. The default is SUPPRESS.

INCLUDE

displays entity reports even if all counter values are zero.

SUPPRESS

does not display entity reports if all counter values are zero.

ZERO-VALUES { INCLUDE | SUPPRESS }

determines whether values less than 0.005 are displayed as zeros or blanks. The default is SUPPRESS.

INCLUDE

displays zeros if counter values are less than 0.005.

SUPPRESS

displays blanks if counter values are less than 0.005.

The ZERO-VALUES attribute has no effect on structured files.

LISTACTIVE Entity Specification Special Cases

The LISTACTIVE command has special entity specification syntax for several entities.

DEVICE and DISC

```
LISTACTIVE [ DEVICE | DISC ] [ device-spec [ , listactive-spec]
```

device-spec

identifies the device or disc to measure, specified as:

Non-CLIM syntax:

```
{ devicename ( cpu , svnet , group , module , slot [, scsi-id ] )
}
```

CLIM syntax:

```
{ devicename ( cpu , svnet , sac , lun ) }
```

Syntax for disc only:

```
{ devicename-path ( cpu ) }
```

NOTE: For a LISTACTIVE command, you must name specific devices to be measured; a wildcard (*) cannot be used.

Examples

```
FCSA disk:   LISTACTIVE DISC $SYSTEM (0,X,110,3,1)
             LISTACTIVE DISC $SYSTEM (0,X,110,3,1,101)
CLIM disk:   LISTACTIVE DISC $SYSTEM (0,X,C1002531,201) Path option: LISTACTIVE DISC $SYSTEM-P (0)
```

FILE

```
LISTACTIVE [ FILE ] [ file-spec [ , listactive-spec]
```

file-spec

identifies the file to measure, specified as:

filename (*cpu*, *pin*, *filenum*)

where *cpu*, *pin* is the PID of the opener, and *filenum* is in the range 0 through 255.

LINE and NETLINE

```
LISTACTIVE [ LINE ] [ WAN-spec [ , listactive-spec]
```

WAN-spec

identifies the line to measure, specified as one of:

{ cpu (line-name) }
{ cpu (trackid , clip , line) }

line-name

is the name of the line, in the form $*name7*, where *name7* is up to seven characters.

cpu

is the number of the processors, in the range 0 through 15. The default is all CPUs.

trackid

is the number of the track, in the form %H*hexnum6*, where *hexnum6* is up to six characters. The default is all track IDs.

clip

is the number of the CLIPs, in the range 1 through 6. The default is all CLIPs.

line

is the number of the lines, either 0 or 1. The default is both lines.

PROCESS, SQLPROC, and USERDEF

```
LISTACTIVE [ PROCESS ] [ process-spec [ , listactive-spec]
```

process-spec

identifies the process to measure, specified as one of:

{ *cpu*,*pin* }
{ $*process-name* (*cpu*,) }
{ *disk-filename* (*cpu*, *pin*)

where *cpu* is in the range 0 through 15, and *pin* is in the range 0 through 65534.

TERMINAL

```
LISTACTIVE [ TERMINAL ] [ terminal-spec [ , listactive-spec]
```

terminal-spec

identifies the terminal to measure, specified as:

cpu (*line-name.subdevice*)

cpu is the number of the processor, in the range 0 through 15.

line-name is the name of the line, specified as: { $*name7* | $*num* | *logical-Define-name* }

subdevice is the name of the subdevice, specified as: { #*name7* }

Usage Notes

- You can use LISTACTIVE to read the active records of DISKFILE entities. When you use the command LISTACTIVE DISKFILE, the disk file spec must be a valid and unique disk file name. MEASCOM expands partially specified file names with current SYSTEM and VOLUME defaults.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).
- LISTACTIVE cannot be used to read DISCOPEN or PROCESSH counters:
  - The DISCOPEN record is unavailable to LISTACTIVE because you have no way to uniquely determine the Open Control Block (OCB) number of an active file. MEASCTL needs the OCB to locate the record.
  - PROCESSH records are unavailable because the complexity of the measurement could degrade system performance.

  If you have specified a collection interval, you can use the LIST command to read DISCOPEN and PROCESSH counters from a currently active data file. When you use a collection interval, data is collected in an internal buffer and written to the data file when the buffer is full. One or more collection intervals might pass before data is written to the file.
- The FILE and PROCESS entities display the names of open files. Because a process can open resources other than files (such as a process or $0), the reports for these entities sometimes include file names that are invalid according to Measure syntax. To display such reports, specify the CPU, PIN, and file number of the resource rather than its file name. See the examples under LIST *entity-type* (page 68).
- Information displayed by the LISTACTIVE command can differ from information displayed by the LIST command:
  - For a LISTACTIVE command, MEASCOM requests data from the Measure control process (MEASCTL) in the respective CPU. MEASCTL reads the requested data from the active counter record in system data space.
  - For a LIST command, MEASCOM requests data from the Measure file-handling process (MEASFH) for the respective data file. MEASFH uses information stored in the data file to determine the activity of the counter during the measurement window (or during the report window in the LIST command).
- In Measure H01 and later PVUs, the LISTACTIVE command allows DISKFILE, FILE, and SQLSTMT entity specifications using ANSI SQL names. For the specification syntax, see DISKFILE (page 207), FILE (page 216) or SQLSTMT (page 319).
- In Measure H03, J01, and later PVUs, the LISTACTIVE command allows SERVERNET, DEVICE, and DISC entity specifications using CLIMs. For specification syntax and examples, refer to the SERVERNET (page 302), DEVICE (page 171) and DISC (page 180) entities.
- Measure H04, J02, and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size, if the report output file is a structured file.

Related Commands

Command	Function	Page
LIST	Reads counter values from a data file	68
SET REPORT	Specifies a default report format	108

Examples

- To display information about the disk $NEW01 on a NonStop K-series server:

```
+ SET REPORT FORMAT BRIEF
+ LISTACTIVE DISC $NEW01 (2,0,25,2)
Disc $NEW01 (4160 -- 895MB)   PID 2,8   Ldev 5   C,C,U (0,%25,%2)
Local System \BILLS From 18 Oct 1994, 14:35:33 For 113 Seconds

Disc-Busy-Time        39.53 %    Disc-Rate        16.04
Request Qtime          0.62 #    Request-Qlen-Max     3 #
```

- To display information about the disk $DATA01 on a NonStop S-series server:

```
+ LISTACTIVE DISC $DATA01 (X, DISK-DEVICE-14)
Disc $DATA01      (4160 -- 4GB)   Type 3  Subtype 23 PID 0,287
Cfg:    DISK-DEVICE-14
SAC:   CONTROLLER-2
GMS  1,4,402   SvNet X   Logical Device 34
Local System \ACCT   From 28 Oct 1995, 16:43:24  For 179 seconds

Disc-Busy-Time          43.51 %  Disc-Rate           13.24
Request-Qtime              0.42 #  Request-Qlen-Max        2#
```

- This example displays a report for CPU 1. The CPU record is initialized at system load, so the measurements reflect CPU activity from system load to when the report was generated.

```
+ LISTACTIVE CPU 1
Cpu  1 VLX          Initial Lock Pgs 2048      Mem Pages  8192
Memory MB 16        PCBs    200                Page Size  2048
Local System \BILLS From 10 Oct 1994, 18:45:18 For 187 Hours

Cpu-Busy-Time        0.67%   Swaps                0.01
Cpu-Qtime            0.01#   Cpu-Qlen-Max          13#
Mem-Qtime                    Mem-Qlen-Max           4#
Dispatches           5.63    Intr-Busy-Time       0.35%
Process-Ovhd                 Send-Busy-Time
Disc-IOs             0.02    Cache-Hits           0.01
Transactions                 Response-Time
Page-Requests                Page-Scans
Ending-Free-Mem      4092    Ending-UCME
Ending-UDS            500    Ending-SDS            1500
Ending-UCL            100      Ending-SCL            2000
```

LISTALL *entity-type*

This command reads data from the current data file (the file most recently specified in an ADD MEASUREMENT command) and displays a report for each entity included in the *entity-spec* parameter.

The LISTALL command is similar in function and syntax to the LIST *entity-type* command. However, LISTALL differs in that it lists each interval record within the specified measurement window.

A double prompt (++) signals that another report follows the one on the screen. Press Return (or any other key) to view the next report. Press Ctrl-Y or type BREAK at the double prompt to ignore subsequent reports:

- If you entered multiple commands on the previous command line, BREAK causes the current command to be interrupted and the next command to be executed.
- If you did not enter multiple commands, BREAK interrupts the current command, and the MEASCOM prompt returns.

Syntax

```
LISTALL [ / OUT filename / ] [ entity-type ] entity-spec
    [ , list-option ] ...
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

After the LISTALL command executes, MEASCOM resumes writing its output to the current OUT file (typically, the terminal). This option does not affect the contents of the log file.

entity-type

is one of:

```
CLUSTER     DISCOPEN   OPDISK     SERVERNET    TMF
CONTROLLER  DISKFILE   OSSCPU     SQLPROC      USERDEF
CPU         FILE       OSSNS      SQLSTMT
DEVICE      LINE       PROCESS    SYSTEM
DISC        NETLINE    PROCESSH   TERMINAL
```

The *entity-type* keyword is optional if you specified it as the command object by using the ASSUME command.

entity-spec

identifies the entity to be measured. The identifiers you use are specific to each entity type. For identifier syntax, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).

list-option

is one of:

BY *item-name* [(ASCENDING) | (DESCENDING)]

sorts the report in ascending or descending order.

item-name

is one of:

{ *counter* }
{ *identification-item* }

where

counter is a counter name. By default, counter items are sorted in descending order. Special considerations exist for the PROCESSH, DISC, and CPU entity types:

- For PROCESSH, *counter* is a procedure name.
- For DISC cache counters, *counter* must be preceded by C0-, C1-, C2-, or C3- to specify the size of the cache blocks to be considered.
- For CPU, if a counter is not specified, the CPUs are listed in numeric order.

identification-item is an identification item such as CPU-NUM or PROCESS-NAME. You can use any valid identification item for the specified entity. By default, identification items are sorted in ascending order.

CR-NAME-LEN { SHORT | LONG }

controls the displayed length of procedure (code-range) names. The default is SHORT.

SHORT

truncates procedure (code-range) names (if necessary) at 32 characters. Mangled procedure names are not demangled prior to display.

LONG

displays procedure (code-range) names both in their SHORT form and, if longer than 32 characters, in their entirety on a subsequent line. Mangled procedure names are not demangled for display.

CR-NAME-FORM { STANDARD | DEMANGLED | BOTH }

controls whether demangled procedure (code-range) names are displayed (if applicable). The default is STANDARD.

STANDARD

displays procedure (code-range) names in the form specified in the code file (mangled) or EDIT file (demangled).

DEMANGLED

demangles procedure (code-range) names, if necessary, prior to display.

BOTH

displays procedure (code-range) names in both STANDARD and DEMANGLED forms.

CR-NAME-QUAL { UNQUALIFIED | QUALIFIED }

controls whether procedure (code-range) names are displayed with object file name qualifiers, if available, in the Code-Range Name column on the line immediately following the traditional line of code-range output. The default is UNQUALIFIED.

UNQUALIFIED

displays procedure names in traditional form with no qualifiers.

QUALIFIED

displays procedure names together with the Guardian or OSS object file name of the associated code. This can be useful in differentiating between like-named procedures in different object files. Guardian file names have their associated CRVSN appended to the end of the name.

DOTS { ON } { OFF }

specifies whether report displays include connecting dots between labels and numeric values. Valid only if STYLE is ZMS. The default is OFF.

ON

displays dots to connect labels to formatted numbers.

OFF

does not display dots to connect labels to formatted numbers.

FOR *duration*

specifies the duration of the report window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement ends or when measurement of the entity ends, whichever is earlier.

duration

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

FORMAT { BRIEF } { NORMAL } { STRUCTURED }

sets the format of the report. The default is NORMAL.

BRIEF

displays an abbreviated report that contains the most commonly used counters for each entity type. For information on which counters are included in brief-format reports, see the DDL record for the specified entity type in Chapter 3: Entities and Counters (page 133). Counters included in brief-format reports are in **boldface type** in each DDL record.

NORMAL

displays all counters.

STRUCTURED

writes the report to a structured file for subsequent examination using Enform. A single record consists of all counters for a single entity. (For DDL record format, see the description of the specified entity type in Chapter 3: Entities and Counters (page 133).)

Reports for entities of the same entity type are written to the same file, and the file is named for the entity type. If the file already exists, MEASCOM appends the data to the file. If the file does not exist, MEASCOM creates it. (Structured files are closed when you modify the FORMAT attribute.)

FROM [[*start-date*,] *start-time-of-day*]

specifies the start of the report window. If you omit FROM, the start time of the measurement or of the measurement of the entity is used, whichever is later.

start-date

is the date on which the report window starts, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year (1984 through 2047).

If you omit *date*, the start date of the measurement is used.

start-time-of-day

is the time the report window starts, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *start-time-of-day*, the start time of the measurement is used.

IF { *item-name operation value* }

determines which data records are included in the report. Only records that meet the specified condition are included.

item-name

is one of:

{ *counter* }

{ *identification-item* }

where

counter is a counter name.

- For the PROCESSH entity type, *counter* is a procedure name.
- For the cache counters of the DISC entity type, *counter* must be preceded by C0-, C1-, C2-, or C3- to specify the size of the cache blocks to be considered.

identification-item is a numeric entity identification item of INT or INT(32) type. To determine which identification items can be used, see the MEASDDLS file or the descriptions of each entity descriptor in MEASCONFIGURE (page 358). Character string items such as PROCESS-NAME and fixed-length items such as FROM-TIMESTAMP cannot be used in the IF clause.

operation

is one of:

> (greater than)
< (less than)
= (equal to)
<> (not equal to)

value

is a number in the range 0 through 2147482.999. From Measure G09 and later PVUs the range is 0 through 999999999999.

LOADID *loadid*

specifies the name to be placed in the *loadid* field of the records generated by this command. The LOADID option applies only to structured reports.

loadid

is an alphanumeric string, 1 through 8 characters long. The string can contain letters, numbers, carets (^), hyphens (-), and underscores (_). The first character must be a letter.

When used in the LISTALL command, LOADID should not exceed five characters. Leave three blank characters on the right. The LISTALL command inserts the interval sequence number (for example, 001, 002, 003, ...) in these three right-hand character positions.

STYLE { LEGACY } { ZMS }

sets whether displays or structured data are formatted using the ZMS style interface or using the legacy interface compatible with pre-G11 Measure PVUs. In G-series RVUs, the default is LEGACY; in H-series RVUs, the default is ZMS.

LEGACY

displays and structured records use the legacy style.

ZMS

displays and structured records use the ZMS style.

TO [*end-date*,] *end-time-of-day*

specifies the end of the report window. FOR and TO are mutually exclusive. If you omit both, the window ends when the measurement ends or when measurement of the entity ends, whichever is earlier.

end-date

is the date on which the report window ends, in the same format as the *start-date*.

If you omit *end-date*, the end date of the measurement is used.

end-time-of-day

is the end time of report window, in the same format as the *start-time-of-day*.

If you omit *end-time-of-day*, the end time of the measurement is used.

TOLERANCE { ON | OFF }

is the tolerance to apply in deciding which measurements to include in the report:

- TOLERANCE OFF applies the FROM and TO limits exactly as specified.
- TOLERANCE ON (the default) interprets the FROM and TO limits as a range bounded by plus or minus one-half of the measurement interval. For example, if the specified FROM time is 8:00 and the measurement interval is 30 minutes, the actual FROM time can be as early as 7:45. Similarly, if the specified TO time is 10:00, the actual TO time can be as late as 10:15.

TOTALS { INCLUDE | ONLY | SUPPRESS }

specifies whether TOTALS are displayed. The default is SUPPRESS.

INCLUDE

indicates that both the per-process and aggregated totals are displayed. When aggregated totals are displayed, the set of totals is enclosed in brackets ([…]), and the number of processes that were executing the code being totaled precedes the individual code-range results, also enclosed in brackets (that is, [Totals across 3 processes]).

ONLY

For all entities except PROCESSH and USERDEF, displays only the final TOTALS report. If only one entity report is generated, the TOTALS report is not displayed.

For the PROCESSH entity, only aggregated PROCESSH data is displayed.

NOTE: PROCESSH data can be collected on either a per-code-file basis or a per-procedure basis, dependent on whether a *code-file-spec* was supplied when the PROCESSH measurement was configured (via the ADD PROCESSH command). Thus, the displayed totals can cover an entire code file or each of a set of code ranges within the code file.

SUPPRESS

displays only the entity reports. (The TOTALS attribute is ignored if a report contains only measurements for one entity.)

For PROCESSH, SUPPRESS indicates only per-process data is displayed (Aggregated data is not displayed.)

SUPPRESS has no effect on USERDEF reports.

ZERO-REPORTS { INCLUDE | SUPPRESS }

determines whether records containing all zero values are displayed. The default is SUPPRESS.

INCLUDE

displays entity reports even if all counter values are zero.

SUPPRESS

does not display entity reports if all counter values are zero.

ZERO-VALUES { INCLUDE | SUPPRESS }

determines whether values less than 0.005 are displayed as zeros or blanks. The default is SUPPRESS.

INCLUDE

displays zeros if counter values are less than 0.005.

SUPPRESS
displays blanks if counter values are less than 0.005.

The ZERO-VALUES attribute has no effect on structured files.

Usage Notes

- If a measurement has no intervals or if *entity-type* is PROCESSH, LISTALL produces the same result as the LIST *entity-type* command.
- The LISTALL command can be used only with data file versions C20 and later.
- If the STRUCTURED report option is used (to send the report to a structured file), these list options are not allowed:

  ```
  BY item-name
  RATE ON
  ```

- In G-series RVUs, IF clauses cannot refer to fields that appear only in ZMS style records and reports, even in the ZMS style report mode.
- In H-series RVUs, IF clauses cannot refer to fields that appear only in legacy style records and reports, even in the legacy style report mode.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).
- In Measure H01 and later PVUs, the LISTALL command allows DISCOPEN, DISKFILE, FILE, and SQLSTMT entity specifications using ANSI SQL names. For the specification syntax, see DISCOPEN (page 198), DISKFILE (page 207), FILE (page 216), or SQLSTMT (page 319).
- In Measure H03, J01, and later PVUs, the LISTALL command allows SERVERNET, DEVICE, and DISC entity specifications using CLIMs. For specification syntax and examples, refer to the SERVERNET (page 302), DEVICE (page 171) and DISC (page 180) entities.
- Measure H04, J02, and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size, if the report output file is a structured file.
- Effective with Measure H04, J02, and later PVUs for the CPU and PROCESS entities, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the NATIVE-BUSY-TIME, ACCEL-BUSY-TIME and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Example

Assume these commands to establish hourly intervals for measuring CPU activity:

```
+ ADD CPU *
+ START MEASDATA, INTERVAL 1 HOUR, FROM 8:00, TO 14:00
```

This LISTALL command yields three records, showing the activity of CPU 0 during the intervals 9:00-10:00, 10:00-11:00, and 11:00-12:00:

```
+ LISTALL CPU 0, FROM 9:00, TO 12:00
```

In contrast, this LIST command yields a single record, showing the activity of CPU 0 from 9:00-12:00:

```
+ LIST CPU 0, FROM 9:00, TO 12:00
```

LISTENAME

In Measure G11 and later PVUs, this command translates a Guardian file name and CRVSN into its corresponding external name (ANSI SQL name or OSS pathname). In cases of file name reuse, the CRVSN distinguishes a specific instance of a Guardian file name.

Syntax

```
LISTENAME [ / OUT filename / ] guardian-name[:crvsn]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends the command output to it.

guardian-name

is the Guardian file name to be translated.

crvsn

is the creation version serial number associated with a specific instance of a Guardian file name. To obtain a CRVSN value, view the MEASCOM entity display. The CRVSN is appended to the Guardian file name using a colon.

Example

This example shows a Guardian file name translated to an ANSI SQL name:

```
+ LISTENAME $DATA01.ZSD12345.Z1234567:340359
$DATA.ZSD12345.Z1234567:340259 'TABLE CATALOG_12.SCHEMA_34.TABLE_56
PARTITION PARTITION_78'
```

LISTGNAME

The LISTGNAME command can translate an OSS file pathname (Measure G09 and later PVUs) or an ANSI SQL name (Measure H01 and later PVUs) to its corresponding Guardian file name (*gname*) and creation version serial number (*CRVSN*). In the case of file name reuse, *CRVSN* identifies a specific instance of a Guardian file name.

If the OSS file pathname specified identifies a directory, the contents of the directory are listed. Subdirectories of a directory are listed by their OSS file pathname, and files are listed by their Guardian file name and fully qualified OSS file pathname.

The OSS file pathname translation is affected by the current MEASCOM session and the type of data files being added or deleted. These examples describe how this pathname translation is determined:

Basis for OSS File Pathname Translation	MEASCOM Session History
OSS file-system interfaces	MEASCOM session is not performing an ADD MEASUREMENT command. or MEASCOM session is performing an ADD MEASUREMENT command for an active data file or a data file from the current system that does not contain an OSS journal segment.
Content of the OSS journal	MEASCOM session is performing an ADD MEASUREMENT command for a data file that contains an OSS journal segment.
Cannot be determined	MEASCOM session is performing an ADD MEASUREMENT command for a data file that does not contain an OSS journal segment, and the data file is not from the current node. An error message appears.

Syntax

```
LISTGNAME [ / OUT filename / ] name
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends the command output to it.

"*name*"

can be a fully qualified and valid OSS file pathname or a partial OSS file pathname (in double quotes), or a fully qualified ANSI SQL name.

To get a fully qualified and valid OSS file pathname, combine *name* with the current OSSPATH setting. If the OSS file pathname does not begin with a backslash (/), expand it by putting the current setting for OSSPATH before the backslash.

A fully qualified ANSI SQL name requires either a TABLE or an INDEX, optionally followed by a PARTITION.

Usage Notes

- OSS file pathnames can be long and have the potential to exceed both the display width of a screen and the maximum line length of a Guardian EDIT file. Output containing long OSS file pathnames wraps to subsequent lines of the display or edit file in 80-character segments.
- These warning or error messages might appear:

```
3116 Unable to translate OSS file pathname.
```

Examples

- This information appears when *name* specifies an OSS file:

```
+ LISTGNAME "/Path/to/the/file"
$DATA01.ZYQ00001.Y00003D2:340359       "/Path/to/the/file"
```

- This information appears when *name* specifies an OSS directory:

```
+ LISTGNAME "/Path/to/the"
Directory:                          "/Path/to/the/Sourcefiles"
Directory:                          "/Path/to/the/Objectfiles"
$DATA01.ZYQ00001.Y00003D2:340359    "/Path/to/the/file"
$DATA01.ZYQ00001.Y00003D2:340223    "/Path/to/the/file1"
$DATA01.ZYQ00001.Y00003D2:340864    "/Path/to/the/file2"
```

- This information appears when *name* specifies an ANSI SQL file:

```
+ LISTGNAME 'TABLE CATALOG_12.SCHEMA_34.TABLE_56'
$DATA01.ZSD12345.Z1234567:340259
'TABLE CATALOG_12.SCHEMA_34.TABLE_56 PARTITION PARTITION_01'
$DATA01.ZSD12345.Z1234567:340282
'TABLE CATALOG_12.SCHEMA_34.TABLE_56 PARTITION PARTITION_02'
$DATA01.ZSD12345.Z1234567:340276
'TABLE CATALOG_12.SCHEMA_34.TABLE_56 PARTITION PARTITION_03'
```

LISTPNAME

In Measure G09 and later PVUs, the LISTPNAME command translates a Guardian file name (*gname*) and creation version serial number (CRVSN) to the corresponding OSS file pathname. The CRVSN distinguishes a specific instance of a Guardian file name in cases of file name reuse. If the Guardian file name specified identifies a file that is accessible through more than one path, all paths are listed.

OSS file pathname translation is affected by the current MEASCOM session and the type of data files being added or deleted. For examples that describe how this pathname translation is determined, see the table in LISTGNAME (page 97).

In Measure G11 and later PVUs, LISTPNAME is still available but is superseded by LISTENAME (page 96), which supports both OSS file pathnames and ANSI SQL names.

Syntax

```
LISTPNAME [ / OUT filename / ] gname [:crvsn]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends the command output to it.

gname

is the Guardian file name to be translated.

crvsn

is the creation version serial number associated with a specific instance of a *gname*. *crvsn* can be obtained from the MEASCOM entity displays and is appended to the Guardian file name using a colon.

Usage Notes

- The /G form of a Guardian file name is always assumed to exist as a valid OSS file pathname. The /G name is listed in a LISTPNAME output only if translation was through an OSS journal segment and the access information recorded in the measurement data file indicates that the /G form was used to access the file for a particular record.
- These warning or error messages might appear with this command:

```
3023 WARNING. CRVSN was not specified, translation may be incorrect.

3114 OSS journal segment is required in this context and is not available for access.
```

Examples

This example shows the output from a LISTPNAME command where the *CRVSN* is not specified. In this case, a warning indicates a potentially incorrect OSS file pathname translation.

```
+ LISTPNAME $DATA01.ZYQ00001.Y00003D2
  MEAS 3023 WARNING. Crvsn was not specified, OSS file pathname translation
     may not be correct.
  $DATA01.ZYQ00001.Y00003D2:340259    "/Path/to/the/file"
  $DATA01.ZYQ00001.Y00003D2:340259    "/Another/Path/to/the/file"
  $DATA01.ZYQ00001.Y00003D2:340259    "/Still/another/path"
```

This example shows the output from a LISTPNAME command where the *CRVSN* was specified:

```
+ LISTPNAME $DATA01.ZYQ00001.Y00003D2:340359
  $DATA01.ZYQ00001.Y00003D2:340259    "/Path/to/the/file"
  $DATA01.ZYQ00001.Y00003D2:340259    "/Another/Path/to/the/file"
  $DATA01.ZYQ00001.Y00003D2:340259    "/Still/another/path"
```

LOG

This command starts and stops logging of a session. The log contains all commands entered and their output.

Syntax

```
LOG { TO filename | STOP }
```

TO *filename*

starts logging a session to the specified file. If logging is already in process, MEASCOM closes the log file, opens *filename*, and logs the session to the newly opened file.

filename is the name of the log file. If *filename* already exists, data is appended to the existing file. If *filename* does not exist, MEASCOM creates it.

STOP

stops logging the session and closes the log file.

Usage Note

Logging does not affect MEASCOM command output, and the OUT command does not affect the contents of a log file.

Example

This example starts logging the session. The ENV command shows environmental information, including the current log file:

```
+ LOG TO $DATA.MEAS.LOG04
+ ENV
  System \BUYER
  Volume $DATA.MEAS
  Log to $DATA.MEAS.LOG04
  Assume MEASUREMENT
```

OBEY

This command reads MEASCOM commands from a specified file.

Syntax

```
OBEY filename [ NOECHO ]
```

filename

is the name of a command file that contains MEASCOM commands.

NOECHO

prevents the command file from echoing its commands to an output file.

Usage Notes

- The terms command file and OBEY file are often used interchangeably.
- You can nest up to four command files. When MEASCOM reaches the end of the primary command file, it resumes reading commands from the device where the first OBEY command was entered. Nested files can have different echoing options.
- Much of the command output for MEASCOM is in the correct format for a command file. You can use the OUT command, or the OUT option in some MEASCOM commands, to direct output to a file. With a little editing, you can then use that file as a command file.

Example

This example shows the execution of a command file. The commands and comments following the OBEY command were read from the command file named DAILY:

```
+ OBEY DAILY
+ -- begin obey file
+ Add cpu 7
+ Add process *
+ Add file $*.*.ACCOUNT
+ -- end obey file
```

OSSPATH

Starting with the Measure G09 PVU, the OSSPATH command specifies a default directory for use in expanding OSS file pathnames. OSS file pathnames encountered in MEASCOM that do not begin with a slash (/) are expanded using the current setting of OSSPATH.

Syntax

```
OSSPATH "directory"
```

"*directory*"

is a fully qualified OSS file pathname that identifies a directory to be used as the default directory. Subsequent OSS file pathnames that do not begin with a slash (/) are appended to the default directory name to form a fully qualified OSS file pathname. The default value is slash (/).

Example

This example shows the output from using the OSSPATH command to declare a default directory:

```
+ OSSPATH "/Path/to/the/SourceFiles"
+ LISTGNAME "Server"
  $DATA01.ZYQ00001.Y000019F.340568 "/Path/to/the/SourceFiles/Server"
```

OUT

This command directs command output to a specified file. You can use the OUT command to create a command (OBEY) file based on MEASCOM command output. For example, the output of the INFO MEASUREMENT command can be used, with little editing, to rerun a measurement.

Syntax

```
OUT filename
```

filename

is the name of the listing file. If *filename* already exists, command output is appended to the file. If *filename* does not exist, MEASCOM creates an edit file by that name and sends command output to it.

Usage Note

If command output is already being sent to another file, that file is closed before the new output file is opened.

Example

This example directs command output to the file $DATA.MEAS.OUT04. Output from subsequent commands (such as the ENV command in this example) is not displayed at the terminal.

```
+ OUT $DATA.MEAS.OUT04
+ ENV
```

PAGESIZE

In Measure G09 and later PVUs, this command designates the number of output lines, including the prompt, to be displayed before the double prompt (++). Many of the entity report formats exceed the display area of a standard 24-line by 80-column screen. Because displaying OSS file pathnames can wrap to several lines of text, the length of many entity report formats varies. To prevent data from scrolling off the display area before it is read, use the PAGESIZE command to set a double prompt (++).

MEASCOM also presents a double prompt (++) between individual reports when a group of reports is displayed. The double prompt (++) between reports is not affected by the setting of PAGESIZE and continues to exist between reports.

Syntax

```
PAGESIZE [ integer ]
```

integer

is a value from 6 through 128. When specified, integer - 1 lines of report are displayed followed by the double prompt (++). If no value is set, the default value of 128 is used, and double prompts (++) are displayed only between individual reports.

Usage Notes

- To continue with the display of report data, press ENTER.
- To terminate the display and return to the main MEASCOM prompt, press the BREAK key or enter Ctrl-Y.
- The PAGESIZE command has no effect on reports listed to a spooler or disk output file.
- These warning or error messages might appear with this command:

```
3053 Value out of range; value must be 6:128.
```

Example

To prevent data from scrolling out of a standard 24-line by 80 character display area:

```
+ PAGESIZE 24
```

RESET PLOT

This command resets one or more plot options to their default values.

NOTE: The RESET command accepts attributes, but no values. To set a plot to a value other than the default, refer to the SET PLOT (page 104) command.

Syntax

```
RESET [ PLOT ] { * | plot-attribute [, plot-attribute ] ... }
```

PLOT

is optional if you specified PLOT as the command object by using the ASSUME command.

plot-attribute

is one of:

FOR

resets the duration of the plot window to extend to the measurement end date and time.

FROM

resets the start of the plot window to the measurement start date and time.

SCALE-FROM

resets the lower boundary of the data-value axis to 0.

SCALE-TO

resets the upper boundary of the data-value axis to 100.

TIME-BASE

resets the TIME-BASE attribute to the default value. If a time interval was specified when the data was collected, the default is TIME-BASE ON. Otherwise, the default is TIME-BASE OFF.

TO

resets the end of the plot window to the measurement end date and time.

VERT-BASE

resets the display of the time axis to ON.

WIDE-ITEM

resets the density of the plot to OFF.

RESET REPORT

This command sets one or more report options to default values.

NOTE: The RESET command accepts attributes, but no values. To set a report to a value other than the default, refer to the SET REPORT (page 108) command.

Syntax

```
RESET [ REPORT ] { * | report-attribute [, report-attribute ] ... }
```

REPORT

is optional if you specified REPORT as the command object by using the ASSUME command.

report-attribute

includes:

CR-NAME-LEN

resets the displayed length of procedure (code-range) names to SHORT.

CR-NAME-FORM

resets the demangled procedure (code-range) names to STANDARD.

CR-NAME-QUAL

resets the procedure (code-range) names to UNQUALIFIED.

DOTS

resets the report displays to not include connecting dots between labels and numeric values.

FOR

resets the duration of the report window to extend to the measurement end date and time.

FORMAT

resets the format of the report to NORMAL.

FROM

resets the beginning of the report window to the measurement start date and time.

LOADID

resets the name to be placed in the *loadid* field of structured records as unspecified.

RATE

resets the display of counter values to ON.

STYLE

In G-series RVUs, resets the default to LEGACY; in H-series RVUs, resets the default to ZMS.

TO

resets the end of the report window to the measurement end date and time.

TOTALS

resets the display of TOTALS to SUPPRESS.

ZERO-REPORTS

resets the display of reports with all zero counter values to SUPPRESS.

ZERO-VALUES

resets the display to show blanks (SUPPRESS) if values are less than 0.005.

The ZERO-VALUES attribute has no effect on structured files.

RUN

The RUN command lets you run another process without exiting from Measure. The MEASCOM RUN command is similar to the TACL RUN command.

Syntax

```
RUN [/OUT out-file / ] run-file [ / options / ]
          [ "parameter string" ]
```

out-file

is an output file for any error messages generated by the RUN command.

run-file

is the program file to be run.

options

are run options. Commonly used run options include:

IN *filename*	Specifies an input file for the process being run.
OUT *filename*	Specifies an output file for the process being run.
NOWAIT	Causes the MEASCOM prompt to return immediately after the RUN command executes. If you do not use NOWAIT, Measure pauses until the *run-file* process completes.

Measure does not support all TACL run options. See Usage Notes.

"*parameter*"

is a parameter or a string of parameters to be passed to the newly created process. The parameters must be enclosed in double quotation marks (" ").

Usage Notes

- For basic information about the RUN command and a complete description of its options, see the *TACL Reference Manual.*
- If you specify a program name with no volume or subvolume (for example, RUN FUP), Measure checks the current subvolume. If the file cannot be found or is not an object file, Measure checks $SYSTEM.SYSTEM and $SYSTEM.SYS*nn*. If you specify a remote system (for example, RUN \REMOTE.SURVCOM), Measure checks $SYSTEM.SYSTEM and $SYSTEM.SYS*nn* on the remote system.
- Measure does not fully support certain TACL options:
  - Only the OFF and ON keywords for INSPECT and DEFMODE are supported.
  - The RUN options HIGHPIN and EXTSWAP are not supported.
  - The RUND command is not directly supported. To use DEBUG, enter:

    ```
    RUN program-file / DEBUG /
    ```

Example

To run FUP INFO from the MEASCOM prompt:

```
+ RUN FUP INFO MYSUBVOL.*
```

SET PLOT

This command sets one or more plot attributes that govern subsequent plots, until changed by a new SET PLOT or RESET PLOT command.

Syntax

```
SET [ PLOT ] plot-attribute value [, plot-attribute value ] ...
```

PLOT

is optional if you specified it as the command object by using the ASSUME command.

plot-attribute value

is one of:

FOR *duration*

is the duration of the plot window. FOR and TO are mutually exclusive. This attribute becomes the default for subsequent plots until changed by a new SET PLOT or RESET PLOT command.

duration

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

FROM { [*date*,] *time-of-day* }

specifies the start of the plot window. This attribute becomes the default for all subsequent plots until changed by a new SET PLOT or RESET PLOT command. If you omit FROM, the start time of the measurement or the start time of the entity measurement is used, whichever is later.

date

is the start date of the data to be used for the plot, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *date*, the measurement start date is used.

time-of-day

is the start time of the data to be used for the plot, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *time-of-day*, the measurement start time is used.

SCALE-FROM *number*

sets the lower boundary of the data-value axis.

number

is a number in the range 0 through 999999999999.999. The default is 0.

SCALE-TO *number*

sets the upper boundary of the data-value axis.

number

is a number in the range 0 through 999999999999.999. The default is 100.

TIME-BASE { OFF | ON }

displays a bar graph or a two-axis plot of counter values. If a time interval was specified when the data was collected, the default is TIME-BASE ON. Otherwise, the default is TIME-BASE OFF.

OFF

displays a bar graph of average counter values.

ON

displays a plot of counter values over time.

TO [*date*,] *time-of-day*

specifies the end of the plot window. TO and FOR are mutually exclusive. This attribute becomes the default for subsequent plots until changed by a new SET PLOT or RESET PLOT command.

date

is the end date of the plot in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *date*, the measurement end date is used.

time-of-day

is the end time of the data to be used for the plot, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *time-of-day*, the measurement end time is used.

VERT-BASE { OFF | ON }

controls the display of the time axis. The default is ON.

OFF

for a two-axis plot, displays time on the horizontal axis. For a bar graph, displays the bars vertically (base on the horizontal axis).

ON

for a two-axis plot, displays time on the vertical axis. For a bar graph, displays the bars horizontally (base on the vertical axis).

WIDE-ITEM { OFF | ON }

sets the density of the plot. The default is OFF.

OFF

for a two-axis plot, displays a single plot character for each counter value at each time interval. For a bar graph, displays bars one character wide.

ON

for a two-axis plot, fills in the area between the time axis and the lowest counter value. For a bar graph, displays bars two to six characters wide, depending on the number of counter values on the graph.

Related Commands

Command	Function	Page
ADD PLOT	Specifies a plot	48
LIST PLOT	Displays a plot	80

Example

This example displays a two-axis plot and a bar graph. Both were made using the same data.

```
+ SET PLOT SCALE-TO 8
+ SET PLOT VERT-BASE OFF
+ SET PLOT WIDE-ITEM ON
+ LIST PLOT, INTERVAL 1 MINUTE
    7.99 -------------------
    7.62 -                   -
    7.26 -                   -
    6.90 -                   -
    6.53 -                   -
    6.17 -                   -
    5.81 -    B              -
    5.45 -                   -
    5.08 -  B                -
    4.72 -                   -
    4.36 -                   -
    3.99 -                   -
    3.63 -B A A              -
    3.27 -  * *              -
    2.90 -  * *   B          -
    2.54 -A * *              -
    2.18 -* * *   A      B   -
    1.82 -* * *   *    B     -
    1.45 -* * * B *          -
    1.09 -* * *   * B A      -
    .726 -* * * A *   *      -
    .363 -* * * * * A * A    -
       0 -*-*-*-*-*-*-*-A-
   11:43:14  45  47  49  51
           44  46  48  50

        Min Value = 0          Max Value = 5.86
      A: READ-BUSY-TIME        Disk $BOOKS1 (6, 1, 17, 1)
      B: READ-BUSY-TIME        Disk $BOOKS1 (6, 1, 17, 2)

+ SET PLOT TIME-BASE OFF
+ LIST PLOT, INTERVAL 1 MINUTE
    7.99 ----------------------------------------
    7.62 -                                       -
    7.26 -                                       -
    6.90 -                                       -
    6.53 -                                       -
    6.17 -                                       -
    5.81 -                                       -
    5.45 -                                       -
    5.08 -                                       -
    4.72 -                                       -
    4.36 -                                       -
```

```
3.99 -                                          -
3.63 -                                          -
3.27 -                                          -
2.90 -       -----                              -
2.54 -      |     |                             -
2.18 -      |     |                             -
1.82 - -----|     |                             -
1.45 -|     |     |                             -
1.09 -|     |     |                             -
.726 -|     |     |                             -
.363 -|     |     |                             -
   0 -|     |     |-----------------------------
         A     B

   Min Value = 1.80        Max Value = 2.92
 A: READ-BUSY-TIME         Disk $BOOKS1 (6, 1, 17, 1)
 B: READ-BUSY-TIME         Disk $BOOKS1 (6, 1, 17, 2)
```

SET REPORT

This command governs the format of subsequent reports until changed by a new SET REPORT command, a RESET REPORT command, or the addition of a new data file.

Syntax

```
SET [ REPORT ] report-attribute value [, report-attribute value ] ...
```

REPORT

is optional if you specified REPORT as the command object by using the ASSUME command.

report-attribute value

includes:

CR-NAME-LEN { SHORT | LONG }

specifies the displayed length of procedure (code-range) names. The default is SHORT.

SHORT

truncates procedure (code-range) names (if necessary) at 32 characters. Mangled procedure names are not demangled prior to display.

LONG

displays procedure (code-range) names both in their SHORT form and, if longer than 32 characters, in their entirety on a subsequent line. Mangled procedure names are not demangled for display.

CR-NAME-FORM { STANDARD | DEMANGLED | BOTH }

specifies whether demangled procedure (code-range) names are displayed (if applicable). The default is STANDARD.

STANDARD

displays procedure (code-range) names in the form was specified in the code file (mangled) or EDIT file (demangled).

DEMANGLED

demangles procedure (code-range) names, if necessary, prior to display.

BOTH

displays procedure (code-range) names in both STANDARD and DEMANGLED forms.

CR-NAME-QUAL { UNQUALIFIED | QUALIFIED }

specifies whether procedure (code-range) names are displayed with object file name qualifiers, if available, in the "Code-Range Name" column on the line immediately following the traditional line of code-range output. The default is UNQUALIFIED.

UNQUALIFIED

displays procedure names in traditional form with no qualifiers.

QUALIFIED

displays procedure names with the Guardian or OSS object file name of the associated code. This can be useful in differentiating between like-named procedures in different object files. Guardian file names have their associated CRVSN appended to the end of the name.

DOTS { ON } { OFF }

indicates whether report displays should include connecting dots between labels and numeric values. Valid only if STYLE is set to ZMS. The default is OFF.

ON

displays dots to connect labels to formatted numbers.

OFF

does not display dots to connect labels to formatted numbers.

FOR *duration*

is the duration of the report window. FOR and TO are mutually exclusive.

duration

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

FORMAT { BRIEF } { NORMAL } { STRUCTURED }

sets the format of the report. The default is NORMAL.

BRIEF

displays a subset of the counters. The DDL records listed in this section list the counters for each entity type. The counters that belong to the BRIEF subset are in boldface type. For PROCESSH entities, if TOTALS INCLUDE or TOTALS ONLY is also specified, only code-file level totals (not per-process totals) are displayed.

For more information about which counters are included in brief-format reports, see the boldface counters in the DDL record for the specified entity type in Section 3, Entities and Counters.

NORMAL

displays all counters.

STRUCTURED

writes the report to a structured file for subsequent examination using Enform. A single record consists of all counters for a single entity. Reports for entities of the same entity type are written to the same file, and the file is named for the entity type. If the file already exists, MEASCOM appends the data to the file. If the file does not exist, MEASCOM creates it. (Structured files are closed when you modify the FORMAT attribute.)

FROM [*start-date*], *start-time-of-day*

specifies the beginning of the report window. If you omit FROM, the start time of the measurement or the start time of the entity measurement is used, whichever is later.

start-date

is the start date of the data to be used for the report, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *start-date*, the measurement start date is used.

start-time-of-day

is the start time of the data to be used for the report, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *start-time-of-day*, the measurement start time is used.

LOADID *loadid*

specifies the name to be placed in the *loadid* field of structured records.

loadid

is an alphanumeric string, 1 through 8 characters long. The string can contain letters, numbers, carets (^), hyphens (-), and underscores (_). The first character must be a letter.

RATE { OFF | ON }

determines how counter values are displayed. The RATE attribute has no effect on structured files. The default is ON.

OFF

displays uninterpreted counter values.

ON

displays interpreted counter values (counts per second and percent busy).

STYLE { LEGACY | ZMS }

indicates whether displays and structured data are formatted using the ZMS style external interface (Measure G11 or later) or using the external interface compatible with pre-G11 Measure PVUs. In G-series and earlier RVUs, the default is LEGACY; in H-series RVUs, the default is ZMS.

LEGACY

displays and structured records use the pre-G11 Measure PVU style.

ZMS

displays and structured records use the style introduced in Measure G11.

TO [*end-date*,] *end-time-of-day*

specifies the end of the report window. TO and FOR are mutually exclusive.

end-date

is the end date of the report, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *end-date*, the measurement end date is used.

end-time-of-day

is the end time of the data to be used for the report, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *end-time-of-day*, the measurement end time is used.

TOTALS { INCLUDE | ONLY | SUPPRESS }

specifies whether TOTALS are displayed. The default is SUPPRESS.

INCLUDE

indicates that both the per-process and aggregated totals are displayed. When aggregated totals are displayed, the set of totals is enclosed in brackets ([…]), and the number of processes that were executing the code being totaled precedes the individual code-range results, also enclosed in brackets (that is, [Totals across 3 processes]).

ONLY

For all entities except PROCESSH and USERDEF, displays only the final TOTALS report. If only one entity report is generated, the TOTALS report is not displayed.

For the PROCESSH entity, only aggregated PROCESSH data is displayed.

NOTE: PROCESSH data can be collected on either a per-code-file basis or a per-procedure basis, depending on whether a code-file-spec was supplied when the PROCESSH measurement was configured (via the ADD PROCESSH command). Thus, the displayed totals can cover an entire code file or each of a set of code ranges within the code file.

SUPPRESS

displays only the entity reports. (The TOTALS attribute is ignored if a report contains only measurements for one entity.)

For PROCESSH, SUPPRESS indicates only per-process data is displayed. (Aggregated data is not displayed.)

SUPPRESS has no effect on USERDEF reports.

ZERO-REPORTS { INCLUDE | SUPPRESS }

determines whether records containing only zero values are displayed. The default is SUPPRESS.

INCLUDE

displays entity records even if all counter values are zero.

SUPPRESS

does not display entity records if all counter values are zero.

ZERO-VALUES { INCLUDE | SUPPRESS }

determines whether values less than 0.005 are displayed as zeros or blanks.

INCLUDE

displays zeros if counter values are less than 0.005.

SUPPRESS

displays blanks if counter values are less than 0.005.

The ZERO-VALUES attribute has no effect on structured files. The default is SUPPRESS.

*

resets all report attributes to default values.

Related Commands

Command	Function	Page
ADD MEASUREMENT	Specifies reports	46
LIST *entity-type* LISTACTIVE *entity-type* LISTALL *entity-type*	Displays reports	68 85 90

Examples

- To display all CPU reports in the brief format, total the counter values across the reports, and produce a final report containing the totaled counter values:

```
+ SET REPORT FORMAT BRIEF, TOTALS INCLUDE
+ LIST CPU *

Cpu  6 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \SAIID From 7 Aug 1994, 14:53:01 For 75 Seconds

Cpu-Busy-Time       17.86%    Swaps                0.03
Ending-Free-Mem               Ending-UCME
++
Cpu  7 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \SAIID From 7 Aug 1994, 14:53:01 For 75 Secs

Cpu-Busy-Time        2.22%    Swaps
Ending-Free-Mem               Ending-UCME
++
Totals for     2 Records
From   7 Aug 1994, 14:53:01    For    75 Seconds

Cpu-Busy-Time       20.07%    Swaps                0.03
Ending-Free-Mem               Ending-UCME
```

- To display a full report, with counter values displayed as counts and seconds busy rather than rates and percent busy:

```
+ SET REPORT FORMAT NORMAL, RATE OFF
+ LIST CPU 6

Cpu  6 VLX          Initial Lock Pgs 2048     Mem Pages  8192
Memory MB 16        PCBs    100               Page Size  2048
Local System \SAIID From 7 Aug 1994, 14:53:01 For 75 Seconds

Cpu-Busy-Time       13.52s    Swaps                 2#
Cpu-Qtime           18.56s    Cpu-Qlen-Max         14#
Mem-Qtime           82.42m    Mem-Qlen-Max          5#
Dispatches          3,297#    Intr-Busy-Time     2.49s
Process-Ovhd                  Send-Busy-Time    46.65m
Disc-IOs              143#    Cache-Hits           67#
Transactions                  Response-Time
Page-Requests                 Page-Scans
Ending-Free-Mem      4092     Ending-UCME
Ending-UDS            500     Ending-SDS           1500
Ending-UCL            100     Ending-SCL           2000
```

- To display a report using the ZMS style available in Measure G11 and later PVUs:

```
+ SET REPORT STYLE ZMS
+ LIST CPU 3, RATE OFF

Cpu  3 NSR-T          Init Lock Pgs    269          Mem Pages   131072
Memory MB  2048       PCBs            2200          Pg Size    16384  Bytes
Format Version:  G11  Data Version:  G09  Subsystem Version:   1
Local System \MEASURE From  27 Mar 2002,  5:55:53   For    47 Minutes
------------ Processor ------------------------------------------------
Cpu-Busy-Time               1.94 sec  Dispatches                  3,647 #
Cpu-Qtime                   4.41 sec  Intr-Busy-Time             123.95 ms
Process-Ovhd              155.45 ms   Send-Busy-Time
Comp-Traps                  1,298 #
Native-Busy-Time            1.63 sec  Accel-Busy-Time            165.03 ms
TNS-Busy-Time              18.22 ms   PROCESSH-Samples
------------ Memory ---------------------------------------------------
Starting-Free-Mem         111,999 #    Ending-Free-Mem           128,999 #
Swaps                       1,999 #    Page-Requests               3,999 #
MM-Page-Scans               3,999 #    Page-Scans                 14,995 #
Unsp-Pages-Qtime          499,999 sec  Mem-Qtime
Unsp-Pages-Start               14 #    Unsp-Pages-End                 14 #
Starting-SCL                29,999 #    Ending-SCL                  7,999 #
Starting-UCL                25,999 #    Ending-UCL                  1,999 #
Starting-Free-CIDs          31,999 #    Ending-Free-CIDs           31,999 #
Ending-UCME
------------ Device I/O -----------------------------------------------
Disc-IOs                         4 #    Cache-Hits                   141 #
Transactions                            Response-Time
------------ Messaging ------------------------------------------------
Link-Prepush-Msgs       7,989,999 #    Readlinkcache-All        7,289,999 #
Link-Readlink-Msgs        102,999 #    Readlinkcache-Ctrl      22,959,999 #
Link-Large-Msgs           102,999 #    Readlinkcache-None
                                       Replyctrlcache-Msgs      7,289,999 #
------------ ServerNet Transfers Initiated by this Local Processor ----------
to/fm     READS INTO CPU     READ BYTES    WRITES FROM CPU    WRITE BYTES
CPU-0              458 #        337,750 #            452 #        195,478 #
CPU-1
                  .
                  .
                  .
CPU-15
See TOTAL for sum over all initiating CPUs: total data into & from each CPU.
```

- To display a report using dots to connect labels to formatted numbers:

```
+ SET REPORT STYLE ZMS
+ SET REPORT DOTS ON
+ LIST DISKFILE *

Diskfile &D0TAE.SAFE.GUARD                          File Format 1
Device Name $D10D                                   Pool
Local CPU 2           File Type  Key Sequenced      File Code 889
Format Version:  G11  Data Version:  G09  Subsystem Version:   1
Local System \SYSTEM  From  27 Mar 2002,  5:55:53   For  69.9 Seconds
------------ File ------------------------------------------------------
Open-Qtime ..............  999799 sec  Transient-Opens ..............  238 #
Ending-EOF ..............  45,989 B    File-Growth-Ratio ..........  1.00
Extent-Allocations
------------ Logical I/O -----------------------------------------------
Lockwait-Time ..............  3.14 sec  Requests ..................  2,999 #
Max-Lockwait-Time ..........  3.14 sec  Requests-Blocked
Lock-Timeouts                          Lock-Bounces
Cache-Read-Hits ...........  2,999 #    Cache-Write-Hits ..........  2,479 #
Block-Splits                           Cache-Write-Cleans
Driver-Input-Calls ........  3,999 #    Driver-Output-Calls ........2,999 #
DBIO-Input-Calls ............  178 #    DBIO-Output-Calls ............  29 #
SQL-Inserts ..................  1 #    SQL-Deletes
SQL-Updates                            SQL-Ending-Rows ..............  1 #
OSS-Cache-Callbacks                    OSS-Callback-Writes
OSS-Block-Writes                       OSS-Block-Write-Bytes
```

- This example, on G-series, displays collected PROCESSH samples with full, demangled, and qualified code-range names (LONG, DEMANGLED, QUALIFIED):

```
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.TCPIP)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZCRESRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZINETSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZLANCSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.ZCRTLSRL)
+ ADD PROCESSH $SYSTEM.SYS00.TCPIP ($SYSTEM.SYS00.TSYSCLR)
+ SET REPORT CR-NAME-LEN  LONG
+ SET REPORT CR-NAME-FORM DEMANGLED
+ SET REPORT CR-NAME-QUAL QUALIFIED
+ LIST PROCESSH *, RATE OFF, IF CODE-RANGE > 0
+
```

SQLCATALOG

In Measure H01 and later PVUs, this command specifies a default catalog for use in expanding partially qualified ANSI SQL names.

Syntax

```
SQLCATALOG ['catalog-name']
```

catalog-name

is an ANSI SQL catalog identifier. If *catalog-name* is not specified, the default empty value is used.

Example

To specify a default catalog for expanding partially qualified ANSI SQL names:

```
+ SQLCATALOG 'CATALOG_12'
```

SQLSCHEMA

In Measure H01 and later PVUs, this command specifies a default schema, and optionally a default catalog, for use in expanding partially qualified ANSI SQL names.

Syntax

```
SQLSCHEMA ['[catalog-name.]schema-name']
```

catalog-name

is an ANSI SQL catalog identifier. If no default catalog is specified and there is no previous default catalog, a warning message is displayed.

schema-name

is an ANSI SQL schema identifier.

If neither *catalog-name* nor *schema-name* is specified, the default empty value is used.

Example

To specify a default schema for expanding partially qualified ANSI SQL names:

```
+ SQLSCHEMA 'SCHEMA_34'
```

SETPROMPT

This command is similar to the TACL SETPROMPT command. SETPROMPT lets you redefine the MEASCOM input prompt to include environmental information, such as the current assumed object or the current log file. For more information on the TACL SETPROMPT command, see the *TACL Reference Manual*.

Syntax

```
SETPROMPT { ASSUME   }
          { LOG      }
          { NONE     }
          { OUT      }
          { SYSTEM   }
          { VOLUME   }
          { SWAPVOL  }
          { COMMENTS }
          { WARNINGS }
```

ASSUME
: displays the current assumed object.

LOG
: displays the current log file.

NONE
: displays the standard MEASCOM prompt (the line number and a plus sign). The default is NONE.

OUT
: displays the current output file.

SYSTEM
: displays the current system name.

VOLUME
: displays the current volume and subvolume name.

SWAPVOL
: displays the current swap volume.

COMMENTS
: displays either DISPLAY ALL or SUPPRESS ALL, whichever is currently active for the COMMENTS command. To determine which comments are individually displayed or suppressed, use the ENV command.

WARNINGS
: displays either DISPLAY ALL or SUPPRESS ALL, whichever is currently active for the WARNINGS command. To determine which warnings are individually displayed or suppressed, use the ENV command.

Usage Notes

- If you set the MEASCOM prompt to a parameter that is currently inactive, no information appears in the prompt. For example, if you set the prompt to LOG while logging is not in effect, the prompt does not display a log file name. However, when logging starts, the prompt changes to the current log file name.
- The prompt reverts to the default setting (NONE) when you exit Measure.

Examples

- To set the MEASCOM prompt to display the current volume name:

```
12+ SETPROMPT VOLUME
$DATA MEAS13+
```

- To set the MEASCOM prompt to display the current assumed object, then change the assumed object:

```
14+ SETPROMPT ASSUME
MEASUREMENT 15+ ASSUME PLOT
PLOT 16+
```

SHOW PLOT

This command displays the setting of one or all plot attributes.

Syntax

```
SHOW [ / OUT filename / ] [ PLOT ] [ attribute ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

Following execution of the SHOW PLOT command, MEASCOM redirects its output to the current OUT file (typically, the terminal). This attribute does not affect the contents of the log file.

PLOT

is optional if you specified PLOT as the command object by using the ASSUME command.

attribute

is one of:

FOR	TIME-BASE
FROM	TO
SCALE-FROM	VERT-BASE
SCALE-TO	WIDE-ITEM

For descriptions of these attributes, see SET PLOT (page 104).

If you omit *attribute*, MEASCOM displays all attributes.

Usage Notes

- The SHOW PLOT command output is a list of the commands used to set the plot attributes. Use /OUT *filename*/ to create a command (OBEY) file that you can later use to restore the current set of attributes.
- MEASCOM terminates a PLOT with TIME-BASE ON when it detects that no more records match the time range.

Example

To display all plot attributes:

```
+ SHOW PLOT
  Set Plot Vert-Base  on
  Set Plot Time-Base
  Set Plot Scale-From       0.000
  Set Plot Scale-To       100.000
  Set Plot Wide-Item  off
  Set Plot From
  Set Plot To
  Set Plot For
```

SHOW REPORT

This command displays the setting of one or all report attributes.

Syntax

```
SHOW [ / OUT filename / ] [ REPORT ] [ attribute ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

Following execution of the SHOW REPORT command, MEASCOM redirects its output to the current OUT file (typically, the terminal). This attribute does not affect the contents of the log file.

REPORT

is optional if you specified it as the command object by using the ASSUME command.

attribute

is one of:

CR-NAME-FORM	FORMAT	TO
CR-NAME-LEN	FROM	TOTALS
CR-NAME-QUAL	LOADID	ZERO-REPORTS
FOR	RATE	ZERO-VALUES

For descriptions of these attributes, see SET REPORT (page 108).

If you omit *attribute*, MEASCOM displays all attributes.

Usage Notes

- The SHOW REPORT command output is a list of the commands used to set the report attributes. Use /OUT *filename*/ to create a command (OBEY) file that you can later use to restore the current set of attributes.
- In Measure G12 and later PVUs, code-space identifiers are displayed only for TNS code files (including accelerated TNS code files).

Examples

- To display all report attributes:

```
+ SHOW REPORT
Set Report Format normal
Set Report Rate on
Set Report Totals suppress
Set Report Zero-Values suppress
Set Report Zero-Reports suppress
Set Report From
Set Report For
Set Report LoadId
Set Report CR-NAME-LEN  short
Set Report CR-NAME-FORM standard
Set Report CR-NAME-QUAL unqualified
+
```

- To only display the current value of the CR-NAME-LEN:

```
+ SHOW REPORT CR-NAME-LEN
Set Report CR-NAME-LEN short
```

START MEASSUBSYS

This command starts the Measure subsystem.

Syntax

```
START [ MEASSUBSYS ]
START [ MEASSUBSYS ] , cpu
```

MEASSUBSYS

is optional if you specified MEASSUBSYS as the command object by using the ASSUME command.

cpu

is the number of a CPU on your system. Specify *cpu* when the Measure subsystem is already running and you need to restart MEASCTL in a specific CPU.

Under normal conditions, you should not have to use the *cpu* parameter. If a CPU goes down and then is restarted, MEASMON automatically starts a new MEASCTL process in that CPU. However, if an attempt to start a MEASCTL using MEASMON fails, you can start MEASCTL manually by issuing START MEASSUBSYS with the *cpu* parameter.

The *cpu* parameter is valid only when the Measure subsystem is running.

Related Command

Command	Function	Page
STOP MEASSUBSYS	Stops the Measure subsystem	126

CAUTION: Do not stop the MEASCTL swap volume while Measure is running. Doing so causes system failure. For more information, see Usage Notes.

Usage Notes

- The START MEASSUBSYS command can be executed only by a super-group user—that is, a user who has user ID (255,*n*).
- To avoid having the MEASCTL swap volume brought down inadvertently, use only $SYSTEM or a mirrored volume as the swap volume. If you start MEASSUBSYS by using a swap volume other than $SYSTEM, MEASCOM issues a warning message. If possible, stop MEASSUBSYS and then restart it:

  ```
  MEASCOM/SWAP $SYSTEM/ START MEASSUBSYS
  ```

- If large measurements take excessively long to start, add these DEFINE statements before the START MEASSUBSYS command in your measurement application:

  ```
  ADD DEFINE  =_MEASCTL_SKEW_COPYTIME, FILE A
  ADD DEFINE  =_LOCK_THE_CID_TABLE, FILE A
  ```

  NOTE: The second ADD DEFINE statement applies to Measure G08 and earlier PVUs.

  The FILE A addendum is required for syntax but is not used by TACL.

  These DEFINE statements do not impact the startup time for small measurements.

- The CID table is an internal structure that determines how many entities can be measured. The table assigns one counter identifier (CID) to each measured entity. For example, if you request measurement of five different files, the CID table assigns one CID per file, no matter how many counters the measurement includes for each file. In Measure product versions earlier than G05, the CID table has an upper limit of 32,000 CIDs per processor. In product versions D45, G05, and later, you can increase the CID table limit according to the DEFINE

statements, shown below. For H-series and J-series RVUs, the default size of the CID table is 64,000 CIDs per processor.

To increase the CID table limit, issue one of these DEFINE statements before issuing the START MEASSUBSYS command. Table limits of 192,000 and 256,000 are valid only for H-series and J-series RVUs. Table limits of 512,000 are valid only for Measure H04, J02, and later PVUs.

```
ADD DEFINE  =_SET_MAX_CIDS_TO_64000,  file a
ADD DEFINE  =_SET_MAX_CIDS_TO_96000,  file a
ADD DEFINE  =_SET_MAX_CIDS_TO_128000, file a
ADD DEFINE  =_SET_MAX_CIDS_TO_192000, file a
ADD DEFINE  =_SET_MAX_CIDS_TO_256000, file a
ADD DEFINE  =_SET_MAX_CIDS_TO_512000, file a
```

The `file a` parameter is required for syntactical correctness but is not used by Measure.

For each additional 32,000 CIDs, the Measure performance monitor requires an additional 3.5 megabytes of memory for CIDs and 6.5 megabytes for counter space (a total of 10 megabytes) for each CPU. Increasing the CID table limit might have a noticeable impact on system performance.

- In Measure G09 and later PVUs, the default operation for Measure is not to journal OSS file pathname information. This avoids the potentially performance-intensive task of journal file resolution at measurement shutdown. To override the default, use the OSS qualifier with the START MEASUREMENT request.

  You can also use this DEFINE statement in the TACL session prior to issuing a START MEASSUBSYS command. The DEFINE changes the Measure default mode of operation to always collect and resolve OSS file pathname information unless explicitly requested not to do so in the START MEASUREMENT command:

  ```
  ADD DEFINE  =_OSS_JOURNAL_DEFAULT_ON, FILE A
  ```

  To query the current setting of default journaling mode, use the STATUS MEASSUBSYS command.

- In Measure G11 and later PVUs, the default operation for Measure is not to journal ANSI SQL name information. This avoids the potentially performance-intensive task of journal file resolution at measurement shutdown. To override the default, use the SQL qualifier with the START MEASUREMENT request.

  You can also use this DEFINE statement in the TACL session prior to issuing a START MEASSUBSYS command. The DEFINE changes the Measure default mode of operation to always collect and resolve ANSI SQL name information unless explicitly requested not to do so in the START MEASUREMENT command:

  ```
  ADD DEFINE  =_SQL_JOURNAL_DEFAULT_ON, FILE A
  ```

  To query the current setting of default journaling mode, use the STATUS MEASSUBSYS command.

START MEASUREMENT

This command specifies the file where the measurement data is to be written and starts the measurement.

In Measure G09 and later PVUs, you can enable or disable OSS journal segment construction for the resolution of internal OSS path information in data records.

In Measure H02 and later PVUs, you can use the FILESIZE and NOCOUNTERS options to select the measurement data file size and suppress counter data records. The START MEASUREMENT command is enhanced to allow options to be specified in any order.

Syntax

```
START [ MEASUREMENT ] data-file
   [ , FROM [ start-date ] , start-time-of-day ]
   [ , FOR duration ]
   [ , TO [ end-date ] , end-time-of-day ]
   [ , INTERVAL interval ]
   [ , { OSS | NOOSS } ]
   [ , { SQL | NOSQL } ]
   [ , FILESIZE file-size [ MB | GB ] ]
   [ , NOCOUNTERS ]
```

MEASUREMENT

is optional if you specified MEASUREMENT as the command object by using the ASSUME command.

data-file

is the file to which the measurement data is written. If you allocate your own data file, the maximum segment size is 127.5 MB.

If *data-file* exists, the Measure file-handling process (MEASFH) clears the file before starting the measurement. If *data-file* is currently active, MEASFH returns an error message.

If *data-file* does not exist, MEASFH creates it as an unstructured file with a file code of 175. Primary and secondary extent sizes are 2048 pages each, and the maximum number of extents is 256. This results in a default file size (capacity) of 1024 MB.

If *data-file* is a disk file, MEASCOM performs an implicit ADD MEASUREMENT *data-file*, so you can examine the collected data (using one of the LIST commands) without explicitly entering the ADD MEASUREMENT command.

If *data-file* is a tape file, you must copy that file to a disk file before using MEASCOM to examine the collected data.

FROM [*start-date* ,] *start-time-of-day*

specifies the start of the measurement. If you omit FROM, the measurement begins immediately.

start-date

is the start date of the measurement, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *start-date*, the current date is used.

start-time-of-day

is the start time of the measurement, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *start-time-of-day*, the current time is used.

FOR *duration*

specifies the duration of the measurement. FOR and TO are mutually exclusive. If you omit both, you must use the STOP MEASUREMENT command to stop the measurement.

duration

is a time interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

TO [*end-date* ,] *end-time-of-day*

specifies the stop time of the measurement. TO and FOR are mutually exclusive. If you omit both, you must use the STOP MEASUREMENT command to stop the measurement.

end-date

is the date the measurement is to stop, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *end-date*, the current date is used.

end-time-of-day

is the time the measurement is to stop, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *end-time-of-day*, you must use the STOP MEASUREMENT command to stop the measurement.

INTERVAL *interval*

specifies the collection interval—that is, the length of time for which Measure collects data before writing it to the measurement data file. If you omit INTERVAL, the Measure control process writes the counter values to the file twice: once when the measurement (or measurement of a transient entity) begins and once when the measurement (or measurement of a transient entity) ends.

interval

is a collection interval in one of these formats:

n SECOND[S]
n MINUTE[S]
n HOUR[S]

where *n* is an integer in the range 1 through 9999.

CAUTION: A collection interval increases the load on system resources and can affect performance during measurement. For information about collection intervals and performance, see Usage Note (page 124).

OSS

specifies that an OSS journal segment for OSS file pathname resolution is constructed as part of the measurement data. This option overrides the default setting for OSS journal segment construction.

NOOSS

specifies that an OSS journal segment for OSS file pathname resolution is not constructed as part of the measurement data. This option overrides the default setting for OSS journal segment construction.

SQL

specifies that the SQL journal segment for SQL name resolution is constructed as part of the measurement data. This option overrides the default setting for SQL journal segment construction.

NOSQL

specifies that the SQL journal segment for SQL name resolution is not constructed as part of the measurement data. This option overrides the default setting for SQL journal segment construction.

FILESIZE *file-size* [MB | GB]

specifies the desired file size (capacity). If qualified by GB, the file size is interpreted as a gigabyte count (count * $1{,}024^3$). If qualified by MB, or if the qualification is omitted, the file size is interpreted as a megabyte count (count * $1{,}024^2$). This option is similar to the corresponding option in the TMF ADD/ALTER AUDITTRAIL command. The minimum value for FILESIZE is 127 MB. The maximum value is 1048572 MB, 1023 GB, or the maximum disk size (if less than 1,048,572 MB, or 1 TB – 4 MB). The default value is 1024 (resulting in files of 1024 MB). The file will have the specified capacity as a minimum; the actual capacity may be larger. If the file already exists, this option is ignored, and the capacity of the existing file is used.

NOCOUNTERS

suppresses counter data records from the measurement data file. A measurement data file is created which contains configuration information only. Active counter records are maintained in system counter space; this option can be used to conserve disk space when only active counter records are intended to be used. If PROCESSH counters are defined as part of the measurement, warning 3026 is displayed to the user that active counters are not available for these counter types. If the OSS or SQL option is specified, warning 3027 is displayed to the user that the OSS/SQL journal segment is not produced with the NOCOUNTERS option in effect.

Related Command

Command	Function	Page
ADD *entity-type*	Configures a measurement	43

Usage Notes

- You should consider stopping all measurements before using the TACL SETTIME command or the SETSYSTEMCLOCK procedure in order to preserve measurement interval accuracy.
- Use INTERVAL only if you must examine data for specific time intervals within a measurement. INTERVAL writes to the data file all counters in all measured CPUs for every interval specified, thus increasing the number of I/Os performed. The smaller the interval value, the heavier the drain on system resources (CPU, disk, memory) and the faster the data-file growth. To conserve system resources, set the INTERVAL value to the longest interval that will provide useful information.

  Very large measurements and small interval times can cause severe performance problems. If you experience problems, reduce the number and type of entities measured or run the measurement without a collection interval.
- In Measure G09 and later PVUs, the Measure subsystem default is not to build an OSS file pathname journal segment unless specifically requested to do so in a START MEASUREMENT command. To switch the default to always build the OSS file pathname journal segment, use a DEFINE before the START MEASSUBSYS command. For more information, see the Usage Notes for START MEASSUBSYS (page 117).
- In Measure G11 and later PVUs, the Measure subsystem default is not to build the SQL name journal segment unless specifically requested to do so in a START MEASUREMENT command. To switch the default to always build the SQL name journal segment, use a DEFINE before the START MEASSUBSYS command. For more information, see the Usage Notes for START MEASSUBSYS (page 117).
- In Measure H02 and later PVUs, the MEASCOM START MEASUREMENT command allows the user to select the measurement data file size, suppress counter data records in the measurement data file, or both.

Examples

- To start a measurement immediately that writes to the data file MAY07 once each hour and stops in eight hours:

```
+ START MEASUREMENT MAY07, FOR 8 HOURS, INTERVAL 1 HOUR
```

- To start a measurement at 12:00 on October 19, 2004, continue the measurement until 12:00 on October 26, 2004, and have the measurement write to the data file OCTDATA every six hours:

```
+ START MEASUREMENT OCTDATA, FROM 19 OCT 2004, 12:00 &
& TO 26 OCT 2004, 12:00, INTERVAL 6 HOURS
```

- In Measure G09 and later PVUs, to build an OSS file pathname journal segment and have it appended to a Measure data file at measurement shutdown:

```
+ START MEASUREMENT MDATA, OSS
```

- In Measure G11 and later PVUs, to build the SQL name journal segment and have it appended to a Measure data file at measurement shutdown:

```
+ START MEASUREMENT MDATA, SQL
```

STATUS MEASSUBSYS

This command displays this status information:

- The number of active or configured measurements and a list of the measurement-data file names
- The number of active MEASCTL processes and a list of the CPUs where they are active

- In Measure G09 and later PVUs, the current setting of several subsystem environmental settings such as the number of CIDs supported, CID table locking, and the current default setting for OSS journal segment construction
- In Measure G11 and later PVUs, the current setting for SQL journal segment construction (ON or OFF)

If you specified MEASSUBSYS as the command object (see the ASSUME (page 51) command), you can omit MEASSUBSYS from the command line.

Syntax

```
STATUS [ / OUT filename / ] [ MEASSUBSYS ]
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

Following execution of the STATUS MEASSUBSYS command, MEASCOM redirects its output to the current OUT file (typically, the terminal).

The OUT option does not affect the contents of the log file.

MEASSUBSYS

is optional if you specified it as the command object by using the ASSUME command.

Usage Note

CID stands for counter identifier. For each counter record in a processor—a counter record corresponds to a measured entity, such as a file or a process—Measure allocates and tracks the counter record through a CID. The number of CIDs supported in a processor is limited. This limit can be specified at START MEASSUBSYS time to be 32,000, 64,000, 96,000, 128,000, 192,000, 256,000, or 512,000 for each CPU. (Values of 192,000 and 256,000 are valid only in H-series and J-series RVUs, and the value of 512,000 is valid only for Measure H04, J02, and later PVUs.) The CID value reported is the CID table size. The current number of CIDs in use is reported in the CPU entity report of each processor.

Examples

- To display the active measurements and MEASCTL processes:

```
+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  1
  $DATA1.PERF.MAY04

Number of Active MEASCTL Processes = 9
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7, 8
```

- In Measure G09 and later PVUs, to show the active measurement and Measure subsystem status, including active MEASCTL processes:

```
+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  1
  $DATA1.SUBVOL.MDATA

Number of Active MEASCTL Processes = 6
  in CPU(s): 0, 1, 2, 3, 4, 5
OSS ON, Skew copy time OFF, Lock CID table OFF, CIDs = 128K/CPU
```

- In Measure G11 and later PVUs, to show the active measurement and Measure subsystem status, including active MEASCTL processes, indicating whether SQL journal segment construction is on:

```
+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  1
```

```
  $DATA1.SUBVOL.MDATA

Number of Active MEASCTL Processes = 8
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7
SQL OFF, OSS OFF, Skew copy time OFF, Lock CID table OFF, CIDs = 32K/CPU
```

- In Measure H01 and later PVUs, to show information about the MEASIP processes:

```
+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  3
  $SYSTEM.SYSTEM.ZASPAA
  $SYSTEM.SYSTEM.ZASPZOO
  $SYSTEM.MEASURE.DOFF3

Number of Active MEASCTL Processes =  8
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7
Number of Active MEASIP Processes =  8
  in CPU(s): 0, 1, 2, 3, 4, 5, 6, 7

OSS OFF, SQL OFF, Skew copy time OFF, Lock CID table ON,
CIDs = 64K/CPU
```

- In Measure J01 and later PVUs, a MEASIP process is active on each IPU. This is reflected in the total number of running MEASIP processes:

```
+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements =  0

Number of Active MEASCTL Processes =  4
  in CPU(s): 0, 1, 2, 3
Number of Active MEASIP Processes =  8
  in CPU(s): 0, 1, 2, 3
OSS OFF, SQL OFF, Skew copy time OFF, Lock CID table ON,
CIDs = 64K/CPU
```

STATUS MEASUREMENT

This command displays this information about currently active measurements:

- Number of entities being measured
- Number of words in system data space currently allocated for each entity type
- Start time, stop time, and the collection interval of the measurement
- Current size of the data file
- Maximum size of the data file
- Percent of the file used
- In Measure G09 and later PVUs, the indication of whether OSS journal segment construction is ON or OFF
- In Measure G11 and later PVUs, the indication of whether SQL journal segment construction is ON or OFF

Syntax

```
STATUS [/OUT filename/] [ MEASUREMENT ] data-file
```

OUT *filename*

directs command output to *filename*. If *filename* does not exist, MEASCOM creates an EDIT file by that name and writes command output to it. If *filename* does exist, MEASCOM opens the file and appends command output to it.

Following execution of the STATUS MEASUREMENT command, MEASCOM redirects its output to the current OUT file (typically, the terminal). This option does not affect the contents of the log file.

data-file

is an active measurement data file. The data file need not be accessible by the MEASCOM (through the ADD MEASUREMENT command) because MEASCOM requests the information from the Measure coordinating process (MEASMON) rather than reading it from the data file.

MEASUREMENT

is optional if you specified MEASUREMENT as the command object by using the ASSUME command.

Usage Note

- Stop time is displayed only if you explicitly specified the time in a START MEASUREMENT command.
- In Measure H02 and later PVUs, the output produced by the MEASCOM STATUS MEASUREMENT command is enhanced to display statistics for very large data files and to display whether counter data records have been suppressed for the measurement.

Examples

- To display information about the currently active measurement writing to OCT18:

```
+ STATUS MEASUREMENT OCT18
From 18 Oct 1994, 14:35:33
Cpu                       1 Entities              46 Words
Disc                     64 Entities            7552 Words
Current EOF             178 B          Maximum EOF  1556480 B
                                       Percentage Used  0.3 %
```

- In Measure G11 and later PVUs, to show the active measurement and the status of OSS and SQL journal segment construction:

```
+ STATUS MEASUREMENT MDATA
From 26 Feb 2003, 15:59:54
File                      13 Entities            1144 Words
Current EOF             8192 B         Maximum EOF       133693440 B
OSS Journal OFF    SQL Journal OFF     Percentage Used          0.00 %
```

- In Measure H02 and later PVUs, output produced by STATUS MEASUREMENT can display whether counter data record suppression was specified for the measurement. If counter data records are suppressed, the display shows “Counter Suppression ON.” Otherwise, the display shows “Counter Suppression OFF.” Here is an example with the counter suppression turned off:

```
+ STATUS MEASUREMENT MEASXX
From 16 Aug 2006, 13:52:32
Cpu                       1 Entities             380 Words
Current EOF             8192 B         Maximum EOF   1099510579200 B
OSS Journal OFF    SQL Journal OFF     Percentage Used          0.00 %
Counter Suppression OFF
```

STOP MEASSUBSYS

This command stops all measurements and the subsystem processes.

Syntax

```
STOP [ MEASSUBSYS ]
```

MEASSUBSYS

is optional if you specified MEASSUBSYS as the command object by using the ASSUME command.

Usage Notes

- The STOP MEASSUBSYS command can be executed only by a super-group user; that is, a user who has user ID (255,*).
- If you enter this command interactively while measurements are active, MEASCOM warns you about the measurements and asks you to confirm the command. If you enter this command noninteractively or from a command (OBEY) file, MEASCOM stops the subsystem without asking you for confirmation.

Related Commands

Command	Function	Page
START MEASSUBSYS	Starts the Measure subsystem	117
STATUS MEASSUBSYS	Determines whether any measurements are active	123

Example

To start MEASCOM, check the status of the Measure subsystem, then stop the subsystem:

```
43> MEASCOM
MEASURE Performance Monitor- T9086D30 - (01 MAR 95)
Copyright Tandem Computers Incorporated 1986-1995
1+ STATUS MEASSUBSYS
Number of Active (or Configured) Measurements = 0
2+ STOP MEASSUBSYS
```

STOP MEASUREMENT

This command stops the specified measurement.

Syntax

```
STOP [ MEASUREMENT ] data-file [ , NO ADD ]
   [ , { TO | AT } [ end-date , ] end-time-of-day ]
```

MEASUREMENT

is optional if you specified MEASUREMENT as the command object by using the ASSUME command.

data-file

is an active measurement data file.

If *data-file* is a disk file and the NO ADD option is not specified, MEASCOM performs an implicit ADD MEASUREMENT *data-file*, allowing you to examine the collected data (using one of the LIST commands) without first explicitly entering an ADD MEASUREMENT command.

If *data-file* is a tape file, you must copy the file to a disk file before using MEASCOM to examine the collected data.

NO ADD

prevents the implicit ADD MEASUREMENT *data-file* command from being performed when the measurement stops. Use the NO ADD option when you want to stop a measurement but do not want to look at the data file.

{ TO | AT } [*end-date* ,] *end-time-of-day*

specifies the time the measurement is to stop. TO and AT are synonymous. If you omit the TO or AT clause, MEASCOM stops the measurement immediately.

end-date

is the date the measurement is to stop, in one of these formats:

{ [*d*]*d mmm*[*yyyy*] }
{ *mmm* [*d*]*d*[*yyyy*] }

where

dd is a day of the month, a number in the range 1 through 31.

mmm is the first three letters of the month; for example: JAN, MAR, OCT.

yyyy is the year. Valid years are 1984 through 2047.

If you omit *date*, the current date is used.

end-time-of-day

is the time the measurement is to stop, in the format:

hh:*mm*[:*ss*]

where

hh is the hour (0 through 23).

mm is minutes (0 through 59).

ss is seconds (0 through 59).

If you omit *end-time-of-day*, MEASCOM stops the measurement immediately.

Examples

- To stop a measurement immediately and start the MEASFH process, which prepares the data file so you can use the LIST command to view the data:

```
+ STOP MEASUREMENT $DATA.MEAS.MAY04
```

- To stop a measurement immediately but not start the MEASFH process; thus, the data is not available to the LIST command:

```
+ STOP MEASUREMENT $DATA.MEAS.MAY04, NO ADD
```

SWAPVOL

This command specifies the swap volume where work files are created. The command applies for the remainder of the MEASCOM session or until another SWAPVOL command is executed.

Syntax

```
SWAPVOL [ $volume ]
```

$*volume*

is the name of a volume. If the volume name is omitted, the swap volume resets to the MEASCOM swap volume, which is the default.

Usage Note

By default, swap files are created on the MEASCOM swap volume, which might have insufficient space for temporary files. In this case, use the SWAPVOL command at the beginning of the MEASCOM session so MEASCOM puts its temporary files on a volume that has enough space.

SYSTEM

This command sets the default system name used in file-name expansions. Initially, the system is specified by the operating system command interpreter (TACL).

Syntax

```
SYSTEM [ \system ]
```

\system

is a system name. If omitted, the default defined by your command interpreter becomes the MEASCOM default.

Example

To change the default system name:

```
+ SYSTEM \SELL
```

TIME

This command displays the current system date and time.

Syntax

```
TIME [ \system ]
```

\system

is a system name. TIME returns the local civil time of this system. If you do not specify a system, TIME returns your local system time.

Examples

- To display the time of the local system:

  ```
  + TIME
  20 November 1994 18:02:25
  ```

- To display the time of the system \NY:

  ```
  + TIME \NY
  20 November 1994 21:02:30
  ```

VOLUME

This command sets the default device name, subvolume name, or both for file-name expansions. Initially, the default device and subvolume are specified by the command interpreter.

Syntax

```
VOLUME [   $device             ]
       [ [ $device.]subvolume ]
```

$*device*

is a device name. If you omit $*device*, the default device does not change. If you omit both $*device* and *subvolume*, the defaults specified by the command interpreter become the MEASCOM defaults.

subvolume

is a subvolume name. If you omit *subvolume*, the default subvolume does not change. If you omit both $*device* and *subvolume*, the defaults specified by the command interpreter become the MEASCOM defaults.

Examples

- To change the default volume and subvolume:

  ```
  + VOLUME $DATA2.MAINT
  ```

- To change the default volume and subvolume to those defined by your command interpreter:

  ```
  + VOLUME
  ```

WARNINGS

This command specifies which warning messages are displayed. Warning messages are identified by the word WARNING and a number in the range 3000 through 3049. You can have MEASCOM display all warnings except those you specify or suppress all warnings except those you specify.

Syntax

```
WARNINGS { DISPLAY  { ALL | warn-num [, warn-num ...]}}
         { SUPPRESS { ALL | warn-num [, warn-num ...]}}
```

DISPLAY ALL

causes all warnings to be displayed except those specified by one or more succeeding WARNINGS SUPPRESS commands. The default is WARNINGS DISPLAY ALL.

SUPPRESS ALL

suppresses all warnings except those specified by one or more succeeding WARNINGS DISPLAY commands.

warn-num

is the number of a warning (3000 through 3049).

Usage Note

To suppress comments (identified with the word COMMENT), use the COMMENTS command.

Example

To suppress warning message 3013:

```
+ WARNINGS SUPPRESS 3013
```

!

This command retrieves a previously executed command from the history buffer and executes it again.

Syntax

```
!  [  number ]
   [ -number ]
   [  text   ]
```

number

is the number of the line in the history buffer that contains the command to be reexecuted. (See the first example in Examples.)

-number

is the number of history buffer lines to subtract from the current line to arrive at the command to be reexecuted. (See the second example in Examples.)

text

is a text string. The most recently entered command that starts with the text string is retrieved. (See the third example in Examples.)

If no option is specified, the last command entered is reexecuted. (See the fourth example in Examples.)

Related Commands

Command	Function	Page
HISTORY	Displays the history buffer	61
FC	Retrieves, modifies, and reexecutes a previous MEASCOM command	59

Examples

- To reexecute the command on history buffer line 15:

  ```
  21+ ! 15
  ```

- To reexecute the command in the history buffer 3 lines before the current line (in this case, line 24 is the current line, so the command from history buffer line 21):

  ```
  24+ ! -3
  ```

- To reexecute the most recently entered command that begins with the text string SE (for example, SET REPORT FORMAT BRIEF):

  ```
  25+ ! SE
  ```

- To reexecute the last command entered:

  ```
  28+ !
  ```

3 Entities and Counters

This chapter describes the Measure entity types and their associated counters:

Table 3-1 Measure Entity Types

Entity Type	Measured System Resource	Page
CLUSTER	Fiber Optic Extension (FOX) and ServerNet/FX cluster traffic. Does not apply to H-series and J-series RVUs.	150
CONTROLLER	I/O activity on device controllers on HP NonStop K-series servers only. (Disabled in Measure D30 (T9086ACV), D44 (T9086ACR), and D45 (T9086ACQ). To enable it, use the define on page 154.)	154
CPU	Processors.	156
DEVICE	I/O devices, such as tape drives. Does not include disks, communication lines, subdevices, and asynchronous terminals. Each of these devices has its own entity type. For G-series and later RVUs, this entity applies to Tape and Open SCSI IOPs.	171
DISC or DISK	Disks.	180
DISCOPEN or DISKOPEN	I/O operations performed by a single opener process on a specified disk file (physical file access).	198
DISKFILE	I/O operations performed by all opener processes on a specified disk file (physical file access).	207
EXTNAMES	A file containing an external record format for ANSI SQL. EXTNAMES, not a formal Measure entity in the command interface, is a record format and structured output file that results from a LISTOSSNAMES command.	n.a.
FILE	I/O operations performed by a local user process on an explicitly opened file (logical file access).	216
LINE	Communications lines, such as SNAX/XF and X25AM lines.	236
NETLINE	Network communications lines, such as Expand lines.	243
OPDISK	Optical disks (D-series and Himalaya K-series servers).	251
OSSCPU	OSS elements in each system processor.	256
OSSNAMES	OSS file pathname mapping. The OSSNAMES file, not a formal Measure entity in the command interface, is a record format and structured output file that results from a LISTOSSNAMES command. The file is used for linking internal format OSS PATHID and CRVSN data in other record types to the external OSS file pathname translation.	n.a.
OSSNS	OSS name server processes.	266

Table 3-1 Measure Entity Types *(continued)*

Entity Type	Measured System Resource	Page
PROCESS	Processes.	272
PROCESSH	Process code range histograms.	291
SERVERNET	ServerNet addressable controllers (SACs) and interprocessor communication (IPC) on HP NonStop S-series and NS-series servers. In G08 Measure, this entity applies to HP NonStop ServerNet Cluster and remote interprocessor communication (RIPC).	302
SQLPROC	SQL/MP or SQL/MX processes.	313
SQLSTMT	SQL/MP or SQL/MX statements within the SQL process.	319
SYSTEM	Network traffic through Expand line handlers (local end only).	330
TERMINAL	Terminal and subdevice I/O.	333
TMF	HP NonStop Transaction Management Facility (TMF) transactions.	338
USERDEF	User-defined application instrumentation.	342

Counters Overview

Interpreting Counter Values

The Measure performance monitor uses various types of counters to accumulate data. When counters are displayed in a MEASCOM report, the value displayed is affected by the current REPORT RATE and REPORT FORMAT attributes.

REPORT RATE Setting	Counter Values Are Shown as...
OFF	Raw data; no mathematical operations are performed on them.
ON	Averages or percentages; specific interpretation depends on the counter type.

To create structured files of Measure data that other products, such as Enform, can access, use the REPORT FORMAT STRUCTURED attribute. IF REPORT FORMAT is STRUCTURED, counter values are written to the output file as raw data. REPORT FORMAT STRUCTURED is not affected by the REPORT RATE setting.

NOTE: Counter descriptions in this section assume REPORT FORMAT NORMAL or BRIEF and REPORT RATE OFF.

Measure scales displayed values to fit within 10 characters, including commas and periods. Units for the formatted value are indicated by these symbols:

#	indicates decimal units of ones
K	indicates decimal units of thousands (kilo)
M	indicates decimal units of millions (mega)
G	indicates decimal units of billions (giga)
T	indicates decimal units of trillions (tera)

For example, a megabyte would be displayed as 1048576 #.

Accumulating Counters

Accumulating counters count the number of bytes passing to or from a file, disk, or other entity. A value is added to the counter whenever data is transferred to or from the entity.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT FORMAT is STRUCTURED	The number of bytes transferred during the time interval	
REPORT RATE is OFF	The number of bytes transferred during the time interval	#, K, M, G, or T
REPORT RATE is ON	The number of bytes transferred for each second during the time interval	per-second values: /s, K/s, M/s, G/s, or T/s[1]

1 ZMS style only. No units are displayed in the legacy format. It is assumed to be /s with no scaling, and might overflow.

Measure supports both 32-bit and 64-bit accumulating counters. In G-series RVUs, some accumulating counters exist in two versions, a 32-bit version and a 64-bit version. The 64-bit version is distinguished by a counter name ending in the characters -F. For example, SENT-BYTES is a 32-bit counter; SENT-BYTES-F is the corresponding 64-bit counter. In H-series and J-series RVUs, all such accumulating counters have 64 bits, and no counter in ZMS style records has a name ending in -F. For example, on H-series and J-series RVUs SENT-BYTES is a 64-bit counter. SENT-BYTES-F is no longer a counter name in ZMS style but remains available to applications requesting data in legacy style.

For an example of an accumulating counter, see the FILE entity counter MESSAGE-BYTES (page 227).

Average Queue Time Counters (ZMS style Only)

Average queue time counters measure the average time that each item visiting a queue spends there. This derived counter is formed by dividing a queue counter by a corresponding incrementing counter for items visiting the queue.

The implementation of the queue counter affects whether the average queue time counter includes the average service time to process the event and whether it is exclusively queueing time, as indicated by the counter's suffix:

Suffix	Includes Service Time
-AQT	No
-ART	Yes

Average queue time counters are not affected by setting RATE ON or OFF. Units are displayed as ms, sec, hr, day, or wk.

For an example of an average queue time counter, see Home-Trans in the ZMS style report in TMF (page 338).

Average Service Time Counters (ZMS style Only)

Average service time counters measure the average time to satisfy each request of a resource. This derived counter is formed by dividing a busy or queue-busy counter for a resource by a corresponding incrementing counter for requests of the resource. Average service time counters are identified in displays by the suffix -AST.

Average service time counters are not affected by setting RATE ON or OFF. Units are displayed as ms, sec, hr, day, or wk.

Busy Counters

Busy counters measure the time a resource is busy. The counter uses a busy or idle state word and a 64-bit busy-time accumulator to record the cumulative busy time in microseconds.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT RATE is OFF.	The time busy during the interval	ms (milliseconds), sec, hr, day, or wk
REPORT RATE is ON.	The percent of time busy during the interval	a percentage of the interval length
REPORT FORMAT is STRUCTURED.	The time busy expressed in microseconds	

For an example of a busy counter, see the CPU entity counter CPU-BUSY-TIME (page 161).

In H-series and J-series RVUs, there are two types of busy counters:

- BUSY counters, maintained with conventional timers
- TCELLBUSY counters, maintained with timer cells

For an overview of timer cells and related counters, see Timer Cell Counters (page 139).

Elapsed Counters

Elapsed counters measure start-to-finish time for an event.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT RATE is OFF.	The elapsed time	ms, sec, hr, day, or wk
REPORT RATE is ON.	The average elapsed time for each event	ms, sec, hr, day, or wk
REPORT FORMAT is STRUCTURED.	The elapsed time expressed in microseconds	

For an example of an elapsed counter, see the SQLSTMT entity counter ELAPSED-SORT-TIME (page 326).

Incrementing Counters

Incrementing counters count events. The counter is advanced each time the event occurs.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT FORMAT is STRUCTURED.	The number of increments during the time interval	
REPORT RATE is OFF.	The number of increments during the time interval	#, K, M, G, or T
REPORT RATE is ON.	The number of increments for each second during the time interval	per-second values: /s, K/s, M/s, G/s, or T/s

Measure supports both 32-bit and 64-bit incrementing counters. In G-series RVUs, some incrementing counters exist in two versions, a 32-bit version and a 64-bit version. The 64-bit version is distinguished by a counter name ending in the -F. In H-series and J-series RVUs, all counters that had existed in two versions now exist in only one, 64-bit version. Counter names ending in -F do not exist in ZMS style but remain available to applications requesting data in legacy style.

For an example of an incrementing counter, see the CLUSTER entity counter MESSAGES RECEIVED.

Max Queue Counters (Legacy Style Only)

Max queue counters track the maximum length of a queue. Report format options do not affect max queue counters.

Due to the sharing of counter records among up to 64 measurement requests, there is no way to reset max queue values. As records exist in the system longer, the value of max queue counters diminishes. The reported value becomes irrelevant to the measured period.

In G11 and later Measure PVUs, max queue counters are no longer tracked in the instrumentation. They are removed from new format records, and are reported as 1 in Legacy Style records and displays of G11 or later data.

For an example of a max queue counter, see the PROCESS entity counter MAX-MQCS-INUSE (page 283).

Max Value Counters

Max value counters track the maximum value of a given measurement during the measurement interval. Report format options do not affect max value counters.

The only use of a max value counter is Max-Lockwait-time in the DISCOPEN record. Units are formatted as ms, sec, hr, day, or wk.

For an example of a max value counter, see the DISCOPEN entity counter MAX-LOCKWAIT-TIME (page 204).

Queue Counters

Queue counters measure the total amount of time that all queued elements spend on the queue.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT RATE is OFF.	The total amount of time all elements spent on the queue, expressed in seconds or milliseconds. For example, if one process is queued for two seconds and another process is queued for three seconds, the total queue time is five seconds.	ms, sec, hr, day, or wk
REPORT RATE is ON.	The average number of queued elements (AQL) during the time interval. For queue counters, RATE ON is not simply an average or percentage of RATE OFF. RATE ON provides average queue length, not average time spent on the queue.	AQL (average queue length) in decimal units
REPORT FORMAT is STRUCTURED.	The total amount of time all elements spent on the queue, expressed in microseconds.	

For an example of a queue counter, see the CPU entity counter CPU-QTIME (page 161).

In H-series and J-series RVUs, there are two types of busy counters:

- QUEUE counters, maintained with conventional timers
- TCELLQUEUE counters, maintained with timer cells

For an overview of timer cells and related counters, see Timer Cell Counters (page 139).

Queue-Busy Counters

Queue-busy counters measure the time during which a resource accessed through a queue is busy. They are used to measure device busy time in the ServerNet environment where multiple requests (I/Os) can be queued to a device at the same time.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT RATE is OFF.	The total amount of time the queue was busy during the interval, independent of how many requests were on the queue at any time. For example, for two processes placed on the queue simultaneously, if both processes are queued for one second and one of the processes remains queued for two more seconds, the total queue-busy time is three seconds.	ms, sec, hr, day, or wk
REPORT RATE is ON.	The percent of time busy during the interval.	a percentage of the interval length
REPORT FORMAT is STRUCTURED.	The total amount of time that there were elements on the queue, expressed in microseconds.	

For an example of a queue-busy counter, see the DEVICE entity counter READ-QBUSY-TIME (page 177).

In H-series and J-series RVUs, there are two types of queue busy counters:

- QBUSY counters, maintained with conventional timers
- TCELLQBUSY counters, maintained with timer cells

For an overview of timer cells and related counters, see Timer Cell Counters (page 139).

Response Time Counters

Response time counters measure the interval, in microseconds, between a read from a terminal and the subsequent write to it. Whenever a value is added to a response time counter, a corresponding transaction counter is advanced.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT RATE is OFF.	The accumulated response time	ms, sec, hr, day, or wk
REPORT RATE is ON.	The average response time for each transaction	ms, sec, hr, day, or wk
REPORT FORMAT is STRUCTURED.	The accumulated response time in microseconds	

In reports with RATE ON, response time counters are a special case of queue time counters in that the accumulated response time value is divided by a transaction count.

For an example of a response time counter, see the TERMINAL entity counter RESPONSE-TIME (page 337).

Sampling Counters

Sampling counters measure the approximate busy time for PROCESSH code ranges. At random time intervals, the Measure subsystem activates a sampling interrupt that examines the interrupted process. If the process is under PROCESSH measurement, the sampling procedure increments

the process-busy-samples counter, increments the code-space-busy-samples counter if a measured code space is busy, and increments the appropriate code-range-busy-samples counter.

Attribute Setting	Output Value Is...	Units Displayed As...
REPORT FORMAT is STRUCTURED.	The number of sampling interrupts that found the measured code executing.	
REPORT RATE is OFF.	The number of sampling interrupts that found the measured code executing.	#, K, M, G, or T
REPORT RATE is ON.	The percent of sampling interrupts that found the measured code space executing.	percentage of total samples for the code space

For an example of a sampling counter, see the PROCESSH entity counter PROCESS-BUSY-SAMPLES (page 297).

Snapshot Counters

Snapshot counters return a measurement value taken at a specific moment, such as at the start or end of a measurement. No computation is done on the returned value; it is simply reported. Report format options do not affect snapshot counters.

Snapshot counters are not affected by setting RATE ON or OFF. Units are displayed as #, K, M, G, or T.

For examples of snapshot counters, see the DISC entity counters STARTING-FREE-SPACE and ENDING-FREE-SPACE (page 191).

Syslink Counters

Syslink counters measure the time required to complete a message link to a remote system. Whenever a value is added to a syslink counter, a corresponding link counter is advanced. The syslink counter is a special case of a response time counter where the transaction is a message link.

Attribute Setting	Output Value Is...
REPORT RATE is OFF.	The link time in seconds
REPORT RATE is ON.	The average link time (in seconds) for each transaction
REPORT FORMAT is STRUCTURED.	The link time in microseconds

For examples of syslink counters, see the SYSTEM entity counters LINKS (page 332) and LINK-TIME (page 332).

Timer Cell Counters

The dynamics of loose processor synchronization in HP Integrity NonStop NS-series systems (which run H-series and J-series RVUs) make the standard time-of-day functions less accurate than on systems running G-series RVUs when viewed at microsecond granularity.

For very fine resolution measurements, events that can complete within the context of a dispatch or a simple message exchange between two processes in the same processor, measured time values might be distorted. This is not an issue for longer running timers that encompass the completion of interprocessor messages or I/O. The degree of error in such measurements is marginal relative to the length of the event.

To provide more accurate timing services for Measure and other applications, the H-series and J-series architecture provides a mechanism called timer cells. This mechanism makes it possible to keep accurate fine granularity timers without incurring the cost of loose processor

synchronization on all timer updates. Timer cell values are synchronized only when copied at intervals or reported to the requesting application.

The timer cell counter types are TCELLBUSY, TCELLQUEUE, and TCELLQBUSY. They operate under the same instrumentation logic as the BUSY, QUEUE, and QBUSY counter types, respectively. Measure uses the new counter types for several predefined entities. You can also specify the new types for user-defined (USERDEF) entities.

Identifying Data File Errors

The first field of the output record for each measured entity type indicates whether a problem prevented measurement or record allocation.

Table 3-2 Legacy Style Error Field Values

Value	Meaning
0	No problem.
1	The data record was not allocated because the maximum number of entries allowed under concurrent measurement was exceeded.
2	The data record was not allocated because the maximum counter space allowed was exceeded.
3	The entity was not measured because code sampling was already in progress. Applies to PROCESSH entity type only.
4	The entity was not measured because of a MEASCTL internal error. The XPTR, an internal addressing structure, could not be found.
65535	A field overflowed in an accumulating or incrementing counter. The data record is written, but the capacity of one or more 32-bit fields was exceeded. Accurate values are contained in equivalent 64-bit fields. This is a warning, not an error.
65534	(G-series RVUs) Byte-count field overflow. The capacity of one or more 16-bit byte-count fields has been exceeded. See the equivalent 32-bit fields for byte-count data. This is a warning rather than an error. It applies to the DISC entity type only.

Values of 0 or -1 indicate that there is data in the record. If you have older applications that determine the presence of data by testing for 0, you might need to change those applications. For example, test for a positive value, which indicates an error condition.

Table 3-3 ZMS style Format Error Field Values

Value	Meaning
ERROR = 0	No errors or warnings regarding data in the record.
ERROR > 0	Error occurred, data is not present. The object to measure is identified but a counter record was not started because of: 1 = ERR^NOCIDENTRY — Data record error: CID table overflow 2 = ERR^NOCOUNTERSPACE — Data record error: Counter space overflow 3 = ALREADY^INUSE — Data record error: Sampling conflict 4 = ERR^XPTR^INSTALL — Data record error: Measure internal error
ERROR < 0	Warning, data is present but with noted exceptions. For the specific cause, see the remaining values:
.<1>	Numeric overflow. In totals record accumulation, one or more counter fields overflowed.
.<2>	Subsystem version mismatch. The Measure PVU that formatted the record is earlier than that of the instrumented product subsystem that produced this data record. There is more data in the internal record than MEASFH can format. Update the Measure PVU on the measured system.

Table 3-3 ZMS style Format Error Field Values *(continued)*

Value	Meaning
.<3>	Template-version mismatch. The application requesting the data record is compiled with an older version of MEASDDLS, or the structured data file was created with a record template older than the MEASFH version that produced the record. More data is available for reporting but could not be passed in the interface. Update the application or convert the structured data file to the new size.
.<4:11>	Reserved for future use.
.<12>	ZMSPROC record only, PROCESSH instrumentation for this process is delayed.
.<13>	ZMSPROC record only, USERDEF instrumentation for this process is delayed.
.<14>	ZMSPROC record only, SQL instrumentation for this process is delayed.
.<15>	ZMSPROC record only, OSSNS instrumentation for this process is delayed.

Common Entity Header Fields

These fields appear in the header of all entity records and are defined once here instead of being repeated in every entity description.

DDL Header Fields (Legacy Style)

```
RECORD entity. FILE is "entity" ENTRY-SEQUENCED.
 02 error                       type binary 16 unsigned.

* time items:
 02 from-timestamp              type binary 64.
 02 to-timestamp                type binary 64.
 02 delta-time                  type binary 64.

* measurement identification items:
 02 system-name                 type character 8.
 02 os-version.
    03 letter                   type character 1.
    03 number                   type binary 8.
 02 loadid                      type character 8.
 02 load-id                     redefines loadid.
    03 prefix-id                type character 5.
    03 interval-id              type character 3.
 02 cpu-num                     type binary 16 unsigned.
```

DDL Header Fields (ZMS Style)

NOTE: You should always use TEMPLATE-VERSION (page 143) to get the external records corresponding to a format that you can handle.

There are three ways to do this:

- Use MEASCOM, which uses the template version for the current version.
- From a user program, call `Measread_diff_()` with the requested template version.
- For structured files, use a pre-created structured file, with a record length corresponding to the requested template version for the entity in question. When data collected on systems of differing product versions are stored in the same structured files, the template version for all records will be the one used to create the structured file. The template version from the first record will be used to format all records in the file.

If there are fields in the data file that do not exist in the version indicated by the requested template version, they will not be present in the requested records.

If there are fields in the version indicated by the requested template version that do not exist in the data file, they will be present in the requested records, filled with zeros, and you can use the DATA-VERSION (page 144) field to judge if those zero-filled fields are valid or not.

```
RECORD zms<i>entity</i>. FILE is "zms<i>entity</i>" ENTRY-SEQUENCED.
 02 loadid                      type character 8.
 02 load-id                     redefines loadid.
    03 prefix-id                type character 4.
    03 interval-id              type character 4.
 02 system-name                 type character 8.
 02 cpu-num                     type binary 16 unsigned.
 02 os-version.
    03 letter                   type character 1.
    03 number                   type binary 8.
 02 reserved                    type character 4.
 02 object-uid                  type character 16.
 02 from-timestamp              type binary 64.
 02 to-timestamp                type binary 64.
 02 delta-time                  type binary 64.
 02 template-version            type binary 64.
 02 format-version.
    03 letter                   type character 1.
    03 number                   type binary 8.
 02 data-version.
    03 letter                   type character 1.
    03 number                   type binary 8.
 02 subsystem-version           type binary 16 unsigned.
 02 error                       type binary 16.
end
```

ERROR

Indicates whether an error prevented Measure from collecting data or writing an output record for this entity. For values, see Identifying Data File Errors (page 140).

OBJECT-UID

(ZMS Style only) Reserved for future use.

FROM-TIMESTAMP

Starting time of the measurement or transient entity. Specified as Julian-based, local civil time (LCT).

TO-TIMESTAMP

Ending time of the measurement or transient entity. Specified as Julian-based, local civil time (LCT).

DELTA-TIME

Duration of the measurement, in microseconds.

SYSTEM-NAME

Name of the system on which the measurement was taken.

OS-VERSION

Version ID of the operating system when the measurement was taken. This field is divided into two subfields: LETTER and NUMBER.

LOADID

Redefined. See LOAD-ID.

LOAD-ID

Used only for structured files. If the LOADID clause of the MEASCOM LIST command is used when this record is written to the structured file, this field contains a unique identification string for the record. If the LOADID clause is not used, this field contains blanks. The default is blanks (no LOADID).

A value provided by the user when requesting structured record output. LOADID is a string 8 characters long that can contain letters, numbers, carets (^), hyphens (-), and underscores (_). The first character must be a letter. If fewer than 8 characters are specified, the string can be space-filled or can contain a generated interval sequence number.

When used with the LISTALL command, if LOADID is 5 characters or fewer, the LOADID is generated with an appended sequence number indicating the interval the data record represents. Date intervals are numbered sequentially, from one. If LOADID is 3 characters or fewer, the generated LOADID might contain spaces.

If you want interval numbering in LISTALL requests for ZMS style structured records, specify a 4-character LOADID. If you use a 5-character LOADID, the fifth character is put in the INTERVAL-ID portion of the field. Generated LOADIDs are not often used for LISTALL data. Applications that use this feature should be evaluated to determine whether 4-character or 5-character LOADIDs should be specified in the future.

CPU-NUM

Number of the CPU on which the measurement was taken. For interprocessor communications (IPC) records, CPU-NUM is MYCPU.

TEMPLATE-VERSION

(ZMS Style only) The record template version from MEASDDLS for the format of the external record. When data collected on systems of differing product versions are stored in the same structured files, the template version for all records is the one used to create the structured file. The template version from the first record will be used to format all records in the file.

FORMAT-VERSION

(ZMS Style only) The Measure product version of the program that formatted the entity report or structured record.

DATA-VERSION

(ZMS Style only) The Measure product version of the system that collected the Measure data file that produced this record, and the MEASFH product version compatible with the data file.

SUBSYSTEM-VERSION

(ZMS Style only) This value is set to 1 for the initial H-series and J-series versions of all subsystems. If the value is greater than one, consult the subsystem documentation for information about changes that affect the Measure counters. In G-series RVUS, the value is 0.

Measure Support for Open System Services (OSS)

For more information on the interoperability between the OSS and Guardian environments, see:

- *Open System Services Library Calls Reference Manual*
- *Open System Services Management and Operations Guide*
- *Open System Services Shell and Utilities Reference Manual*
- *Open System Services System Calls Reference Manual*
- *Open System Services User's Guide*

Handling of OSS File Pathnames

- OSS file pathnames are always entered and displayed in double quotation marks (" ") and are case sensitive. OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files within that immediate directory are included. Files contained within directories subordinate to the specified directory are not included.
- To support both the Guardian and OSS environments, the use and intended meaning of pname is standardized. pname is a long-established abbreviation for a Guardian process name, as well as an OSS shell command:
  - pname in double quotation marks ("pname") is an OSS file pathname.
  - pname preceded by a dollar sign ($pname) or alone (pname) is a Guardian process name.
- OSS file pathnames can be long (up to 1023 characters in length). In displaying these long OSS file pathnames, Measure displays the first 80 characters and wraps any remaining characters to the next line. Measure displays the name in 80-character blocks until the entire OSS file pathname is displayed. Additionally, if you use the cut-and-paste feature on these longer OSS file pathnames, you might need to cut and paste the name in sections to avoid picking up extraneous carriage return characters from the end of each 80-character line.
- The POSIX standards allow the use of nonprintable (nondisplayable) characters in OSS file pathnames. However, names that use these characters might be handled differently according to differences in the user interface, workstation, and terminal environments. Neither the OSS utilities nor Measure deal with this issue.
- Although Measure G09 and later PVUs provide entity support for OSS file pathnames, be cautious when you use OSS file pathnames in measurement specifications because pathnames can overrun the buffer limit. For detailed measurements, break the measurement into several separate measurement configurations. Measure supports up to 64 concurrent measurements.
- Do not confuse the meanings of OSSPID and PID:
  - OSSPID is a numeric value displayed in many Measure entity reports, and PID is a Guardian process identifier consisting of CPU, PIN.
  - PID is used extensively in the Measure command interface, but the Measure command interface does not support OSSPID.
- OSS file pathnames are null-terminated strings. In Measure, the length of a string field is required and includes the null character

Guardian File-Name Reuse

Guardian file-name values are based on an internal value or cell in the OSS file system, commonly referred to as an inode number. When a file and all references to it are deleted, reuse of the inode number and the file name can occur.

When a file is created on disk, a unique creation version serial number (CRVSN) is assigned to the file. The CRVSN provides a mechanism to distinguish between various instances of a Guardian file name. Measure uses the mechanism to avoid file-name resolution problems due to inode number reuse. If Guardian file name and CRVSN values do not match those currently associated in the OSS name server data, assume that reuse occurred and that no OSS file pathname will be displayed.

If Guardian file name and CRVSN values do not match the values currently associated in the OSS Name Server data, it is assumed that file-name reuse occurred. When file-name reuse occurs, no associated OSS file pathname is displayed, and this message is generated: `*** Translation Not Available ***`.

Handling of the Creation Version Serial Number (CRVSN)

The Creation Version Serial Number (CRVSN) is a 48-bit value. Measure displays the CRVSN as a decimal string of up to 15 characters following the Guardian name (for example, $DATA01.ZYQ00001.Y00003D2:**340359**). Measure stores the CRVSN value in data records as a string of six characters (48 bits).

Measure Identification (MID)

OSS file pathnames are contained within a data structure called a PATHID. PATHIDs are internal and are passed around in control blocks. When an OSS file pathname is externalized, it is passed to the OSS Name Server for file pathname translation.

To support reporting of OSS file pathnames, Measure captures PATHID and CRVSN information for each OSS file pathname and stores the information in Measure data. The association of a PATHID and a CRVSN value in Measure data is a Measure ID (MID). The addition of the MID data does not increase the size of your Measure data file because the MID information and representation of the internal file name are compressed.

OSS Journal Segment

The OSS Journal Segment is a set of OSS file pathname translations captured for Measure data collection. It provides OSS file pathname translations for data records in the Measure data file when such translations are no longer available from the OSS Name Server. The OSS Name Server cannot translate the OSS file pathname due to time constraints and the relocation of the data file to another node or network.

The main feature of OSS file pathname support in Measure is the creation of an OSS journal segment. The OSS journal segment ensures portability of OSS file pathnames translation between systems and over time. It also eliminates the need to access the OSS Name Server of the system being measured to analyze or report on OSS objects in Measure data.

OSS journal segment construction is not required when you use or view OSS file pathnames in Measure data or when you analyze Measure data on the local system. OSS journal segment construction is required when OSS file pathname translation must be available to reporting applications in structured Measure data output.

Measure builds the OSS journal segment at measurement shutdown and inserts it in the Measure data file. The journal segment is loaded into memory and attached to an application process or MEASCOM when a Measure data file is opened for analysis. This feature is optional. The default is no journal segment.

Entity Report Formats

The length of entity reports varies because the display of OSS file pathnames might add a variable number of lines to a report if the name is longer than a single line can display.

To control scrolling of Measure entity reports, use the PAGESIZE command. PAGESIZE lets you specify the exact number of lines to display before another prompt is issued. For details, see PAGESIZE (page 101).

Measure Support for ANSI SQL Names

SQL/MX objects such as tables and indexes have ANSI SQL names in addition to their underlying Guardian names. Measure G11 and later PVUs can provide ANSI SQL names in certain Measure output.

For more information on ANSI SQL names and SQL/MX, see the *SQL/MX Reference Manual* and the *SQL/MX Installation and Management Guide*.

Handling of ANSI SQL Names

- ANSI SQL names are always displayed and entered in single quotes (') and are case-sensitive. ANSI SQL delimited identifiers enclosed in double quotes (") are permitted inside the single quotes. For example:

```
'Catalog_12.Schema_34.Table_56'
'Catalog_12."A.B.C PARTITION D$"" <>".Table_56'
```

NOTE: When an ANSI SQL name is passed as an argument to a callable procedure, it should not be enclosed in single quotes. See Chapter 4: Measure Callable Procedures (page 349).

- ANSI SQL names can refer to specific objects or to a set of objects within a specific schema and catalog. If a catalog is specified, only files within that immediate catalog are included. Files contained within schema subordinate to the specified catalog are not included.
- Measure compresses any white space (one or more blanks) within single quotes that is not within double quotes.
- Measure wraps displayed ANSI SQL names from one line to the next, as needed. If you cut and paste a wrapped ANSI SQL name from a Measure display, carriage return characters might be included that are not part of the name. To avoid this, cut and paste the name one line at a time.
- To enter commands longer than 132 characters, continuation characters ("&") must be used to break the command input lines into smaller pieces. Entering a text string terminated by the ampersand character will cause the next input line to be treated as a continuation of the input. Successive input lines entered in this fashion are treated as if they were entered as one string. For example:

```
+ ADD DISKFILE 'long_long_long_long_long_long_long_&
& long_catalog.short_schema.short_object PARTITION&
& short_partition2'
```

- ANSI SQL identifiers can be long (up to 128 characters). The maximum length of an identifier in delimited form is 258 characters (two double-quotes to delimit the identifier, plus 256 double-quote characters to express an identifier consisting of 128 double-quote characters).
  - An ANSI SQL object name without a name space qualifier has a maximum undelimited length of 386 characters (three undelimited identifiers of 128 characters plus two periods). In delimited form, an object name has a maximum length of 776 characters (three delimited identifiers of 258 characters, plus two periods).
  - An ANSI SQL object name with a name space qualifier has a maximum undelimited length of 393 characters (7 additional characters for the MODULE keyword, plus a

space). In delimited form, an object name has a maximum length of 783 characters (7 plus 776).

- An SQL/MX partition name without a name space qualifier has a maximum undelimited length of 525 characters (four undelimited identifier fields of 128 characters, plus two periods, plus nine characters for the PARTITION keyword, plus two spaces). In delimited form, a partition name has a maximum length of 1045 characters (four delimited identifiers of 258 characters, plus two periods, plus nine characters for the PARTITION keyword, plus two spaces).
- An SQL/MX partition name with a name space qualifier has a maximum undelimited length of 531 characters (6 additional characters for the TABLE or INDEX keyword followed by a space). In delimited form, a partition name has a maximum length of 1051 characters (6 plus 1045).
- The maximum ANSI SQL name length is 1051 characters for a maximum length partition name expressed in delimited form.

SQL/MX does not support all of these theoretical limits. And when SQL/MX returns a syntax error, Measure reports a syntax error. See the *SQL/MX Reference Manual*.

SQL Journal Segment

A measurement data file can include the SQL journal segment for use in translating between ANSI SQL, Guardian names, and ANS UIDs. The SQL journal segment ensures portability of SQL name translation between systems and over time.

The journal segment includes information such as the SQL/MX object and partition names, Guardian name, ANS UIDs, ANSI name space, and SQLMID (an internal Measure handle that represents a particular instance of the SQL/MX file).

SQL journal segment construction is not required when you analyze Measure data on the system where the measurement occurred. SQL journal segment construction is required when name translation must be available to reporting applications in structured Measure data output (see MEASLISTEXTNAMES (page 411)).

Measure builds the SQL journal segment during the measurement and inserts it into the Measure data file during measurement shutdown. The journal segment is loaded into memory and attached to an application process or MEASCOM when a Measure data file is opened for analysis.

This feature is optional. The default is no journal segment.

ANSI SQL object names can be fully qualified or partially qualified. If partially qualified, the omitted catalog and schema fields are resolved using MEASCOM session environment values (see the SQLCATALOG (page 114) and SQLSCHEMA (page 114) commands).

Entity Report Formats

The length of entity reports varies because the display of ANSI SQL names might add a variable number of lines to a report if the name is longer than a single line can display.

To control scrolling of Measure entity reports, use the PAGESIZE command. PAGESIZE lets you specify the exact number of lines to display before another prompt is issued. For details, see PAGESIZE (page 101).

Measure Support for DLLs

Measure G12 and later PVUs support measurement of DLLs.

Code Space Specification

With the introduction of DLLs, which are implemented in position-independent code (PIC), this traditional model of code-space is no longer applicable:

- User code
- User library
- System code
- System library

Except for legacy TNS code (including accelerated TNS code), a code space is no longer required when adding PROCESSH entities and is no longer displayed by commands that show PROCESSH data.

These MEASCOM commands are affected by this change:

- The ADD PROCESSH command no longer requires you to specify a code space, except for legacy TNS code (including accelerated TNS code).
- Output of these commands is now organized around code files rather than code spaces, except for legacy TNS code (including accelerated TNS code):
  - INFO PROCESSH
  - INFO MEASUREMENT
  - LIST PROCESSH
  - LISTALL PROCESSH

Aggregate Data Measurement

In order to collect, analyze and display information about DLL execution, Measure now supports the collection and display of aggregate data about library code across all processes executing the same version of that code. This enhancement required the TOTALS attribute to be extended to the PROCESSH entity through these MEASCOM commands:

- LIST *entity-type* (page 68)
- LISTALL *entity-type* (page 90)
- RESET REPORT (page 103)
- SET REPORT (page 108)

You can accommodate these settings:

- TOTALS ONLY

  Only aggregate counts are displayed.

- TOTALS INCLUDE

  Both per-process and aggregate counts are displayed.

- TOTALS SUPPRESS

  No aggregate counts are displayed.

If only one process is executing code specified in a measurement, per-process and aggregated counts are identical.

Additional Information About DLLs in This Manual

Command Interfaces

- LIST *entity-type* (page 68)
- LISTALL *entity-type* (page 90)
- LISTACTIVE *entity-type* (page 85)
- RESET REPORT (page 103)
- SET REPORT (page 108)
- SHOW REPORT (page 116)

Information Displays

- INFO *entity-type* (page 62)
- INFO MEASUREMENT (page 64)
- LIST *entity-type* (page 68)
- LISTACTIVE *entity-type* (page 85)
- LISTALL *entity-type* (page 90)

Entity Specifications

- PROCESSH (page 291)

External Record Definitions

- See the PROCESSH external record definition in DDL Record for PROCESSH Entities (ZMS Style) (page 296).
- See the zmsproch-id. external record definition in DDL Record for PROCESSH Entities (ZMS Style) (page 296).

Callable Procedures

- MEAS_CODERANGENAME_DEMANGLE_ (page 358)

Accessing ZMS style Records (MEASDDLZ)

The MEASDDLZ file lets applications access ZMS style data records with minimal impact.

The ZMS style record structure adds a level of naming to each component of the record. The Legacy Style does not have this additional naming level. For example, the field cpu.dispatches in a legacy style record is equivalent to the zmscpu.ctr.dispatches in the ZMS style record structure.

Converting existing applications and ENFORM reports from the legacy naming format to the ZMS style naming format is a significant edit of record references. It also involves removal of fields that might be referenced by programs, and the possible combination of equivalent fields that appeared in two sizes (for example, process.sent-bytes and process.sent-bytes-f).

To simplify this process, use the MEASDDLZ file as described in the *Measure User's Guide*.

MEASDDLZ presents the ZMS style counter widths and locations in the record using the legacy style record naming convention. Record template names use legacy style names (for example, PROCESS rather than ZMSPROC). Fields that have been combined are referenced by redefines.

MEASDDLZ and the ZMS style format are available in the Measure G11 PVU. In H01 and later PVUs, the ZMS style interface is the default. The legacy interface will no longer be enhanced, and MEASDDLZ will no longer be maintained to reflect changes in the ZMS style records. Counters will be added or modified only in ZMS style records.

CLUSTER

The CLUSTER entity type provides information about the number of FOX messages sent and received by all processes on the local system. In this manual, the term FOX refers to:

Component	Server
FOX II	Cyclone or VLX
TorusNet	NonStop K-series
Servernet/FX	NonStop S-series

A FOX link involves at least two processes on different systems: the linker that initiated the link and the listener that accepted the link. Because the Measure subsystem can measure only the local system, the measurement data reflects only one side of each FOX link: that of the linker or the listener, but not both.

Topic	Page
Entity specification syntax	150
DDL record for CLUSTER entities	150
Usage notes for all CLUSTER entities	153
Usage notes for G-series CLUSTER entities	153

The CLUSTER entity is not relevant to H-series and J-series RVUs. A similar entity on H-series and J-series RVUs is SERVERNET.

Entity Specification Syntax for CLUSTER Entities

To describe CLUSTER entities:

```
CLUSTER entity-spec
```

CLUSTER
: collects information about FOX traffic.

entity-spec
: is specified as:

  {*}{\system[(cpu)]}

  *
  : indicates all FOX traffic between the local system and all remote systems.

  \system
  : is the system name or number of a remote system. If you specify *system*, the measurement includes all FOX traffic between the local system and this remote system.

  cpu
  : is the number of a CPU on the local system. If you specify *cpu*, the measurement includes all FOX messages sent or received by this CPU. The default is all CPUs.

DDL Record for CLUSTER Entities (Legacy Style)

This is the DDL record for CLUSTER entities. The fields included in BRIEF reports are in **boldface type**.

The CLUSTER DDL record for G-series entities is identical to the record for D-series entities except:

- Longer byte-count fields are provided to reduce the possibility of field overflow. Each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Note for G-Series CLUSTER Entities (page 153).
- The ERROR field can indicate a field overflow in a 32-bit byte-count field.

```
RECORD cluster. FILE is "cluster" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 remote-system               type binary 16 unsigned.
 02 remote-system-name           type character 8.

* counter value items:
 02 messages-sent               type binary 32 unsigned.
 02 sent-bytes                  type binary 32 unsigned.
 02 returned-bytes              type binary 32 unsigned.
 02 messages-received           type binary 32 unsigned.
 02 received-bytes              type binary 32 unsigned.
 02 reply-bytes                 type binary 32 unsigned.

* F40 counter value items:
 02 sent-bytes-f                type binary 64.
 02 returned-bytes-f            type binary 64.
 02 received-bytes-f            type binary 64.
 02 reply-bytes-f               type binary 64.
end
```

DDL Record for CLUSTER Entities (ZMS Style)

The ZMS Style DDL record for CLUSTER entities is supported on Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsclstr-id.
 02 remote-system-name          type character 8.
 02 remote-system               type binary 16 unsigned.
 02 reserved-1                  type character 6.
end
```

Counter Fields DDL Definitions

```
DEFINITION zmsclstr-ctrs.
 02 messages-sent               type binary 64.
 02 sent-bytes                  type binary 64.
 02 returned-bytes              type binary 64.
 02 messages-received           type binary 64.
 02 received-bytes              type binary 64.
 02 reply-bytes                 type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsclstr. FILE is "zmsclstr" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsclstr-ctrs.
```

```
 02 id                          type zmsclstr-id.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

REMOTE-SYSTEM

System number of the remote system for which FOX traffic was measured.

REMOTE-SYSTEM-NAME

System name of the remote system for which FOX traffic was measured.

MESSAGES-SENT

Number of FOX messages sent by all linker processes running on the local system.

Counter type: Incrementing.

SENT-BYTES

Number of FOX message bytes sent by all linker processes running on the local system.

When a process sends or receives a FOX message, the MESSAGES-SENT or MESSAGES-RECEIVED counter of the PROCESS record for that process is also advanced.

For D-series and G-series RVUs, this is a 32-bit counter.

For G-series RVUs, SENT-BYTES-F field is a 64-bit version of SENT-BYTES.

Counter type: Accumulating.

RETURNED-BYTES

Number of FOX message bytes received by all linker processes running on the local system.

For D-series and G-series RVUs, this is a 32-bit counter.

For G-series RVUs, the RETURNED-BYTES-F field is a 64-bit version of RETURNED-BYTES.

Counter type: Accumulating.

MESSAGES-RECEIVED

Number of FOX message requests received by all listener processes running on the local system.

Counter type: Incrementing.

RECEIVED-BYTES

Number of FOX message bytes received by all listener processes running on the local system.

For D-series and G-series RVUs, this is a 32-bit counter.

For G-series RVUs, the RECEIVED-BYTES-F field is a 64-bit version of RECEIVED-BYTES.

Counter type: Accumulating.

REPLY-BYTES

Number of FOX message bytes sent by all listener processes running on the local system in reply to messages from remote linker processes.

For D-series and G-series RVUs, this is a 32-bit counter.

For G-series RVUs, the REPLY-BYTES-F field is a 64-bit version of REPLY-BYTES.

Counter type: Accumulating.

SENT-BYTES-F

For G-series RVUs, same as SENT-BYTES but accommodates larger values (64 bits rather than 32 bits).

RETURNED-BYTES-F

For G-series RVUs, same as RETURNED-BYTES but accommodates larger values (64 bits rather than 32).

RECEIVED-BYTES-F

For G-series RVUs, same as RECEIVED-BYTES but accommodates larger values (64 bits rather than 32).

REPLY-BYTES-F

For G-series RVUs, same as REPLY-BYTES but accommodates larger values (64 bits rather than 32).

Usage Notes for All CLUSTER Entities

- CLUSTER counters are advanced by the message system during network operations that send data directly to another system over a FOX link.

  CLUSTER does not count messages sent through the network line handler. Those messages are counted in the NETLINE and SYSTEM counter records. For systems connected by a FOX link, only messages requiring security checks go through the network line handler.

  For example, assume you use FUP COPY to copy a file to a remote system connected by a FOX link. Because the file security must be checked, the OPEN message is sent through the network line handler and is counted by the NETLINE and SYSTEM counters. The file data is sent over the FOX line and is counted by the CLUSTER entity.

- When a process sends or receives a FOX message, the MESSAGES-SENT or MESSAGES-RECEIVED counter of the PROCESS record for that process is also advanced.
- For detailed information on FOX transfers and Expand line handlers, see the *Expand Configuration and Management Manual* and the *Expand Network Management and Troubleshooting Manual*.

Usage Note for G-Series CLUSTER Entities

The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field SENT-BYTES-F collects the same data as the 32-bit field SENT-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

The 32-bit fields are still active and continue to return values. If no field overflow occurs, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

Convert your applications to use the 64-bit fields; 32-bit fields might be deactivated in a future PVU.

In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST DEVICE BY SENT-BYTES, not LIST DEVICE BY SENT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

Usage Note for H-Series and J-Series

CLUSTER entities are not supported in H-series and J-series RVUs, because the underlying component is no longer supported on the system. MEASCOM does not return a syntax error if

you configure a CLUSTER entity (so old scripts that configure CLUSTER entities don't have to be modified right away), but no records are configured or returned.

CONTROLLER

This entity measures I/O activity on controllers for disks, tape, terminals, and other devices on systems running D-series RVUs. Starting with the D45 product version of Measure, the default is to have the CONTROLLER entity disabled. For more information on this default setting, see Usage Note for CONTROLLER Entities (page 156).

To enable the CONTROLLER entity, before starting the Measure subsystem:

```
ADD DEFINE =_MEASURE_CONTROLLERS, FILE A
```

To measure ServerNet addressable controllers (SACs) on systems running G-series RVUs, use the SERVERNET entity.

NOTE: You can measure ServerNet addressable controllers with a D-series measurement application if it specifies all controllers (ADD CONTROLLER *) or only a CPU number (ADD CONTROLLER 2). To measure specific SACs, you must modify the entity identifiers.

Entity Specification Syntax for CONTROLLER Entities

To describe a CONTROLLER entity:

```
CONTROLLER entity-spec
```

CONTROLLER

collects information about one or more controllers on the system.

entity-spec

is specified as:

```
{ * [ ( type ) ]                                }
{ cpu [ , channel [ , ctrl ] ] [ ( type ) ] }
```

*

measures all controllers in all CPUs.

cpu

is the number of the CPU in which the controller to be measured is configured.

channel

is the channel number of the controller to be measured. Use an asterisk (*) to indicate all channels. The default is all channels.

ctrl

is the controller number of the controller to be measured (0 through 31). Use an asterisk (*) to indicate all controllers. The default is all controllers.

type

is the HP product number of the controller (such as 3128). The default is all controller types matching the specified cpu, channel, ctrl set.

DDL Record for CONTROLLER Entities

This is the DDL record for CONTROLLER entities. The fields included in BRIEF reports are in **boldface type**.

```
RECORD contrl.  FILE is "contrl" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
* entity identification items:
 02 channel                    type binary 16 unsigned.
 02 ctrl                       type binary 16 unsigned.
 02 ctrl-type                  type binary 16 unsigned.

* counter value items:
 02 requests                   type binary 32 unsigned.
 02 total-io-bytes             type binary 64.
 02 io-qtime                   type binary 64.
 02 io-qlen-max                type binary 16 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

CHANNEL

Number of the channel on which the measurement was taken.

CTRL

Number of the controller on which the measurement was taken.

CTRL-TYPE

The product number, such as 3128.

REQUESTS

Total number of requests to the controller by all I/O processes (IOPs) connected to this controller path.

Counter type: Incrementing.

TOTAL-IO-BYTES

Total number of bytes transferred to and from the controller by all IOPs connected to this controller path.

Counter type: Accumulating.

IO-QTIME

The time that the controller is busy performing I/Os. If the average number of I/Os is less than 1, the number can be interpreted as the controller busy percentage.

Counter type: Queue.

IO-QLEN-MAX

(Legacy Style only) Maximum number of outstanding I/Os on the controller queue described by the IO-QTIME counter.

Counter type: Max queue.

Usage Note for CONTROLLER Entities

Because of differences in implementation, CONTROLLER counters are not directly comparable to device-level counters. For example, a CONTROLLER REQUESTS counter value is not equal to the sum of the REQUESTS counter values for all devices that share that controller.

Measuring controller activity has questionable usefulness and is not generally advised. Architectural differences between the communication and disk subsystems preclude correct data being collected for communication devices. In addition, compatibility problems between controller entity measurements and the online configuration utility program (COUP) can cause serious system stability issues (for example, system halts). Therefore, do not use the CONTROLLER entity.

The requests and data bytes returned from storage controllers are valid and can potentially be useful in assessing system performance. To enable the CONTROLLER entity in a test environment, set the define outlined in CONTROLLER (page 154).

CPU

The CPU entity type provides information about one or more CPUs in the local system:

Entity Specification Syntax for CPU Entities

To describe a CPU entity:

```
CPU entity-spec
```

CPU

collects information about one or more CPUs in the local system.

entity-spec

is specified as:

```
{ *   }
{ cpu } [, cpu ] ...
```

*

indicates all CPUs.

cpu

is the number of a CPU to be measured. To measure multiple CPUs, specify a comma-separated list of CPU numbers.

The default value is all CPUs.

DDL Record for CPU Entities (Legacy Style)

This is the DDL record for CPU entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

The CPU DDL record for G-series entities is identical to the CPU record for D-series entities except:

- Counters are added to support direct bulk I/O transfers.
- The existing DISK-IOS counter also includes direct bulk I/O requests.
- The SEND-BUSY-TIME field is no longer used.
- Fields are added to support changes in memory handling, and the activity measured by some existing fields changed.
- Longer byte-count fields are provided to reduce the possibility of field overflow. Each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Notes for G-Series CPU Entities (page 170).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.
- As of G05, CPU-QTIME does not measure the amount of time processes stay on the ready list for page faults. As such, MEM-QTIME is always zero.
- As of G08, instrumentation for the ServerNet Cluster is added.

```
RECORD cpu. FILE is "cpu" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
 * entity identification items:
 02 cpu-type                 type binary 16 unsigned.

 * redefinition for D25
 02 memory-pages             type binary 16 unsigned.
 02 mem-mb                   type binary 16 unsigned
    redefines memory-pages.

 02 pcbs                     type binary 16 unsigned.
 02 lcbs                     type binary 16 unsigned.

* counter value items:
 02 cpu-busy-time            type binary 64.
 02 cpu-qtime                type binary 64.
 02 cpu-qlen-max             type binary 16 unsigned.
 02 mem-qtime                type binary 64.
 02 mem-qlen-max             type binary 16 unsigned.
 02 dispatches               type binary 32 unsigned.
 02 swaps                    type binary 32 unsigned.
 02 intr-busy-time           type binary 64.
 02 process-ovhd             type binary 64.
 02 send-busy-time           type binary 64.
 02 disc-ios                 type binary 32 unsigned.
 02 cache-hits               type binary 32 unsigned.
 02 transactions             type binary 32 unsigned.
 02 response-time            type binary 64.
 * redefinition for D25:
 02 memory-pages32           type binary 32 unsigned.
 02 mem-frames               type binary 32 unsigned.
    redefines memory-pages32.

 02 cpu-subtype              type binary 16 unsigned.

* fields for TNS/R specific counters:
 02 accel-busy-time          type binary 64.
 02 tns-busy-time            type binary 64.
 02 comp-traps               type binary 32 unsigned.
```

```
* Native Mode busy time
 02 tnsr-busy-time           type binary 64.

* New memory counters for D25:
 02 page-size-bytes          type binary 16 unsigned.
 02 mem-initial-lock         type binary 32 unsigned.
 02 page-requests            type binary 32 unsigned.
 02 page-scans               type binary 32 unsigned.
 02 Starting-free-mem        type binary 32 unsigned.
 02 Ending-free-mem          type binary 32 unsigned.
 02 Starting-UCME            type binary 32 unsigned.
 02 Ending-UCME              type binary 32 unsigned.

 02 Starting-UDS             type binary 32 unsigned.
 02 Ending-UDS               type binary 32 unsigned.
 02 Starting-UDS-lock        type binary 32 unsigned.
 02 Ending-UDS-lock          type binary 32 unsigned.

 02 Starting-SDS             type binary 32 unsigned.
 02 Ending-SDS               type binary 32 unsigned.
 02 Starting-SDS-lock        type binary 32 unsigned.
 02 Ending-SDS-lock          type binary 32 unsigned.

 02 Starting-UCL             type binary 32 unsigned.
 02 Ending-UCL               type binary 32 unsigned.
 02 Starting-UCL-lock        type binary 32 unsigned.
 02 Ending-UCL-lock          type binary 32 unsigned.

 02 Starting-SCL             type binary 32 unsigned.
 02 Ending-SCL               type binary 32 unsigned.
 02 Starting-SCL-lock        type binary 32 unsigned.
 02 Ending-SCL-lock          type binary 32 unsigned.
 02 Ending-Free-CIDs         type binary 32 unsigned
    redefines Ending-SCL-Lock.

* New counters for G04
 02 unsp-pages-qtime           type binary 64.
 02 unsp-pages-qlen-max        type binary 16 unsigned.
 02 unsp-pages-start           type binary 32 unsigned.
 02 unsp-pages-end             type binary 32 unsigned.

* New identifier in G05:
 02 processor-status           type binary 32 unsigned.

* New counters for G05:
 02 disc-ios-f                 type binary 64.
 02 cache-hits-f               type binary 64.
 02 svnet                      occurs 16 times.
   03 read-requests            type binary 32 unsigned.
   03 write-requests           type binary 32 unsigned.
   03 read-bytes               type binary 64.
   03 write-bytes              type binary 64.

 * New counters for G08:
 02 link-prepush-msgs          type binary 64.
 02 link-readlink-msgs         type binary 64.
 02 link-large-msgs            type binary 64.
 02 readlinkcache-all          type binary 64.
 02 readlinkcache-ctrl         type binary 64.
 02 readlinkcache-none         type binary 64.
 02 replyctrlcache-msgs        type binary 64.
 02 ending-free-cids           type binary 32 unsigned.
                              redefines ending-scl-lock
 02 reserved                   type character 40.
```

```
end
```

DDL Record for CPU Entities (ZMS Style)

The ZMS Style DDL record for CPU entities is supported on Measure G11 and later PVUs. The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmscpu-id.
 02 cpu-type                 type binary 16 unsigned.
 02 cpu-subtype              type binary 16 unsigned.
 02 mem-mb                   type binary 16 unsigned.
 02 pcbs                     type binary 16 unsigned.
 02 page-size-bytes          type binary 32 unsigned.
 02 mem-frames               type binary 32 unsigned.
 02 mem-initial-lock         type binary 32 unsigned.
 02 ipus                     type binary 16 unsigned.
 02 reserved-1               type character 2.
end
```

Counter Fields DDL Definition

```
DEFINITION zmscpu-ctrs.
 02 cpu-busy-time            type binary 64.
 02 cpu-qtime                type binary 64.
 02 dispatches               type binary 64.
 02 swaps                    type binary 64.
 02 intr-busy-time           type binary 64.
 02 process-ovhd             type binary 64.
 02 disc-ios                 type binary 64.
 02 cache-hits               type binary 64.
 02 transactions             type binary 64.
 02 response-time            type binary 64.
 02 comp-traps               type binary 64.
 02 native-busy-time         type binary 64.
 02 accel-busy-time          type binary 64.
 02 tns-busy-time            type binary 64.
 02 page-requests            type binary 64.
 02 page-scans               type binary 64.
 02 mm-page-scans            type binary 64.
 02 Starting-free-mem        type binary 32 unsigned.
 02 Ending-free-mem          type binary 32 unsigned.
 02 Starting-UCME            type binary 32 unsigned.
 02 Ending-UCME              type binary 32 unsigned.
 02 Starting-UCL             type binary 32 unsigned.
 02 Ending-UCL               type binary 32 unsigned.
 02 Starting-SCL             type binary 32 unsigned.
 02 Ending-SCL               type binary 32 unsigned.
 02 Starting-Free-CIDs       type binary 32 unsigned.
 02 Ending-Free-CIDs         type binary 32 unsigned.
 02 unsp-pages-qtime         type binary 64.
 02 unsp-pages-start         type binary 32 unsigned.
 02 unsp-pages-end           type binary 32 unsigned.
 02 link-prepush-msgs        type binary 64.
 02 link-readlink-msgs       type binary 64.
 02 link-large-msgs          type binary 64.
 02 readlinkcache-all        type binary 64.
 02 readlinkcache-ctrl       type binary 64.
 02 readlinkcache-none       type binary 64.
 02 replyctrlcache-msgs      type binary 64.
 02 processh-samples         type binary 64.
```

```
* New Counters for H01
  02 Starting-timer-cells     type binary 32 unsigned.
 02 Ending-timer-cells       type binary 32 unsigned.

* New IPU specific counters for H03/NSMA
* (a new array must be inserted before an existing array)
 02 ipu                      occurs 16 times.
   03 ipu-busy-time          type binary 64.
   03 ipu-qtime              type binary 64.
   03 ipu-dispatches         type binary 64.

 02 svnet                    occurs 16 times.
   03 read-requests          type binary 64.
   03 write-requests         type binary 64.
   03 read-bytes             type binary 64.
   03 write-bytes            type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmscpu. FILE is "zmscpu" ENTRY-SEQUENCED.
 02 hdr                        type zmsheader.
 02 ctr                        type zmscpu-ctrs.
 02 id                         type zmscpu-id.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

CPU-TYPE

One of these values:

1	TNS II
2	TXP
3	VLX
4	CLX (except CLX 2000)
5	Cyclone
6	NSR-L
7	NSR-N, NSR-P, or NSR-K
8	NSR-W
9	NSR-G, NSR-T, NSR-V, NSR-X, NSR-Y, NSR-Z, NSR-D, NSR-E, NSR-H, NSR-J, NSR-Y
10	NSE

CPU-SUBTYPE

Binary value indicating the subtype for the CPU.

MEMORY-PAGES

Redefined. See MEM-MB.

MEM-MB

Total memory size in megabytes.

PCBS

Number of process control blocks (PCBs) configured by the SYSGEN program.

LCBS

No longer used; returns 0.

PAGE-SIZE-BYTES

Number of bytes in a memory frame for the CPU being measured. This value is 2048 for CISC processors, 4096 for Cyclone/R, CLX/R, and K1000 processors, and 16384 for newer processors.

MEM-FRAMES

Number of frames (physical pages) in the CPU configuration. (This value is the PHYSCL value in the PEEK utility.)

MEM-INITIAL-LOCK

Number of page frames locked by the SYSGEN program. The difference between MEM-FRAMES and MEM-INITIAL-LOCK is called SWAPBL in PEEK. The value is the number of memory page frames locked during system load and available for use by the operating system. The value depends on the operating system.

IPUS

In Measure J01 and later PVUs, supports the reporting of the number of IPUs comprising the logical CPU. IPU-specific counters are:

IPU-BUSY-TIME	IPU-QTIME	IPU-DISPATCHES

IPU-BUSY-TIME
: The time during which this IPU was busy.

IPU-QTIME
: The time (in microseconds) that processes spent on the ready list. Same as CPU-QTIME but for an IPU.

IPU-DISPATCHES
: The number of dispatches for that IPU.

CPU-BUSY-TIME

The time that the CPU was busy.

For TNS/R systems, CPU-BUSY-TIME is the sum of ACCEL-BUSY-TIME, TNS-BUSY-TIME, TNSR-BUSY-TIME, and INTR-BUSY-TIME.

For TNS/E systems, CPU-BUSY-TIME is the sum of ACCEL-BUSY-TIME, TNS-BUSY-TIME, and NATIVE-BUSY-TIME, and includes the INTR-BUSY-TIME

For NSMA systems in Measure J01 and later PVUs, CPU-BUSY-TIME is the aggregate of the BUSY-TIMEs of the individual IPUs that comprise the logical CPU. With REPORT RATE ON, the maximum value of this field is always 100%. With REPORT RATE OFF, this is the raw aggregate value of the BUSY-TIMEs of all its IPUs.

Counter type: Busy.

CPU-QTIME

The time (in microseconds) that processes spent on the ready list. Prior to the G05 Measure PVU, the ready list includes processes waiting on page faults. The CPU MEM-QTIME counter shows the time spent waiting for arrival of message system data.

For G05 and later PVUs, a process no longer remains active in the ready list timer, so the value of MEM-QTIME is zero.

Counter type: Queue.

CPU-QLEN-MAX

(Legacy Style only) Maximum number of items on the queue described by the CPU-QTIME counter. After a system is loaded, all system processes are placed on the ready list. Because this counter is initialized only at system load, it generally reflects a one-time, very long CPU queue.

Counter type: Max queue.

MEM-QTIME

Amount of CPU time that processes in the measured processor spent waiting on page faults. (A page fault does not cause the system to remove the process from the ready list.)

For G05 and later product versions, the activity measured by this counter has changed somewhat because of changes in memory handling. In earlier G-series RVUs (and all D-series RVUs), page-fault requests are handled by the memory manager ($VIRTUAL). Concurrent requests must queue for processing.

As of G05, the handling of page-faulting is a system library function. No queueing occurs because multiple requests are processed concurrently. As a result, MEM-QTIME is always zero.

Related Counter	Function	Page
CPU-QTIME	Measures the time all processes spent on the ready list, including time spent waiting on page faults	161
PROCESS MEM-QTIME	Measures the time a process spent waiting on page faults	162
PROCESS PAGE-FAULTS	Counts the number of page faults generated by a process	280

Counter type: Queue.

MEM-QLEN-MAX

(Legacy style only) Maximum number of items on the queue described by the MEM-QTIME counter.

For G05 and later product versions, zero. For earlier product versions, the maximum number of queued page faults since the processor was loaded.

Counter type: Max queue.

MM-PAGE-SCANS

(ZMS Style only) Reserved for future use.

DISPATCHES

Number of times a process was selected from the ready list and executed by the CPU. The DISPATCHES counter for the PROCESS entity shows the number of times a particular process was dispatched (selected and run).

Counter type: Incrementing.

SWAPS

Number of swap operations (both into and out of memory) performed by the memory manager. The memory manager swaps one or two pages at a time. The CPU SWAPS counter is advanced once for each swap regardless of the number of pages swapped. (The CPU SWAPS counter differs from the DISC SWAPS counter.)

Only pages that are updated are swapped. Code pages are swapped into memory, but because they cannot be updated, they are never swapped out. Data pages are swapped both into and out of memory. However, a data page that has no initial data is not swapped in until it is used.

Depending on memory requirements and the types of pages being swapped in and out, a page fault (as counted by the PROCESS entity) can cause no swaps, one swap, or two swaps:

Situation	Swaps
Code removed or free page available in memory; no initial data page swapped in	0
Code removed or free page available; code or data page with initial data swapped in	1
Used data page swapped out and used data page swapped in	2

Counter type: Incrementing.

INTR-BUSY-TIME

The time that the CPU spent executing interrupt handlers. Most interrupts are caused by the processing of messages and I/O operations. The appropriate system description manual describes each type of interrupt.

For TNS/E systems, this is the total time spend executing system Interrupt Processes (IPs) and system Auxiliary Processes (APs).

Counter type: Busy.

PROCESS-OVHD

The time spent initializing and terminating processes.

Counter type: Busy.

SEND-BUSY-TIME

For D-series RVUs, the time that the CPU spent sending data to other CPUs; that is, the time spent executing SEND instructions. This counter partially overlaps the INTR-BUSY-TIME counter.

For G-series RVUs, no longer used.

Counter type: Busy.

DISC-IOS

Number of I/O disk transfers performed by disk processes in the measured CPU. This count includes direct bulk I/O requests initiated by this CPU for other CPUs.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the DISC-IOS-F field is a 64-bit version of DISC-IOS.

Counter type: Incrementing.

CACHE-HITS

Number of times the required block was found in cache during an I/O operation.

For a read operation (or the read part of a write operation), a cache hit saves a disk I/O. For a write operation, the result of a cache hit depends on the type of file I/O. For buffered I/O, the cache hit saves a disk I/O. For write-through I/O, it does not.

A write operation causes a read in a number of cases. For example, in a key-sequenced file, writing to a data block can require reads of one or more index blocks. Or, if you write a partially filled block to disk, the disk process must read the block from disk, merge the new data into the block, and then write the block to disk.

The DISC entity provides additional cache and I/O information.

For D-series and G-series RVUs, this is a 32-bit counter. For H- J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the CACHE-HITS-F field is a 64-bit version of CACHE-HITS.

Counter type: Incrementing.

TRANSACTIONS

Number of transactions performed by terminal processes on the measured CPU that are in currently active measurements. A transaction is a read from a terminal followed by a write to the terminal. Response time is the interval between the end of the read and the beginning of the write.

Related Counter	Function	Page
RESPONSE-TIME	Measures the response time	164
TERMINAL RESPONSE-TIME and TRANSACTIONS	Measures response time and transactions for a particular terminal	337

Counter type: Incrementing.

RESPONSE-TIME

The time that terminal processes in the CPU and in active measurements spent on terminal responses. A transaction is a read from a terminal followed by a write to the terminal. Response time is the interval between the end of the read and the beginning of the write.

Related Counter	Function	Page
TRANSACTIONS	Measures the number of transactions	337
TERMINAL RESPONSE-TIME and TRANSACTIONS	Measures response time and transactions for a particular terminal	337

Counter type: Response time.

MEMORY-PAGES32

Redefined. See MEM-FRAMES.

ACCEL-BUSY-TIME

The time that the CPU was busy executing accelerated code. This counter applies only to measurements taken on a TNS/R system or a TNS/E system.

For H-series, J-series and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy.

NATIVE-BUSY-TIME

(ZMS Style only) Amount of time the processor was busy executing native code.

For H-series, J-series, and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy

TNS-BUSY-TIME

The time that the CPU was busy executing TNS code. This counter applies only to measurements taken on a TNS/R or TNS/E system.

For H-series, J-series, and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy.

PROCESSH-SAMPLES

(ZMS Style only) This counter is valid only for H-series, J-series, and later RVUs. It reports the raw count of the number of PROCESSH samples in the processor, if PROCESSH measurement is active. If no PROCESSH measurement is active, this counter is not displayed, and its value in the record is zero.

For NSMA systems in Measure J01 and later PVUs, PROCESSH-SAMPLES reports the aggregate value of all the number of samples on all the IPUs and hence is typically equal to n*sampling frequency on that CPU.

COMP-TRAPS

Number of times a compatibility trap occurred on the processor. A compatibility trap is a data misalignment event, an unexpected transition to or from accelerated or unaccelerated code, or a relative segment 2 or 3 problem. See the *EPTRACE Manual* for further details.

Counter type: Incrementing.

TNSR-BUSY-TIME

(Legacy Style only) The time that the CPU was busy executing TNS/R native code. This counter applies only to measurements taken on a TNS/R system and is derived from other counters:

```
CPU-BUSY-TIME - (INTR-BUSY-TIME + TNS-BUSY-TIME + ACCEL-BUSY-TIME) =
TNSR-BUSY-TIME
```

Counter type: Busy.

PAGE-REQUESTS

Number of times the memory manager was invoked to obtain a page frame. (In the PEEK utility, this value is called CLOCK:CALLS.)

Counter type: Incrementing.

PAGE-SCANS

Number of page frames the memory manager examined to obtain a replaceable page. (The PEEK utility provides a SCANS/CALLS value that shows the average number of frames examined to find a replaceable page.)

Counter type: Incrementing.

STARTING-FREE-MEM

Number of free memory frames at the start of the measurement interval.

Counter type: Snapshot.

ENDING-FREE-MEM

Number of free memory frames at the end of the measurement interval.

Counter type: Snapshot.

STARTING-UCME

Number of UCME (uncorrectable memory error) frames at the start of the measurement interval.

Counter type: Snapshot.

ENDING-UCME

Number of UCME (uncorrectable memory error) frames at the end of the measurement interval.

Counter type: Snapshot.

STARTING-UDS

(Legacy Style only) Number of user process data segment frames allocated in physical memory at the start of the measurement interval. This counter includes the number of frames allocated for user process stack segments, extended data segments, and process file segments (PFS). With SPRs to T9050G05 and later PVUs, STARTING-UDS frames are included in the STARTING-UCL counter.

Counter type: Snapshot.

ENDING-UDS

(Legacy Style only) Number of user process data segment frames allocated in physical memory at the end of the measurement interval. With SPRs to T9050G05 and later PVUs, ENDING-UDS frames are included in the ENDING-UCL counter.

Counter type: Snapshot.

STARTING-UDS-LOCK

No longer used.

ENDING-UDS-LOCK

No longer used.

STARTING-SDS

(Legacy Style only) Number of system process data segment frames allocated in physical memory at the start of the measurement interval. This counter includes the number of frames allocated

for system process stack segments, extended data segments, and process file segments (PFS) as well as the CPU system global data segment (SG) and any other extended data segments or frames allocated for operating system use. A system process is any process that falls under the Measure PROCESS entity category of system processes. With SPRs to Measure G05 and later PVUs, STARTING-SDS frames are included in the STARTING-SCL counter.

Counter type: Snapshot.

ENDING-SDS

(Legacy Style only) Number of system process data segment frames allocated in physical memory at the end of the measurement interval. With SPRs to Measure G05 and later PVUs, ENDING-SDS frames are included in the ENDING-SCL counter.

Counter type: Snapshot.

STARTING-FREE-CIDS

(ZMS style only) Number of counter ID control blocks available for new counter records in this processor at the beginning of the reported measurement interval.

ENDING-FREE-CIDS

Number of counter ID control blocks available for new counter records in this processor at the end of the reported measurement interval.

STARTING-SDS-LOCK

No longer used.

ENDING-SDS-LOCK

No longer used.

STARTING-UCL

Number of user process code and library segment frames allocated in physical memory at the start of the measurement interval. This counter includes the number of frames allocated for user process TNS, accelerated, and native code segments for the short address spaces designated as user code (UC) and user library (UL).

Counter type: Snapshot.

ENDING-UCL

Number of user process code segment frames and library segment frames allocated in physical memory at the end of the measurement interval.

Counter type: Snapshot.

STARTING-UCL-LOCK

No longer used.

ENDING-UCL-LOCK

No longer used.

STARTING-SCL

Number of system process code and library segment frames allocated in physical memory at the start of the measurement interval. This counter includes the number of frames allocated for TNS, accelerated, and native code for system process code segments and the short address spaces designated as system code (SC) and system library (SL). It also includes frames allocated for interrupt procedures and millicode (for TNS/R systems).

Counter type: Snapshot.

ENDING-SCL

Number of system process code and library segment frames allocated in physical memory at the end of the measurement interval.

Counter type: Snapshot

STARTING-SCL-LOCK

No longer used.

ENDING-SCL-LOCK

No longer used.

UNSP-PAGES-QTIME

The time that unsponsored pages (shared pages not owned by any process) spent in main memory. For a description of unsponsored pages, see Usage Notes for G-Series CPU Entities (page 170).

Counter type: Queue.

UNSP-PAGES-QLEN-MAX

(Legacy Style only) Maximum number of items on the queue described by the UNSP-PAGES-QTIME counter.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max Queue.

UNSP-PAGES-START

The number of pages on the UNSP-PAGES-QTIME queue at the start of the measurement interval.

Counter type: Snapshot.

UNSP-PAGES-END

The number of pages on the UNSP-PAGES-QTIME queue at the end of the measurement interval.

Counter type: Snapshot.

PROCESSOR-STATUS

(Legacy Style only) Number of processors on the system (16 for all NonStop S-series systems) and status of each processor at the start of the measurement (1 for up, 0 for down.) Obtained from the Guardian PROCESSORSTATUS procedure call. Not used for systems running H-series and J-series RVUs.

DISC-IOS-F

In G-series RVUs, same as DISC-IOS but accommodates larger values (64 bits rather than 32 bits).

CACHE-HITS-F

In G-series RVUs, same as CACHE-HITS but accommodates larger values (64 bits rather than 32 bits).

SVNET

ServerNet activity for the measured processor. This field is divided into these subfields:

READ-REQUESTS	WRITE-REQUESTS	READ-BYTES	WRITE-BYTES

In Measure reports, ServerNet activity values are provided separately for each processor to or from which the measured processor transferred data.

If you generate a report for all processors and use the TOTALS option, the report shows the total ServerNet data transfer activity for each processor. The fields for each specific processor can be interpreted as the total ServerNet data demand by that processor. I/O requests are counted once in these totals, but message system interprocessor communication (IPC) requests are counted twice, once in each processor involved in the transfer.

READ-REQUESTS

Number of ServerNet transfers initiated in the measured processor that brought information into the indicated processor (CPU *n*). For transfers within the measured processor, this count is the number of standard I/O read requests. For transfers with other processors, the count includes the number of DBIO read requests.

Counter type: Incrementing.

WRITE-REQUESTS

Number of ServerNet transfers initiated in the measured processor that sent information out of the indicated processor (CPU *n*). For transfers within the measured processor, this count is the number of standard I/O write requests and the number of message system send requests initiated from the measured processor. For transfers to other processors, the count includes the number of DBIO write requests.

Counter type: Incrementing.

READ-BYTES

Number of bytes transferred by read requests for the indicated processor (CPU *n*).

Counter type: Accumulating.

WRITE-BYTES

Number of bytes transferred by write requests for the indicated processor (CPU *n*).

Counter type: Accumulating.

LINK-PREPUSH-MSGS

Number of messages transferred to another CPU using the prepush protocol.

Counter type: Incrementing.

LINK-READLINK-MSGS

Number of messages transferred to another CPU not using the prepush protocol but within the readlinkcache message size.

Counter type: Incrementing.

LINK-LARGE-MSGS

Number of messages transferred to a CPU that exceeded the readlinkcache message size.

Counter type: Incrementing.

READLINKCACHE-ALL

Number of received messages for which both control and data bytes were cached in readlinkcache buffers.

Counter type: Incrementing.

READLINKCACHE-CTRL

Number of received messages for which only control bytes were cached in readlinkcache buffers.

Counter type: Incrementing.

READLINKCACHE-NONE

Number of received messages for which neither control nor data bytes were cached in readlinkcache buffers.

Counter type: Incrementing.

REPLYCTRLCACHE-MSGS

Number of message replies for which control was carried in the interrupt packet.

Counter type: Incrementing.

RESERVED

Reserved for future use.

Usage Notes for All CPU Entities

- Unlike counters for other entity types, CPU counters are allocated and initialized only when the CPU is loaded. The counters are not reinitialized at the start of each measurement. Thus, the CPU counter provides an overview of system performance over an extended period.

  CPU counters can direct you to potential problems that you can examine in greater detail using other entities. In particular, the PROCESS CPU-BUSY-TIME counter measures the time spent executing specific processes. (The PROCESS CPU-NUM field lets you match processes to CPUs.)
- For more information on system operation (how the system performs I/O, executes processes, does paging, and so on), see the appropriate system description manual.

Usage Note for D-Series CPU Entities

In Measure D30 and later PVUs, these counters are no longer supported:

STARTING-UDS-LOCK	ENDING-UDS-LOCK	STARTING-SDS-LOCK	ENDING-SDS-LOCK
STARTING-UCL-LOCK	ENDING-UCL-LOCK	STARTING-SCL-LOCK	ENDING-SCL-LOCK

Usage Notes for G-Series CPU Entities

- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field DISC-IOS-F collects the same data as the 32-bit field DISC-IOS. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. If there is no field overflow, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

  Convert your applications to use the 64-bit fields; 32-bit fields might be deactivated in a future release.

In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST CPU BY DISC-IOS, not LIST CPU BY DISC-IOS-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

- In Measure G05 and later PVUs, sponsored and unsponsored memory pages are available. Pages are sponsored in memory by the process that causes them to be present (usually, but not always, the process that defines the segment). If a process that is the sponsor of shared pages terminates, the pages become temporarily unsponsored. The next sharing process to reference an unsponsored page becomes the sponsor for that page. These CPU counters provide information about unsponsored pages:

UNSP-PAGES-QTIME	UNSP-PAGES-START
UNSP-PAGES-QLEN-MAX	UNSP-PAGES-END

- For a discussion of the different types of message system transfer protocols and concepts, (pre-push, post-pull, linker and listener, and so on), see the *NonStop S-Series Server Description Manual*.
- In Measure G08 and later PVUs, Measure has several new counters for ServerNet Cluster support.
- The number of free CIDs in a processor is determined by the maximum number of CIDs allowed and the current usage by all active measurements. Twenty-four CIDs are used for internal list headers. You can configure the number of CIDs through DEFINEs. This configuration parameter is read when the START MEASSUBSYS command is issued.

Usage Notes for H-Series and J-Series CPU Entities

In H-series, J-series, and later RVUs, all byte count fields are 64-bit counters. The suffix -F no longer appears in counter names in ZMS format; however, counters with names ending in -F remain available to applications requesting data in legacy style.

DEVICE

The DEVICE entity type provides information about devices such as tape drives or printers. In G-series and later RVUs, the DEVICE entity applies only to tape drives.

NOTE: Some devices are identified by specific entity types: DISC for disks, LINE and NETLINE for communication lines, and TERMINAL for subdevices and asynchronous terminals.

Entity Specification Syntax for DEVICE Entities

To describe DEVICE entities:

You can use a D-series measurement application to measure devices on NonStop S-series servers if the application specifies all devices (ADD DEVICE *) or specifies only *$device* or *$device (cpu)*. If the application specifies *controller*, *channel*, or *unit*, you must modify the entity identifiers to measure NonStop S-series devices.

`DEVICE entity-spec`

DEVICE

collects information about one or more devices on the local system.

entity-spec

is specified as:

```
{ *                                                          }
{ $device [ ( cpu [ , chan [ , ctrl [ , unit ] ] ] ) ] }
{ $device [ ( cpu [, svnet [, group [, module [, slot
[, scsi-id ] ] ] ] ] ) ]                                     }
{ $device [ ( cpu [, svnet [, sac [, lun ] ] ] ) ]           }
```

*

indicates all devices in all CPUs.

$*device*

is the name of the device to measure. To indicate all devices, use an asterisk (*).

cpu

is the number of the CPU in which the device to be measured is configured. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

chan

(D-series) is the channel number of the device to measure. The default is all channels.

ctrl

(D-series) is the controller number of the device to be measured (0 through 31). The default is all controllers.

unit

(D-series) is the unit number of the device to be measured. The default is all units.

svnet

(G-series and later RVUs) is X to indicate the X fabric, Y to indicate the Y fabric, or an asterisk (*) to indicate both fabrics. The Measure product returns all device records associated with the fabric you specify. The default is to return records associated with both fabrics.

Devices connected by a ServerNet device adapter (ServerNet D/A) do not have an X or Y fabric. To indicate these devices, use an asterisk (*). In this case, you can use an asterisk (*) in the LISTACTIVE command if the rest of the entity specification describes one device.

Devices connected by an FCSA or CLIM are dual fabric adapters, that is, they are connected to both the X and Y fabrics. For these devices, only "*" or "X" should be specified in the LIST command, and only "X" should be specified in the LISTACTIVE command.

group

(G-series and later RVUs) is the group number of the device to measure. (The group corresponds to the physical enclosure.) To indicate all groups, use an asterisk (*). The default is all groups.

module

(G-series and later RVUs) is the module number of the device to be measured. To indicate all modules, use an asterisk (*). The default is all modules.

slot

(G-series and later RVUs) is the slot number of the device to be measured. To indicate all slots, use an asterisk (*). The default is all slots.

scsi-id

(G-series and later RVUs) is the SCSI port identifier of the device to be measured. To indicate all SCSI ports, use an asterisk (*). The default is all SCSI ports. This is used for ServerNet/DA and FCSA devices, but not for CLIM devices.

sac

is the name of the SAC the device is connected to. If the device is a CLIM device, *sac* is the name of the CLIM.

lun

is the logical unit number within the CLIM. The HP NonStop operating system supports a range of 0 to 65535 for *lun*. Measure supports a range of 0 to 65534, after reserving 65535 (-1) for the wildcard (*). *lun* can only be specified for CLIM devices.

DDL Record for DEVICE Entities (Legacy Style)

This is the Legacy Style DDL record for DEVICE entities. The fields included in BRIEF reports are in **boldface type** (`requests` appears only in G-series brief reports). This record will not change after the Measure G10 PVU.

The DDL record for G-series and later DEVICE entities is identical to the record for D-series DEVICE entities except:

- The CHANNEL field is redefined to SERVERNET.
- Different entity identification fields are used. The CTRL and UNIT fields are no longer used.
- A SCSI-ID identifier is added to identify devices connected by a ServerNet/DA or FCSA.
- Counters are added to measure busy time in the queue-based ServerNet environment. The READ-BUSY-TIME and WRITE-BUSY-TIME counters are no longer used.
- Counters are added to measure direct-bulk I/O operations.
- Existing I/O counters such as READS and WRITES also include direct-bulk I/O operations.
- Longer byte-count fields are provided to reduce the possibility of field overflow. In G-series RVUs, each 32-bit byte-count field has a 64-bit counterpart. In H-series and J-series RVUs, all byte-count fields have 64 bits. For information on using the 64-bit fields, see Usage Note for G-Series DEVICE Entities (page 179).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.

```
RECORD device. FILE is "device" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                        type binary 16 unsigned.
 02 device-name                type character 8.
 02 logical-device             type binary 16 unsigned.
 02 ctrl                       type binary 16 unsigned.
 02 unit                       type binary 16 unsigned.
 02 device-type                type binary 16 unsigned.
 02 device-subtype             type binary 16 unsigned.

* counter value items:
 02 requests                   type binary 32 unsigned.
 02 read-busy-time             type binary 64.
 02 write-busy-time            type binary 64.
 02 reads                      type binary 32 unsigned.
 02 writes                     type binary 32 unsigned.
 02 input-bytes                type binary 32 unsigned.
```

```
 02 output-bytes              type binary 32 unsigned.

* new entity identification item for D10:
 02 channel                   type binary 16 unsigned.
 02 servernet                 type binary 16 unsigned
                              redefines channel.

* F40 new entity identification items:
 02 config-name               type character 64.
 02 adapter-name              type character 64.
 02 SAC-name                  type character 64.
 02 GMS.
    03 group                  type binary 32 unsigned.
    03 module                 type binary 32 unsigned.
    03 slot                   type binary 32 unsigned.

* F40 new counter value items:
 02 read-qbusy-time           type binary 64.
 02 read-qtime                type binary 64.
 02 read-qlen-max             type binary 16 unsigned.
 02 write-qbusy-time          type binary 64.
 02 write-qtime               type binary 64.
 02 write-qlen-max            type binary 16 unsigned.
 02 device-qbusy-time         type binary 64.
 02 input-bytes-f             type binary 64.
 02 output-bytes-f            type binary 64.

* New identifier in G05:
 02 scsi-id                   type binary 64.
* New counters for G05:
 02 dbio-reads                type binary 32 unsigned.
 02 dbio-writes               type binary 32 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for DEVICE Entities (ZMS Style)

The ZMS Style DDL record for DEVICE entities is supported on Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsdev-id.
 02 pin                       type binary 16 unsigned.
 02 device-type               type binary 16 unsigned.
 02 device-subtype            type binary 16 unsigned.
 02 servernet                 type binary 16 unsigned.
 02 device-name               type character 8.
 02 logical-device            type binary 32 unsigned.
 02 GMS.
    03 group                  type binary 32 unsigned.
    03 module                 type binary 32 unsigned.
    03 slot                   type binary 32 unsigned.
 02 SCSI-id                   type binary 64.
 02 plpt redefines SCSI-id.
    03 plpt-flags             type binary 8 unsigned.
    03 path                   type binary 8 unsigned.
    03 lun                    type binary 16 unsigned.
    03 reserved-2             type binary 16 unsigned.
    03 target-id              type binary 16 insigned.
 02 config-name               type character 64.
 02 adapter-name              type character 64.
```

```
 02 SAC-name                      type character 64.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsdev-ctrs.
 02 requests                      type binary 64.
 02 reads                         type binary 64.
 02 writes                        type binary 64.
 02 input-bytes                   type binary 64.
 02 output-bytes                  type binary 64.
 02 read-qbusy-time               type binary 64.
 02 read-qtime                    type binary 64.
 02 write-qbusy-time              type binary 64.
 02 write-qtime                   type binary 64.
 02 device-qbusy-time             type binary 64.
 02 DBIO-reads                    type binary 64.
 02 DBIO-writes                   type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsdev. FILE is "zmsdev" ENTRY-SEQUENCED.
 02 hdr                           type zmsheader.
 02 id                            type zmsdev-id.
 02 ctr                           type zmsdev-ctrs.
end
```

PIN

Process identification number of the I/O process.

DEVICE-NAME

Name of the measured device.

LOGICAL-DEVICE

Logical device number of the measured device.

CTRL

(Legacy Style only) For D-series, controller number of the measured device. For G-series RVUs, no longer used; returns zero.

UNIT

(Legacy Style only) For D-series, unit number of the measured device. For G-series and later RVUs, no longer used; returns zero.

DEVICE-TYPE

Type of device that was measured (tape, printer, and so on). For a complete list of device types, see the *Guardian Operations Reference Summary*. For G-series and later RVUs, the DEVICE entity measures only tape (DEVICE-TYPE 4) and open SCSI (DEVICE-TYPE 8).

DEVICE-SUBTYPE

An additional identifier for DEVICE-TYPE. For a list of device subtypes, see the *Guardian Operations Reference Summary*.

REQUESTS

Number of requests received by the I/O process. To determine how long requests were queued before being read by the I/O process, measure the I/O process and examine the PROCESS RECV-QTIME counter.

Counter type: Incrementing.

READ-BUSY-TIME

(Legacy Style only) For D-series RVUs, the time spent reading from the device. For G-series and later RVUs, no longer used; returns zero.

Counter type: Busy.

WRITE-BUSY-TIME

(Legacy Style only) For D-series RVUs, the time spent writing to the device. For G-series RVUs, no longer used; returns zero.

Counter type: Busy.

READS

Number of read operations (from device to memory) performed by the I/O process. In addition to programmatic read operations, internal operations (such as retries on I/O operations) also modify this counter. A read operation to a key-sequenced file can result in multiple reads because the index blocks must be read before the data blocks.

Counter type: Incrementing.

WRITES

Number of write operations (from memory to device) performed by the device process. In addition to programmatic write operations, internal operations (such as writing volume labels) also modify this counter. A write operation to a key-sequenced file can result in multiple writes because of block splits or updates to the index blocks.

Counter type: Incrementing.

INPUT-BYTES

Number of bytes read from the device. Because the I/O process modifies this counter before the I/O operation, if the write fails, this counter might not be accurate. In addition to programmatic read operations, internal operations (such as retries on I/O operations) also modify this counter.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the INPUT-BYTES-F field is a 64-bit version of INPUT-BYTES.

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the device. Because the I/O process modifies this counter before the I/O operation, if the write fails, this counter might not be accurate. In addition to programmatic write operations, internal operations (such as writing volume labels) also modify this counter.

For D-series and G-series RVUs, this is a 32-bit counter. For H- and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the OUTPUT-BYTES-F field is a 64-bit version of OUTPUT-BYTES.

Counter type: Accumulating.

CHANNEL

(Legacy Style only) For D-series RVUs, channel number of the device. For G-series RVUs, redefined. See SERVERNET.

SERVERNET

ServerNet fabric used by this specific path to the device. The value is 0 for X fabric or for paths that go through dual fabric adapters.

Devices connected by a ServerNet/DA do not have an X or Y fabric. For these devices, the ServerNet field (SvNet in reports) is displayed as an asterisk (*).

Devices connected by an FCSA or CLIM based adapter are dual fabric adapters, that is, they are connected to both X and Y fabrics. For these devices, the ServerNet field is displayed as "X".

CONFIG-NAME

Logical name of the physical device accessed through this path. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

ADAPTER-NAME

Logical name associated with the adapter on which the SAC resides. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

SAC-NAME

Logical name of the ServerNet addressable controller (SAC) supporting this device for this path. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

GMS

Physical location address (group, module, slot). The GMS field is divided into three subfields:

GROUP	is the group number.
MODULE	is the module number.
SLOT	is the slot number.

You can use the keywords GROUP, MODULE, and SLOT in the IF and BY clauses of the LIST and LISTALL commands.

READ-QBUSY-TIME

The time spent in a state in which read requests of any number were queued to this device in this processor.

Counter type: Queue busy.

READ-QTIME

Total time spent by read requests queued to this device.

Counter type: Queue.

READ-QLEN-MAX

(Legacy Style only) Maximum number of read requests queued to this device at any time since the counter record was allocated.

Counter type: Max queue.

WRITE-QBUSY-TIME

The time spent in a state in which write requests of any number were queued to this device in this processor.

Counter type: Queue busy.

WRITE-QTIME

Total time spent by write requests queued to this device.

Counter type: Queue.

WRITE-QLEN-MAX

(Legacy Style only) Maximum number of write requests queued to this device at any time since the counter record was allocated.

Counter type: Max queue.

DEVICE-QBUSY-TIME

The time spent in a state in which requests of any number or type were queued to this device in this processor.

Because of concurrent activity in the write and read queues, WRITE-QBUSY-TIME plus READ-QBUSY-TIME does not equal DEVICE-QBUSY-TIME.

Counter type: Queue busy.

INPUT-BYTES-F

(G-series Legacy Style only) Same as INPUT-BYTES but accommodates larger values (64 bits rather than 32).

OUTPUT-BYTES-F

(G-series Legacy Style only) Same as OUTPUT-BYTES but accommodates larger values (64 bits rather than 32).

SCSI-ID

SCSI port identifier for this device.

PLPT structure

Starting with the Measure H03 PVU, this structure replaces the SCSI-ID field. More attributes were necessary to identify a CLIM device, so the SCSI-ID field (which had a range of 0 to 999) was reduced to 16 bits and became the PLPT.TARGET-ID field. This structure is used in descriptor and data records. The PLPT subfields are PLPT-FLAGS, LUN, PATH, and TARGET-ID.

PLPT-FLAGS

The PLPT-FLAGS are: MEAS_CLIM_REL, MEAS_PATH_SEL and MEAS_CLIM_DEVICE.

MEAS_CLIM_REL	This identifies this field as a PLPT structure rather than a SCSI-ID field. Since the *scsi-id* could have a -1F (wildcard) in it (if it is a descriptor), this flag only has meaning if the PLPT-FLAGS field is not all 1s. To maintain downward compatibility, this bit is not set for descriptors returned via MEASLISTCONFIG unless the device configuration was specifically for a CLIM device.
MEAS_PATH_SEL	This indicates that selection is only by *path* and device name, and not other attributes like *lun* or *scsi-id*. This flag only has meaning if the PLPT-FLAGS field is not all 1s and is only used in descriptors. (Not used with the DEVICE entity.)
MEAS_CLIM_DEVICE	This indicates that this descriptor or data record is for a CLIM device.

LUN

Logical unit number. This is an unique number assigned to a disk drive within a storage CLIM. This is unique only within a particular CLIM. For pre-CLIM devices, this is 0. For CLIM devices, the valid range is 0 to 65534. This field is treated as an unsigned number. If *lun* is in a descriptor, the wildcard is all 1s

PATH

(Not used with the DEVICE entity.) Connection between the NonStop operating system and the disk drive. On the NonStop operating system, there can be two paths (backup and primary) to a disk drive. For CLIM devices, due to the fault tolerance requirement, this requires two storage CLIMs, with one path through each CLIM. A mirrored NonStop operating system volume actually consists of two disk drives, so such a volume can have four paths associated with it. The valid ranges are 0 to 3 for Primary, Backup, Mirror and Mirror-backup, respectively. If *path* is in a descriptor, the wildcard is all 1s.

TARGET-ID

This is the SCSI device ID returned by the SCSI Interface Manager. Prior to the Measure H03 PVU, this was stored in the 64-bit SCSI-ID field. This value only requires 16 bits, so starting with the Measure H03 PVU this value is stored in this smaller field in order to use the free upper 48 bits for *lun* and *path*. The valid values for *target-id* are 0 to 999. If *target-id* is in a descriptor, the wildcard is all 1s.

DBIO-READS

The number of read operations to the measured device that were enabled for direct bulk I/O to a requesting application processor. This counter is a subset of READS.

Counter type: Incrementing.

DBIO-WRITES

The number of write operations to the measured device that were enabled for direct bulk I/O from a requesting application processor. This counter is a subset of WRITES.

Counter type: Incrementing.

Usage Notes for All DEVICE Entities

- The DEVICE counters track all device activity. Device activity includes both I/O from user and system processes. It also includes internal activities such as writing to the Free Space Table and writing file and volume labels. To compare and balance the workload of two or more devices, use the DEVICE counters.
- SUBSYSTEM-VERSION for ZMSDEV records is provided by the device I/O process; for example, the tape process (OTPPROCP).

Usage Notes for Measure H03 and J01 PVUs

For storage CLIM devices, the DEVICE entity specification syntax is updated to include *lun* and *path* and not use *scsi-id*. As with FCSA devices, this entity specification can be used in the ADD *entity-type*, DELETE *entity-type*, LIST *entity-type*, LISTACTIVE *entity-type*, and LISTALL *entity-type* commands.

Usage Note for G-Series DEVICE Entities

The DEVICE entity in G-series Measure is used only for tapes. TIL and THL are obsolete, and all printer support is through LAN-attached devices. Tape is supported as a SCSI device and shares module driver code with disk operations.

The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field INPUT-BYTES-F collects the same data as the 32-bit field INPUT-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

The 32-bit fields are currently active and continue to return values. If there is no field overflow, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

Convert your applications to use the 64-bit fields; 32-bit fields might be deactivated in a future release.

In MEASCOM commands and command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST DEVICE BY INPUT-BYTES, not LIST DEVICE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

Usage Note for H-Series and J-Series Device Entities

In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.

SUBSYSTEM-VERSION for Device records is provided by the device I/O process, for example, the tape process OTPPROCP.

Examples

Command	Description
ADD DEVICE $TAPE	Measure device named $TAPE
ADD DEVICE *	Measure all devices
ADD DEVICE $* (2)	Measure all devices in CPU 2
ADD DEVICE $* (*, *, 110, 3, 1, 2)	Measure all devices attached to a controller whose GMS is 110,3,1 and whose *scsi-id* is 2 (ServerNet/DA or FCSA storage devices only)
ADD DEVICE $* (*,*,CLIM3)	Measure all devices attached to the SAC named CLIM3

DISC

The DISC entity type provides information about one or more disks on the local system:

Entity Specification Syntax for DISC Entities

To describe DISC entities:

```
DISC entity-spec
```

NOTE: You can use a D-series measurement application to measure disks on NonStop S-series servers if the application specifies all disks (ADD DISC *) or specifies only *$disk* or *$disk (cpu)*. If an application specifies *controller*, *channel*, or *unit*, you must modify the entity identifiers to measure NonStop S-series disks.

DISC

collects information about one or more disks in the local system.

entity-spec

is specified as:

```
{ *                                                          }
{ $disk [ ( cpu [ , channel [ , ctrl [, unit ] ] ] ) ] }
{ $disk [ ( cpu [ , svnet [ , group [ , module  [ , slot
                                [ , scsi-id ] ] ] ] ] )] }
{ $disk [ ( cpu [, svnet [, sac [, lun ] ] ] ) ]       }
{ $disk[path] }
```

*

measures all disks in all CPUs. If any specifiers follow the asterisk (*), you must use $*.

$*disk*

is the name of the disk to be measured. To indicate all disks, use an asterisk (*).

cpu

is the number of the CPU in which the disk is configured. The default is all CPUs. To indicate all CPUs, use an asterisk (*).

channel

(D-series) is the channel number of the disk to be measured. The default is all channels.

ctrl

(D-series) is the controller number of the disk to be measured. The default is all controllers.

unit

(D-series) is the unit number of the disk to be measured. The default is all units.

svnet

(G-series and later RVUs) ServerNet fabric used by this specific path to the device. The Measure subsystem measures all disk records relative to the specified fabric. The value is 0 for X fabric, 1 for Y fabric. The default is to measure records for both fabrics.

Disks connected by a ServerNet/DA do not have an X or Y fabric. To indicate these disks, use an asterisk (*). In this case, you can use an asterisk (*) in the LISTACTIVE command as long as the rest of the entity specification describes one disk.

Disks connected by an FCSA or CLIM are dual fabric adapters, that is, they are connected to both the X and Y fabrics. For these disks, only "*" or "X" should be specified in the LIST command, and only "X" should be specified in the LISTACTIVE command.

group

(G-series and later RVUs) is the group number of the disk to be measured. (The group number corresponds to the physical enclosure.) To indicate all groups, use an asterisk (*). The default is all groups.

module

(G-series and later RVUs) is the module number of the disk to be measured. Use an asterisk (*) to indicate all modules. The default is all modules.

slot

(G-series and later RVUs) is the slot number of the disk to be measured. To indicate all slots, use an asterisk (*). The default is all slots.

scsi-id

(G-series and later RVUs) is the SCSI port identifier of the disc to be measured. To indicate all SCSI ports, use an asterisk (*). The default is all SCSI ports. This is used for ServerNet/DA and FCSA discs, but not for CLIM discs.

sac

is the name of the SAC the disc is connected to.

lun

is the logical unit number within the CLIM. The NonStop operating system supports a range of 0 to 65535 for *lun*. Measure supports a range of 0 to 65534, after reserving 65535 (-1) for the wildcard (*).

path

applies to FCSA storage disks as well as storage CLIM disks. Valid values for *path* can be "-P", "-B", "-M" or "-MB" (specifying primary, backup, mirror-primary, mirror-backup, respectively.

DDL Record for DISC Entities (Legacy Style)

This is the Legacy Style DDL record for DISC entities. The fields included in BRIEF reports are in **boldface type**. This record will not change after the Measure G10 PVU.

The DDL record for G-series DISC entities is identical to the record for D-series DISC entities except:

- The CHANNEL field is redefined to SERVERNET.
- Different entity identification fields are used. The CTRL and UNIT fields are no longer used.
- A SCSI-ID identifier is added to identify disks connected by a ServerNet/DA or FCSA.
- Counters are added to measure busy time in the queue-based ServerNet environment. The READ-BUSY-TIME and WRITE-BUSY-TIME counters are no longer used.
- Counters are added to measure direct-bulk I/O operations.
- Existing I/O counters such as READS and WRITES include direct-bulk I/O operations.
- Longer byte-count fields are provided to reduce the possibility of field overflow. In G-series RVUs, each 32-bit byte-count field has a 64-bit counterpart. Similarly, a 16-bit byte count field has a 32-bit counterpart. In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. For information on using the 64-bit or 32-bit fields, see Usage Notes for G-Series DISC Entities (page 196).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.

```
RECORD disc. FILE is "disc" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                              type binary 16 unsigned.
 02 device-name                      type character 8.
 02 logical-device                   type binary 16 unsigned.
 02 ctrl                             type binary 16 unsigned.
 02 unit                             type binary 16 unsigned.
 02 device-type                      type binary 16 unsigned.
 02 device-subtype                   type binary 16 unsigned.
 02 disc-process-type                type binary 16 unsigned.

* counter value items:
 02 request-qtime                    type binary 64.
 02 request-qlen-max                 type binary 16 unsigned.
 02 requests                         type binary 32 unsigned.
 02 read-busy-time                   type binary 64.
 02 write-busy-time                  type binary 64.
 02 seek-busy-time                   type binary 64.
 02 reads                            type binary 32 unsigned.
 02 writes                           type binary 32 unsigned.
 02 seeks                            type binary 32 unsigned.
 02 input-bytes                      type binary 32 unsigned.
 02 output-bytes                     type binary 32 unsigned.
 02 swaps                            type binary 32 unsigned.
 02 cblks-inuse-qtime                type binary 64.
 02 cblks-inuse-max                  type binary 16 unsigned.
```

```
 02 ablks-inuse-qtime             type binary 64.
 02 ablks-inuse-max               type binary 16 unsigned.
 02 c                             occurs 4 times.
   03 hits                        type binary 32 unsigned.
   03 misses                      type binary 32 unsigned.
   03 faults                      type binary 32 unsigned.
   03 audit-buf-forces            type binary 32 unsigned.
   03 blks                        type binary 16 unsigned.
   03 blks-dirty-qtime            type binary 64.
   03 blks-dirty-max              type binary 16 unsigned.
 02 control-points                type binary 32 unsigned.
 02 control-point-writes          type binary 32 unsigned.
 02 free-space-ios                type binary 32 unsigned.
 02 requests-blocked              type binary 32 unsigned.

* new entity identification item for D10:
 02 channel                       type binary 16 unsigned.
 02 servernet                     type binary 16 unsigned.
 02 capacity                      type binary 64.

* new counter value items for D10:
 02 starting-free-space           type binary 64.
 02 ending-free-space             type binary 64.
 02 starting-free-blocks          type binary 32 unsigned.
 02 ending-free-blocks            type binary 32 unsigned.
 02 cw                            occurs 4 times.
   03 write-dirtys                type binary 32 unsigned.
   03 write-cleans                type binary 32 unsigned.
   03 write-misses                type binary 32 unsigned.
   03 block-splits                type binary 32 unsigned.
 02 volsem-qtime                  type binary 64.
 02 volsem-qlen-max               type binary 16 unsigned.

* SMS changes:
 02 storage-pool                  type character 8.

* new entity identification item for F40:
 02 config-name                    type character 64.
 02 adapter-name                   type character 64.
 02 SAC-name                       type character 64.
 02 GMS.
    03 group                       type binary 32 unsigned.
    03 module                      type binary 32 unsigned.
    03 slot                        type binary 32 unsigned.

* new counter value items for F40:
 02 read-qbusy-time                type binary 64.
 02 read-qtime                     type binary 64.
 02 read-qlen-max                  type binary 16 unsigned.
 02 write-qbusy-time               type binary 64.
 02 write-qtime                    type binary 64.
 02 write-qlen-max                 type binary 16 unsigned.
 02 device-qbusy-time              type binary 64.
 02 input-bytes-f                  type binary 64.
 02 output-bytes-f                 type binary 64.

* New counters for G05
 02 scsi-id                        type binary 64.

* New counters in G05:
 02 DBIO-reads                     type binary 32 unsigned.
 02 DBIO-writes                    type binary 32 unsigned.

* New counters in G07:
 02 cn                            occurs 4 times.
```

```
    03 blks                          type binary 32 unsigned.
    03 blks-inuse-start              type binary 32 unsigned.
    03 blks-inuse-end                type binary 32 unsigned.

* New counters in G08:
 02 defreqs                          type binary 32 unsigned.
 02 defreq-qtime                     type binary 64.
 02 defreq-qlen-max                  type binary 16 unsigned.
 02 deferred-qtime                   type binary 64.
 02 deferred-qlen-max                type binary 16 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for DISC Entities (ZMS Style)

The ZMS Style DDL record for DISC entities is supported on Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsdisc-id.
 02 pin                           type binary 16 unsigned.
 02 device-type                   type binary 16 unsigned.
 02 device-subtype                type binary 16 unsigned.
 02 servernet                     type binary 16 unsigned.
 02 device-name                   type character 8.
 02 logical-device                type binary 32 unsigned.
 02 GMS.
    03 group                      type binary 32 unsigned.
    03 module                     type binary 32 unsigned.
    03 slot                       type binary 32 unsigned.
 02 SCSI-id                       type binary 64.
 02 plpt redefines SCSI-id.
    03 plpt-flags                 type binary 8 unsigned.
    03 path                       type binary 8 unsigned.
    03 lun                        type binary 16 unsigned.
    03 reserved-2                 type binary 16 unsigned.
    03 target-id                  type binary 16 insigned.
 02 config-name                   type character 64.
 02 adapter-name                  type character 64.
 02 SAC-name                      type character 64.
 02 capacity                      type binary 64.
 02 storage-pool                  type character 8.
 02 disc-process-type             type binary 16 unsigned.
 02 reserved-1                    type character 6.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsdisc-ctrs.
 02 request-qtime                 type binary 64.
 02 requests                      type binary 64.
 02 reads                         type binary 64.
 02 writes                        type binary 64.
 02 input-bytes                   type binary 64.
 02 output-bytes                  type binary 64.
 02 swaps                         type binary 64.
 02 control-points                type binary 64.
 02 control-point-writes          type binary 64.
 02 free-space-ios                type binary 64.
 02 requests-blocked              type binary 64.
 02 starting-free-space           type binary 64.
 02 ending-free-space             type binary 64.
```

```
 02 starting-free-blocks          type binary 32 unsigned.
 02 ending-free-blocks            type binary 32 unsigned.
 02 volsem-qtime                  type binary 64.
 02 read-qbusy-time               type binary 64.
 02 read-qtime                    type binary 64.
 02 write-qbusy-time              type binary 64.
 02 write-qtime                   type binary 64.
 02 device-qbusy-time             type binary 64.
 02 DBIO-reads                    type binary 64.
 02 DBIO-writes                   type binary 64.
 02 defreqs                       type binary 64.
 02 defreq-qtime                  type binary 64.
 02 deferred-qtime                type binary 64.
 02 c                             occurs 8 times.
   03 blks                       type binary 32 unsigned.
   03 block-size                 type binary 32 unsigned.
   03 blocks-inuse-start         type binary 32 unsigned.
   03 blocks-inuse-end           type binary 32 unsigned.
   03 block-splits               type binary 64.
   03 hits                       type binary 64.
   03 misses                     type binary 64.
   03 faults                     type binary 64.
   03 audit-buf-forces           type binary 64.
   03 blks-dirty-qtime           type binary 64.
   03 write-cleans               type binary 64.
   03 write-dirtys               type binary 64.
   03 write-misses               type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsdisc. FILE is "zmsdisc" ENTRY-SEQUENCED.
 02 hdr                           type zmsheader.
 02 ctr                           type zmsdisc-ctrs.
 02 id                            type zmsdisc-id.
end
```

PIN

Process identification number of the disk process.

DEVICE-NAME

Name of the device.

LOGICAL-DEVICE

Logical device number of the device.

CTRL

(Legacy Style only) For D-series RVUs, controller number of the device. For G-series RVUs, no longer used; returns zero.

UNIT

(Legacy Style only) For D-series RVUs, unit number of the device. For G-series RVUs, no longer used; returns zero.

DEVICE-TYPE

A value of 3, which indicates a disk volume.

DEVICE-SUBTYPE

An additional identifier for DEVICE-TYPE. For a list of subtype values for disks, see the *System Generation Manual*.

DISC-PROCESS-TYPE

The disk process type, DP2.

REQUEST-QTIME

The time that requests spent on the disk process internal queue (includes any currently active requests). When reading a request, the disk process places it on the internal queue. After processing the request, the disk process removes the request from the internal queue. To determine the length of the disk process external queue (requests waiting to be read by the disk process), measure the disk process and examine the RECV-QTIME counter of the PROCESS entity.

Counter type: Queue.

REQUEST-QLEN-MAX

(Legacy Style only) Maximum number of items on the queue described by the REQUEST-QTIME counter.

Counter type: Max queue.

REQUESTS

Number of I/O requests (READ, WRITE, FILEINFO) received by the disk process. The FILE entity records most of these requests on a file-by-file basis. To determine how long the requests were queued before being read by the disk process, measure the process and examine the RECV-QTIME counter of the PROCESS entity. After reading a request, the disk process places it on an internal queue. REQUEST-QTIME measures the length of the internal queue.

Counter type: Incrementing.

READ-BUSY-TIME

(Legacy Style only) For D-series RVUs, the time spent reading from the disk. This counter includes the time spent reading data and positioning the disk heads to read the data.

For G-series RVUs, no longer used; returns zero.

Counter type: Busy.

WRITE-BUSY-TIME

(Legacy Style only) For D-series RVUs, the time spent writing to the disk. This counter includes the time spent writing data and positioning the disk heads to write the data.

For G-series RVUs, no longer used; returns zero.

Counter type: Busy.

SEEK-BUSY-TIME

(Legacy Style only) No longer used; returns zeros. All seek operations are implicit in read and write operations.

READS

Number of physical read operations (from disk to memory) performed by the disk process. In addition to programmatic read operations, many commands (for example, LISTHEADERS and LISTDEFECTS) also read from the disk.

Counter type: Incrementing.

WRITES

Number of physical write operations (from memory to disk) performed by the disk process. In addition to programmatic write operations, many commands (for example, FORMAT and LOADMICROCODE) also write to the disk.

Counter type: Incrementing.

SEEKS

(Legacy Style only) No longer used; returns zeros. All seek operations are implicit in read and write operations.

INPUT-BYTES

Number of bytes read from the disk. In addition to programmatic read operations, write operations and various commands such as LISTHEADERS and LISTDEFECTS also cause bytes to be read from the disk.

Because the I/O process modifies this counter before an I/O operation, if the read fails, the byte count might not be accurate.

Write operations (see OUTPUT-BYTES) and various commands such as LISTHEADERS and LISTDEFECTS also cause bytes to be read from the disk.

The disk process reads one or more blocks of data during a read operation, depending on the structure of the file and amount of data being read:

- For unstructured files, the disk process reads at least one block of data. If a record crosses block boundaries, the disk process must read two blocks of data. Also, if you are writing a partially filled block to disk, the disk process must read the block, merge the new data into the block, and then write the block to disk.
- For structured files, the disk process reads a block of data as specified by the file block size.
- For key-sequenced files, to read a data block, the disk process might have to read one or more index blocks as well.

Because the disk process modifies this counter before an I/O operation, if the read fails, the byte count might not be accurate.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the INPUT-BYTES-F field is a 64-bit version of INPUT-BYTES.

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the disk. In addition to programmatic write operations, many commands (for example, FORMAT and LOADMICROCODE) also write to the disk.

Because the I/O process modifies this counter before the I/O operation, if the write fails, the byte count might not be accurate.

In addition to programmatic write operations, many commands (for example, FORMAT and LOADMICROCODE) also write to the disk.

The disk process writes one or more blocks of data on a write operation, depending on the structure of the file and amount of data being written:

- For unstructured files, the disk process writes at least one block of data. If a record crosses block boundaries, the disk process must write two blocks of data. In addition, if you are

writing a partially filled block to disk, the disk process must read the block (counted by INPUT-BYTES), merge the new data into the block, and then write the block to disk.

- For structured files, the disk process writes a block of data as specified by the file block size.
- For key-sequenced files, the disk process writes to the data block and might also have to update the index block. Buffered writes can save I/Os because the disk process collects the updates and writes them out only when necessary.

Buffered writes can save I/Os because the disk process collects the updates and writes them out only when necessary.

Because the I/O process modifies this counter before the I/O operation, if the write fails, the byte count might not be accurate.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the OUTPUT-BYTES-F field is a 64-bit version of OUTPUT-BYTES.

Counter type: Accumulating.

SWAPS

Number of swap operations (both into and out of memory) performed for the memory manager.

Unlike CPU swaps, which are not dependent on file block size, the disk process performs I/Os based on the block size of the file and might need to perform more than one I/O to swap the needed data. The DISC SWAPS counter is advanced for each I/O. Therefore, total disk swaps might be greater than total CPU swaps. Only updated pages are swapped. Because code pages are never updated, they are never swapped out of memory.

Counter type: Incrementing.

CBLKS-INUSE-QTIME

(Legacy Style only) No longer used.

CBLKS-INUSE-MAX

(Legacy Style only) No longer used.

ABLKS-INUSE-QTIME

(Legacy Style only) No longer used.

ABLKS-INUSE-MAX

(Legacy Style only) No longer used.

C

Disk cache metrics. This field is divided into these subfields:

Subfield	Description	Counter Type
HITS	Number of times a required block is found in cache on a read or buffered write operation, which saves a disk I/O. This counter includes cache hits on both read and write operations.	Incrementing
MISSES	Number of times a required block is not found in cache on a read operation, which causes a disk I/O. Read operations that do not go through cache are not included in this counter.	Incrementing

Subfield	Description	Counter Type
FAULTS	Number of times the cache tables indicate a required block is in cache, but not because the memory manager is using it for another purpose. Faults occur when a cache is too large or physical memory is too small for a given application. Fault rates greater than 5% of all I/O operations generally indicate you need more memory.	Incrementing
AUDIT-BUF-FORCES	Number of times the disk process had to write to the audit-trail volume before it could write a dirty cache block and thereby free that block for another use. This value should be close to zero.	Incrementing
BLKS	Number of cache blocks allocated. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the maximum number of cache blocks allocated. BLKS is not a counter but a value obtained from an internal disk process data structure.	N.A.
BLKS-DIRTY-QTIME	The time that cache blocks spent dirty. A cache block is dirty when a buffered write operation writes data to it. After the disk process writes the dirty cache block to disk, the block is clean.	Queue
BLKS-DIRTY-MAX	(Legacy Style only) Maximum number of items on the queue described by the BLKS-DIRTY-QTIME counter.	Max queue
BLOCK-SIZE	(ZMS Style only) Cache block size, in bytes.	N.A.
BLOCKS-INUSE-START	(ZMS Style only) The number of cache blocks that were in use at the beginning of the measurement. This value is part of the structured DISC record but does not appear in Legacy Style DISC reports. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the total number of cache blocks in use at the beginning of the measurement.	Snapshot
BLOCKS-INUSE-END	ZMS Style only) The number of cache blocks that were in use at the end of the measurement or at measurement interval copy time. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the total number of cache blocks in use at the end of the measurement or at measurement interval copy time.	Snapshot
BLOCK-SPLITS	(ZMS Style only) Number of block splits (per block size) for the volume.	Incrementing
WRITE-DIRTYS	(ZMS Style only) Number of times a required block is found dirty in cache for a write operation.	Accumulating
WRITE-CLEANS	(ZMS Style only) Number of times a required block is in cache for a write operation. This saves a found clean disk read per write.	Accumulating
WRITE-MISSES	(ZMS Style only) Number of times a required block is not found in cache for a write operation. This requires a disk read before the write can be performed.	Accumulating

In Measure output, each subfield provides four values, one for each size of DP2 cache block:

Field label	Cache block size
C0	512 bytes
C1	1024 bytes
C2	2048 bytes

Field label	Cache block size
C3	4096 bytes
C4	8192 bytes
C5	16346 bytes
C6	32768 bytes
C7	65536 bytes

CONTROL-POINTS

Number of times the disk process wrote a TMF control point record to its audit-trail file. For a discussion of TMF auditing, see the *TMF Operations and Recovery Guide*.

Counter type: Incrementing.

CONTROL-POINT-WRITES

Number of dirty cache blocks that the disk process wrote to disk at TMF control points. For a discussion of TMF auditing, see the *TMF Operations and Recovery Guide*.

Counter type: Incrementing.

FREE-SPACE-IOS

Number of I/O operations to the disk free space table.

The disk free space table lists each available free block on the disk. The disk process modifies the table when file extents are allocated or deallocated. Choosing the proper extent sizes and avoiding unnecessary file creations and deletions can help reduce the number of free space I/Os.

Counter type: Incrementing.

REQUESTS-BLOCKED

Number of times an operation had to wait because the requested file or record was already locked. Typically, a record is locked by a LOCKFILE request, a LOCKREC request, or a TMF auditing operation.

Counter type: Incrementing.

CHANNEL

(Legacy Style only) For D-series RVUs, channel number of the device. For G-series and later RVUs, redefined. See SERVERNET.

SERVERNET

Current access path to this disk object. The value is 0 for X fabric or for paths that go through dual fabric adapters.

Disks connected by a ServerNet/DA do not have an X or Y fabric. For these disks, the ServerNet field (SvNet in reports) is displayed as an asterisk (*).

Disks connected by an FCSA or CLIM based adapter are dual fabric adapters, that is, they are connected to both X and Y fabrics. For these devices, the ServerNet field is displayed as "X".

CAPACITY

Disk volume capacity in bytes. It can be used in the IF and BY clauses.

STARTING-FREE-SPACE

Number of bytes free on the volume when the measurement starts. This counter is displayed by MEASCOM when the REPORT STYLE is ZMS. It can be used in the IF and BY clauses as well as with the PLOT commands.

This counter is updated only for the primary path record. The primary path can be obtained from the `MEASURE LIST DISC` command (Measure PVU H03 or later) or the `SCF INFO` *`DISCNAME`*`, DETAIL` command.

Counter type: Accumulating/Snapshot.

ENDING-FREE-SPACE

The number of bytes free on the volume at the time indicated by the TO time. This counter is updated only for the primary path record. The RATE attribute has no effect on how this counter is displayed. The primary path can be obtained from the `MEASURE LIST DISC` command (Measure PVU H03 or later) or the `SCF INFO` *`DISCNAME`*`, DETAIL` command.

Counter type: Accumulating/Snapshot

STARTING-FREE-BLOCKS

Number of free entries in the free space table for the volume when the measurement starts. The free space table is an internal data structure maintained by DP2. This counter is displayed by MEASCOM when the REPORT STYLE is ZMS. It can be used in the IF and BY clauses as well as with the PLOT commands.

This counter is updated only for the primary path record. The primary path can be obtained from the `MEASURE LIST DISC` command (Measure PVU H03 or later) or the `SCF INFO` *`DISCNAME`*`, DETAIL` command.

Counter type: Accumulating/Snapshot

ENDING-FREE-BLOCKS

The number of free blocks on the volume at the time indicated by the TO time. The counter is updated only for the primary path record. The RATE attribute has no effect on how this counter is displayed. The primary path can be obtained from the `MEASURE LIST DISC` command (Measure PVU H03 or later) or the `SCF INFO` *`DISCNAME`*`, DETAIL` command.

Counter type: Accumulating/Snapshot

CW

(Legacy Style only) Cache write metrics. This field is divided into these subfields:

Subfield	Description	Counter Type
WRITE-DIRTYS	Number of times a required block is found dirty in cache for a write operation.	Accumulating
WRITE-CLEANS	Number of times a required block is found clean in cache for a write operation. This saves a disk read per write.	Accumulating
WRITE-MISSES	Number of times a required block is not found in cache for a write operation. This requires a disk read before the write can be performed.	Accumulating
BLOCK-SPLITS	Number of block splits (per block size) for the volume.	Incrementing

In Measure output, each subfield provides four values, one for each size of DP2 cache block:

Field Name	Cache Block Size
C0	512 bytes
C1	1024 bytes
C2	2048 bytes
C3	4096 bytes

VOLSEM-QTIME

The time that the requests spent waiting for the volume semaphore (exclusive and shared modes).

With REPORT RATE OFF, this is the total queue time for volume semaphore. With REPORT RATE ON, this is the average number of requests in the queue.

Under normal conditions, the value of this counter should be close to zero. Otherwise, it indicates some operations are holding the volume semaphore in exclusive mode, causing all other request operations that require the volume semaphore in share mode to wait.

Counter type: Queue.

VOLSEM-QLEN-MAX

(Legacy Style only) Maximum number of requests in the queue described by the VOLSEM-QTIME counter.

Counter type: Max queue.

STORAGE-POOL

Name of the Storage Management Foundation (SMF) storage pool to which the measured disk is assigned (process name of the storage pool process).

CONFIG-NAME

Logical name associated with this physical disk. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

ADAPTER-NAME

Logical name associated with the adapter in which the SAC resides. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

SAC-NAME

Logical name associated with the SAC used in this path to the physical volume. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

GMS

Physical location address (group, module, slot). The GMS field is divided into three subfields:

GROUP	is the group number.
MODULE	is the module number.
SLOT	is the slot number.

You can use the keywords GROUP, MODULE, and SLOT in the IF and BY clauses of the LIST and LISTALL commands.

READ-QBUSY-TIME

The time spent in a state in which read requests of any number were queued to this disk in this processor.

Counter type: Queue busy.

READ-QTIME

Total time spent by read requests queued to this disk.

Counter type: Queue.

READ-QLEN-MAX

(Legacy Style only) Maximum number of read requests queued to this disk.

Counter type: Max queue.

WRITE-QBUSY-TIME

The time spent in a state in which write requests of any number were queued to this disk in this processor

Counter type: Queue busy.

WRITE-QTIME

Total time spent by write requests queued to this disk.

Counter type: Queue.

WRITE-QLEN-MAX

(Legacy Style only) Maximum number of write requests queued to this disk at any time since measurement was started.

Counter type: Max queue.

DEVICE-QBUSY-TIME

The time spent in a state in which requests of any number or type were queued to this disk in this processor.

Because of concurrent activity in the write and read queues, WRITE-QBUSY-TIME plus READ-QBUSY-TIME does not equal DEVICE-QBUSY-TIME. For more information, see Usage Notes for G-Series DISC Entities (page 196).

Counter type: Queue busy.

INPUT-BYTES-F

(Legacy Style only) For G-series RVUs, same as INPUT-BYTES but accommodates larger values (64 bits rather than 32 bits).

OUTPUT-BYTES-F

(Legacy Style only) For G-series RVUs, same as OUTPUT-BYTES but accommodates larger values (64 bits rather than 32 bits).

SCSI-ID

SCSI port identifier for this disk.

PLPT structure

Starting with the Measure H03 PVU, this structure replaces the SCSI-ID field. More attributes were necessary to identify a CLIM device, so the SCSI-ID field (which had a range of 0 to 999)

was reduced to 16 bits and became the PLPT.TARGET-ID field. This structure is used in descriptor and data records. The PLPT subfields are PLPT-FLAGS, LUN, PATH, and TARGET-ID.

PLPT-FLAGS

The PLPT-FLAGS are: MEAS_CLIM_REL, MEAS_PATH_SEL and MEAS_CLIM_DEVICE.

MEAS_CLIM_REL	This identifies this field as a PLPT structure rather than a SCSI-ID field. Since the *scsi-id* could have a -1F (wildcard) in it (if it is a descriptor), this flag only has meaning if the PLPT-FLAGS field is not all 1s. To maintain downward compatibility, this bit is not set for descriptors returned via MEASLISTCONFIG unless the device configuration was specifically for a CLIM device.
MEAS_PATH_SEL	This indicates that selection is only by *path* and device name, and not other attributes like *lun* or *scsi-id*. This flag only has meaning if the PLPT-FLAGS field is not all 1s and is only used in descriptors.
MEAS_CLIM_DEVICE	This indicates that this descriptor or data record is for a CLIM device.

LUN

Logical unit number. This is an unique number assigned to a disk drive within a storage CLIM. This is unique only within a particular CLIM. For pre-CLIM devices, this is 0. For CLIM devices, the valid range is 0 to 65534. This field is treated as an unsigned number. If *lun* is in a descriptor, the wildcard is all 1s

PATH

Connection between the NonStop operating system and the disk drive. On the NonStop operating system, there can be two paths (backup and primary) to a disk drive. For CLIM devices, due to the fault tolerance requirement, this requires two storage CLIMs, with one path through each CLIM. A mirrored NonStop operating system volume actually consists of two disk drives, so such a volume can have four paths associated with it. The valid ranges are 0 to 3 for Primary, Backup, Mirror and Mirror-backup, respectively. If *path* is in a descriptor, the wildcard is all 1s.

TARGET-ID

This is the SCSI device ID returned by the SCSI Interface Manager. Prior to the Measure H03 PVU, this was stored in the 64-bit SCSI-ID field. This value only requires 16 bits, so starting with the Measure H03 PVU this value is stored in this smaller field in order to use the free upper 48 bits for *lun* and *path*. The valid values for *target-id* are 0 to 999. If *target-id* is in a descriptor, the wildcard is all 1s.

DBIO-READS

The number of read operations to the measured device that were enabled for direct bulk I/O to a requesting application processor. This counter is a subset of READS.

Counter type: Incrementing.

DBIO-WRITES

The number of write operations to the measured device that were enabled for direct bulk I/O from a requesting application processor. This counter is a subset of WRITES.

Counter type: Incrementing.

CN

(Legacy Style only) Disk cache metrics. This field is divided into these subfields:

Subfield	Description	Counter Type
BLKS	Number of cache blocks allocated. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the maximum number of cache blocks allocated. BLKS is not a counter but a value obtained from an internal disk process data structure.	N.A.
BLKS-INUSE-START	The number of cache blocks that were in use at the beginning of the measurement. This value is part of the structured DISC record but does not appear in Legacy Style DISC reports. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the total number of cache blocks in use at the beginning of the measurement.	Snapshot
BLKS-INUSE-END	The number of cache blocks that were in use at the end of the measurement or at measurement interval copy time. When TOTALS ONLY or TOTALS INCLUDE is used, this value is the total number of cache blocks in use at the end of the measurement or at measurement interval copy time.	Snapshot

In Measure output, each subfield provides four values, one for each size of DP2 cache block:

Field Label	Cache Block Size
C0	512 bytes
C1	1024 bytes
C2	2048 bytes
C3	4096 bytes

DEFREQS

The number of requests received by the disk process that are subject to deferral. This counter is a subset of REQUESTS.

Counter type: Incrementing.

DEFREQ-QTIME

The time (in microseconds) that requests subject to deferral spent on the disk process internal queue (including any current active or deferred requests). This counter is a subset of REQUEST-QTIME.

Counter type: Queue.

DEFREQ-QLEN-MAX

(Legacy Style only) The maximum number of items on the queue described by the DEFREQ-TIME counter.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max queue.

DEFERRED-QTIME

The time (in microseconds) that requests subject to deferral actually spent deferred. This counter is a subset of DEFREQ-QTIME.

Counter type: Queue.

DEFERRED-QLEN-MAX

(Legacy Style only) The maximum number of items on the queue described by the DEFERRED-QTIME counter.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max queue.

Usage Notes for All DISC Entities

- The DISC counters track all disk activity, including both file activity and internal activities such as writing to the Free Space Table, writing to the undo area, and writing file and volume labels. Use the DISC counters to compare and balance the workload of two or more disks.

  For information on I/O and caching, see the appropriate system description manual. For information on TMF auditing and audit trails written to disk, see the *TMF Operations and Recovery Guide*.

  The keyword DISK is interchangeable with DISC in any MEASCOM command except HELP.
- The Measure performance monitor does not measure SMF Virtual Disc Processes (VDPs) as DISC entities. For example, if $V1 is a VDP, the command $ADD DISK $V1 does not collect data for $V1 or for the physical disks configured under $V1. However, no error is returned.
- SUBSYSTEM-VERSION for ZMSDISC records is provided by DP2.
- DISC reports include two fields that are calculated from DISC counters. These fields are not defined in the DISC DDL record:

  DISC-BUSY-TIME

  (Legacy Style only) The sum of the DISC WRITE-BUSY-TIME and READ-BUSY-TIME counters.

  DISC-RATE

  The combined rate of READ and WRITE operations to the physical drive. This value is included in the BRIEF format report.

  Counter type: Incrementing.

Usage Notes for G-Series DISC Entities

- To measure device-busy time, use the DEVICE-QBUSY-TIME counter. DEVICE-QBUSY-TIME is a replacement for the D-series field DISC-BUSY-TIME.

  In D-series RVUs, DISC-BUSY-TIME is calculated by adding the values of the DISC WRITE-BUSY-TIME and READ-BUSY-TIME counters. However, in G-series RVUs, DEVICE-QBUSY-TIME does not equal WRITE-QBUSY-TIME plus READ-QBUSY-TIME. In the ServerNet environment, you can have active requests in the read queue and write queue at the same time. Therefore, some overlap can exist between the READ-QBUSY-TIME and WRITE-QBUSY-TIME counters, and adding their values does not produce an accurate busy-time measurement.
- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field INPUT-BYTES-F collects the same data as

the 32-bit field INPUT-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

The 32-bit fields are currently active and continue to return values. If no field overflow occurs, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

Convert your applications to use the 64-bit fields; 32-bit fields might be deactivated in a future PVU.

In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST DEVICE BY INPUT-BYTES, not LIST DEVICE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

- Direct bulk I/O requests are counted as read and write requests to the disk. READ-BUSY-QTIME, READ-QTIME, INPUT-BYTES, and similar counters are updated for both direct bulk I/O and standard I/O.

  If a disk is used for both standard and direct bulk I/O, you cannot determine the byte counts for direct bulk I/O from the DISC counters alone. A detailed analysis of the FILE records for each application process requesting DBIO transfers could provide this information.

- The 32-bit CN[N].BLKS field collects the same data as the older 16-bit C[N].BLKS field. If no field overflow occurs, the 16-bit and the 32-bit fields return the same value. If a 16-bit overflows, the corresponding 32-bit field returns the correct value, and the 16-bit field returns a value of -1. The ERROR field for the measure entity returns -2 to indicate an overflow condition.

  Convert your applications to use the 32-bit fields; 16-bit fields might be deactivated in a future PVU. For MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields.

  In Measure G08 and later PVUs, new counters better evaluate the effectiveness of DP2's Mixed Workload Algorithm in application processing and in assignment of process priorities to better utilize this feature. Measure G08 counters (DEFREQS, DEFREQ-QLEN-MAX, DEFERRED-QTIME, DEFERRED-QLEN-MAX) track the request time and deferred time for requests subject to deferral.

  The Mixed Workload design gives you some control over the impact of running a low-priority SQL query concurrently with high-priority workloads. The lower the priority of the SQL query, the less impact to the high-priority response time. The speed of the query is also reduced as the priority is reduced, especially for a parallel query that has multiple volumes in the same CPU.

  User control over this mixed workload is accomplished by query deferral when concurrent higher priority workloads are active for the volumes. This deferral is small at priority 150 and increases as the priority of the query is reduced. To disable the deferral, run the query at a priority greater than 150.

  The Measure counters for deferrable q-time and deferred q-time provide some indication of the amount of deferral. If the SQL query requires too much time to complete, you see a high ratio between deferrable q-time and deferred q-time. If response time increases significantly during low-priority SQL query activity, you see a low ration between deferrable q-time and deferred q-time. Priority adjustments for the SQL query should correspond to changes in the ratio.

  Changes in platform or application might require some adjustments in the priority ad-hoc SQL query activity. The design does not defer query activity until the query has processed some minimal amount of data. However, some OLTP-type SQL query applications might need to run above priority 150 to avoid deferral.

Future RVUs might adjust the mixed workload design to let applications take full advantage of the increase in CPU and disk capacity. The adjustment in design might require priority adjustments to the mixed workload environment. The most recent changes of the mixed workload algorithm include adjustments to let you increase the speed of the query but cause only moderate increases in response time.

NOTE: For guidelines to control mixed workload applications using priority settings, see the white paper Mixed Workload Priority Guide, Version Id.: 1.0, Date: September 2001.

Usage Notes for H-Series and J-Series DISC Entities

In H-series and J-series RVUs, all byte-count fields have 64 bits. Field names ending in -F are no longer used in ZMS style but remain available to applications that request data in legacy style.

Usage Notes for Measure H03 and J01 PVUs

For storage CLIM devices, the DISC entity specification syntax is updated to include *lun* and *path* and not use *scsi-id*. As with FCSA devices, this entity specification can be used in the ADD *entity-type*, DELETE *entity-type*, LIST *entity-type*, LISTACTIVE *entity-type*, and LISTALL *entity-type* commands.

Examples

Command	Description
ADD DISC $SYSTEM	Measure disk named $SYSTEM (all paths)
ADD DISC *	Measure all disks
ADD DISC $* (2)	Measure all disks in CPU 2
ADD DISC $* (*, *, 110, 3, 1, 2)	Measure all disks attached to a controller whose GMS is 110,3,1 and whose scsi-id is 2 (ServerNet/DA or FCSA storage devices only)
ADD DISC $SYSTEM-B	Measure backup path of $SYSTEM.
ADD DISC $* (*,*,CLIM3)	Measure all disks attached to the SAC named CLIM3

DISCOPEN

The DISCOPEN entity measures the I/O operations performed by the disk process on a specified file (physical file access) on behalf of an individual open of a file. The measured file must be local to the system being measured, but the process accessing the measured file can be local or remote.

In Measure G09 and later PVUs, the DISCOPEN entity supports the use of OSS file pathnames. It displays OSS file pathnames in DISCOPEN reports and provides direct mapping between external structured records and OSSNAMES structured record output.

In Measure H01 and later PVUs, MEASCOM accepts ANSI SQL names in a DISCOPEN entity specification.

Entity Specification Syntax for DISCOPEN Entities

To describe a DISCOPEN entity:

```
DISCOPEN entity-spec [,entity-spec] ...
```

DISCOPEN

collects information about physical access to one or more disk file opens. The measured file must be local. However, the processes accessing that file can be local, remote, or both.

entity-spec

is defined as:

```
filename [(opener)]
```

where *filename* can be any of the following names:

```
{ * }
{ [[$device.]subvolume.]filename[:crvsn] }
{ [$device.]#tempfile }
{ "pathname" }
{ '{TABLE|INDEX} [[catalog.]schema.]object[ PARTITION partition]' }
{ 'SCHEMA [catalog.]schema'  }
{ 'CATALOG {catalog|*}'  }
```

where *catalog*, *schema*, *object* and *partition* are ANSI SQL identifiers. Omitted *catalog* and *schema* fields are resolved by the MEASCOM *catalog* and *schema* environment values.

Wildcard syntax is:

```
TABLE [[catalogC.]schemaS.]tableT - all partitions in tableT
INDEX [[catalogC.]schemaS.]indexI - all index partitions in indexI
SCHEMA [catalogC.]schemaS - all objects in schemaS
CATALOG catalogC - all objects in catalogC
CATALOG * - all objects visible on the node
```

*

measures all file opens.

$*device*

is the name of the volume (device) that contains the file to be measured. To indicate all volumes, use an asterisk (*). The default is the current default volume.

subvolume

is the name of the subvolume that contains the file to be measured. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume.

filename

is the name of the file to be measured. To indicate all files except temporary files, use an asterisk (*). Temporary files are specified separately.

:*CRVSN*

in Measure G10 and later, is the timestamp, creation version serial number, or file name extension necessary to form a unique file name. Use this option to guarantee file name uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

system-name

is the name of the system executing the opener process (the process that accesses *filename*). The default is the local system.

cpu

is the number of the CPU on which the opener process is running. The default is all CPUs. Use *cpu* and *pin* to specify the owner of the opener process.

pin

is the process identification number of the opener process. To indicate all processes, use an asterisk (*). The default is all processes.

#tempfile

is the file identification number of a temporary file to be measured. To indicate all temporary files, use an asterisk (*).

"*pname*"

can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

catalog, schema, object, partition

in Measure H01 and later PVUs, are ANSI SQL identifiers. When omitted, catalog and schema fields are resolved by the MEASCOM catalog and schema environment values. If the name space keywords TABLE or INDEX are omitted, TABLE is the assumed default.

DDL Record for DISCOPEN Entities (Legacy Style)

This is the Legacy Style DDL record for DISCOPEN entities. The fields included in BRIEF reports are in **boldface type**.

The DDL record for G-series DISCOPEN entities is identical to the record for D-series DISCOPEN entities, except for:

- The counters added to support direct bulk I/O (existing I/O counters such as DRIVER-INPUT-CALLS also include direct-bulk I/O operations)
- The display of DP2 busy time on behalf of the SQL operation time
- New identifiers for OSS file pathname support

```
RECORD discopen. FILE is "discopen" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 opener-cpu                  type binary 16 unsigned.
 02 opener-pin                  type binary 16 unsigned.
 02 ocb-number                  type binary 16 unsigned.
 02 opener-system-name          type character 8.
 02 file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 file-type                   type binary 16 unsigned.

* counter value items:
 02 driver-input-calls          type binary 32 unsigned.
 02 driver-output-calls         type binary 32 unsigned.
 02 cache-hits                  type binary 32 unsigned.
 02 cache-write-hits            type binary 32 unsigned.
 02 block-splits                type binary 32 unsigned.
```

```
 02 requests-blocked           type binary 32 unsigned.

 * new entity identification items for D10:
 02 record-type                type binary 16 unsigned.

* new counter value items for D10:
 02 requests                   type binary 32 unsigned.
 02 lockwait-time              type binary 64.
 02 max-lockwait-time          type binary 64.
 02 lock-timeouts              type binary 32 unsigned.
 02 lock-bounces               type binary 32 unsigned.
 02 cache-write-cleans         type binary 32 unsigned.

* SMS changes:
 02 device-name                type character 8.
 02 storage-pool               type character 8.

02 dbio-input-calls                type binary 32 unsigned.
02 dbio-output-calls               type binary 32 unsigned.

* New counters for G08:
02 SQL-operation-time              type binary 64.

* New identifiers for OSS file pathname support:
02 file-name-mid.
   03 pathid                       type character 24.
   03 crvsn                        type character 6.
end
```

DDL Record for DISCOPEN Entities (ZMS Style)

The ZMS style DDL record for DISCOPEN entities is supported on Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsdopen-id.
 02 opener-cpu                 type binary 16 unsigned.
 02 opener-pin                 type binary 16 unsigned.
 02 ocb-number                 type binary 16 unsigned.
 02 file-type                  type binary 16 unsigned.
 02 opener-system-name         type character 8.
 02 file-name.
    03 volume                  type character 8.
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 device-name                type character 8.
 02 storage-pool               type character 8.
 02 file-name-MID.
    03 PATHID                  type character 24.
    03 CRVSN                   type character 6.
 02 record-type                type binary 16 unsigned.
end
```

Counter Fields DDL Definition

```
 02 driver-input-calls         type binary 64.
 02 driver-output-calls        type binary 64.
 02 cache-hits                 type binary 64.
 02 cache-write-hits           type binary 64.
 02 block-splits               type binary 64.
 02 requests-blocked           type binary 64.
 02 requests                   type binary 64.
 02 lockwait-time              type binary 64.
```

```
 02 max-lockwait-time          type binary 64.
 02 lock-timeouts              type binary 64.
 02 lock-bounces               type binary 64.
 02 cache-write-cleans         type binary 64.
 02 DBIO-input-calls           type binary 64.
 02 DBIO-output-calls          type binary 64.
 02 SQL-operation-time         type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsdopen. FILE is "zmsdopen" ENTRY-SEQUENCED.
 02 hdr                        type zmsheader.
 02 ctr                        type zmsdopen-ctrs.
 02 id                         type zmsdopen-id.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

OPENER-CPU

CPU number of the process that opened the measured file.

OPENER-PIN

Process identification number of the process that opened the measured file.

OCB-NUMBER

Identification number of the open control block (OCB).

OPENER-SYSTEM-NAME

Name of the system that opened the measured file.

FILE-NAME

Name of the measured file. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, FILE-NAME represents the physical file name that was specified when the disk file was opened. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, FILE-NAME represents a location-independent logical file name that was used when the disk file was opened. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

FILE-TYPE

One of these values, which indicates the file type:

0	Unstructured
1	Relative
2	Entry-sequenced
3	Key-sequenced

DRIVER-INPUT-CALLS

Number of read operations. The CACHE-HITS counter counts cache hits on read operations.

Counter type: Incrementing.

DRIVER-OUTPUT-CALLS

Number of write operations. The CACHE-WRITE-HITS counter counts cache hits on write operations.

Counter type: Incrementing.

CACHE-HITS

Number of times a read operation found the required block in cache, which saves a disk I/O.

If you write a partially filled block to disk, the disk process must read the block (possibly incrementing this counter), merge the new data into the block, and then write the block to disk. If you write to a new file, the disk process reads the new blocks from cache (incrementing this counter) unless no spare blocks are in cache.

The DRIVER-INPUT-CALLS counter counts all read operations, both cache hits and misses.

Counter type: Incrementing.

CACHE-WRITE-HITS

Number of times a write operation found the required block in cache and saved a read I/O (to bring the block into cache). A write I/O might also be saved if the file has the buffered option set ON (that is, if multiple writes can occur to a dirty block before it is physically written).

The DRIVER-OUTPUT-CALLS counter counts write operations that caused a disk I/O.

Counter type: Incrementing.

BLOCK-SPLITS

Number of blocks split during writes to the file.

A block split occurs when you add or lengthen a record in a key-sequenced file and the data block can no longer contain the entire record. Splitting a data block can cause the index block above the data block to split as well.

You should keep block splits to a minimum because they require several I/O operations, but you cannot entirely avoid them when inserting many new records in a key-sequenced file.

Counter type: Incrementing.

REQUESTS-BLOCKED

Number of times an I/O operation had to wait because the requested file or record was locked. Typically, a record lock is caused by a LOCKFILE request, a LOCKREC request, or a TMF auditing operation.

Counter type: Incrementing.

RECORD-TYPE

The type of internal counter record used to record the instrumentation for the DISCOPEN entity.

If the value is 0, the file opened was either a temporary file, a program (object file), or an edit file. For these types of files, a short form internal record is used, and values are not recorded for certain counters. For more information, see Usage Notes for All DISCOPEN Entities (page 205).

REQUESTS

Number of requests for this open.

Counter type: Incrementing.

LOCKWAIT-TIME

The time spent waiting for locks.

The REQUESTS-BLOCKED counter counts the number of blocked requests—that is, the number of times requests are queued to locks. When the REPORT RATE value is on, this counter returns the average wait time (LOCKWAIT-TIME divided by REQUESTS-BLOCKED).

Counter type: Busy.

MAX-LOCKWAIT-TIME

Maximum wait time per lock. This number can be used as a guide to set the timeout value in the application. REPORT RATE ON has no effect on this counter value.

Counter type: Max value.

LOCK-TIMEOUTS

Number of timeouts on locks (error 40). When a timeout value for a request expires, the file system (Enscribe and SQL/MP) sends a cancel request to the disk process. If the disk process finds the request still waiting for the lock to be granted, the request is removed from the lockwait queue, and this counter is advanced.

Counter type: Incrementing.

LOCK-BOUNCES

Number of bounced locks (FELOCKED or error 73) returned by the disk process to the file system. For SQL tables, a bounced lock is a result of using the CONTROL TABLE command with the RETURN-IF-LOCKED option. For Enscribe files, a bounced lock is a result of using SETMODE function 4 with Param1.<15> = 1 (alternate lock mode).

Counter type: Incrementing.

CACHE-WRITE-CLEANS

Number of times a required block is found clean in cache for a write operation. This saves a disk read to bring the block in.

Counter type: Incrementing.

DEVICE-NAME

Disk device on which the measured file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

STORAGE-POOL

Name of the SMF storage pool to which the measured disk is assigned (process name of the storage pool process).

DBIO-INPUT-CALLS

Number of direct bulk I/O read operations to a requesting processor. This counter is a subset of DRIVER-INPUT-CALLS.

DBIO-OUTPUT-CALLS

Number of direct bulk I/O write operations from a requesting processor. This counter is a subset of DRIVER-OUTPUT-CALLS.

SQL-OPERATION-TIME

The DP2 process busy-time accumulated while DP2 was working on SQL subset requests on behalf of this DISCOPEN.

Counter type: Busy.

FILE-NAME-MID

FILE-NAME-MID has two subfields: PATHID and CRVSN.

- PATHID is an internal format representation of an OSS file or SQL/MX file.
- CRVSN is a creation version serial number that identifies a unique instance of an OSS or SQL/MX file.

Usage Notes for All DISCOPEN Entities

- The Measure subsystem tracks only local entities. Therefore, the DISCOPEN entity includes I/O operations performed on local files only. Processes accessing the files can be local, remote, or both, depending on the measurement configuration.
- The DISCOPEN counter record is attached to the open control block (OCB) entry for a disk process. Because OCBs for backup disk processes are not measured, restart DISCOPEN measurements if the primary disk process is disabled or moved.
- There are two forms of DDL record for the DISCOPEN entity: a short form and a long form. The long record contains several additional counters that measure file locking. Not all file opens require these measurements, so the MEASCTL process allocates the short or long record at measurement time, based on the file type. The short record is allocated to temporary files ($vol.#nnnnnnn), edit files (file code 101), and object files (file code 100). The long record format is allocated to all other file types.

  The long record format is used for structured output files. If the data in the record is collected with the short format, these counter fields contain zeros in the structured file:

REQUESTS	MAX-LOCKWAIT-TIME	LOCK-BOUNCES
LOCKWAIT-TIME	LOCK-TIMEOUTS	CACHE-WRITE-CLEANS

- Both DISCOPEN and DISKFILE measure physical access to a file. When deciding when to use DISCOPEN and when to use DISKFILE, consider:
  - DISCOPEN creates a separate counter record for each file open. Opener processes are identified individually. If the same process opens a file more than once, multiple DISCOPEN records are created.
  - DISKFILE measures disk file access per file. DISKFILE creates one counter record for each measured file and does not track individual opener processes.
  - For indirect opens, such as an open of an alternate-key file, DISCOPEN creates a separate record. DISKFILE does not create a separate record for indirect opens.
  - Both DISCOPEN and DISKFILE create a new record if a file closes and reopens during a measurement.
  - DISCOPEN provides a detailed view of disk-file activity. DISCOPEN counters are especially useful for detecting potential problems with the structure of key-sequenced database files.
  - If you do not need a high level of detail, DISKFILE reduces the amount of data you must analyze while still letting you detect potential problems with file structure and contention.

  To measure logical access to a file, use the FILE entity type. For a description, see FILE (page 216).

- DISCOPEN and DISKFILE entities count requests to the storage driver software. These counts do not distinguish between requests that are satisfied out of cache (logical request only) and requests that are satisfied through an I/O transfer (physical I/O).

  In writes to mirrored disks, the request counts are incremented once even though two physical writes occurred.
- The keyword DISKOPEN is interchangeable with DISCOPEN in any MEASCOM command except HELP.

Usage Notes for G-Series DISCOPEN Entities

- The SQL-OPERATION-TIME counter, when summed across all DISCOPEN records for a disk, is not inclusive of all DP2 processing time or of all SQL-related processing time. Many DP2 processing activities cannot be associated with a particular DISCOPEN. Because non-SQL-OPERATION-TIME operations are too small in duration to be measured, the intent of the SQL-OPERATION-TIME counter is to identify SQL operations that are major consumers of DP2 process-busy-time and to associate the requests with a particular opening process.
- In the Guardian file system, a file can be opened through normal Guardian file naming or through the OSS file pathname: /G/*volume*/*subvol*/*filename*.
- For OSS file-system opens of the DISCOPEN record, the OSS file pathname that was used to open the file is displayed. For Guardian file-system opens of the DISCOPEN record, the PATHID information for the record is incomplete, and the request to translate the information cannot occur. In this case, the OSS file pathname displayed is obtained from translation of the Guardian file name and CRVSN value. If more than one path to the file name is defined, all OSS file pathnames are displayed.
- A LIST DISCOPEN * report request or a report request from the Guardian file system displays an OSS file pathname for the record, but any subsequent LIST requests does not return the same OSS file pathname record. This behavior indicates that the file-system access was through the Guardian file system or that the file being listed is a Guardian file name. For more information on the distinctions between the Guardian and OSS file systems and environments, see the *Open System Services User's Guide*.
- In Measure G11 and later PVUs, ANSI SQL names appear in displays.

Example

The report output for the DISCOPEN entity includes the ANSI SQL name for files that have an ANSI SQL name. Here is an example:

```
Disc File Open $DATA.ZSD12345.Z1234567
ANSI SQL Name 'TABLE Catalog_12.Schema_45.Table_56 PARTITION Partition_78'
Device Name $DATA                                  Pool
File Type Key Sequenced                            Local CPU  0
Opener Process: System \HURTS    CPU  0    Pin   285
Format Version:  H01  Data Version:  H01  Subsystem Version:  1
Local System \HURTS   From  12 Jan 2003, 19:35:28   For  46.9 Seconds
------------ Requests ------------------------------------------------------
Lockwait-Time                          Requests                          3 #
Max-Lockwait-Time                      Requests-Blocked
Lock-Timeouts                          Lock-Bounces
------------ Logical I/O ---------------------------------------------------
Cache-Hits                       6 #   Cache-Write-Hits
Block-Splits                           Cache-Write-Cleans
Driver-Input-Calls               6 #   Driver-Output-Calls
DBIO-Input-Calls                       DBIO-Output-Calls
------------ SQL -----------------------------------------------------------
SQL-Operation-Time
```

DISKFILE

This entity measures the I/O operations performed by all opener processes on a specified disk file (physical file access). DISKFILE uses FCB to measure openers and creates one counter record for each measured file. The DISKFILE entity type does not track individual opener processes unless a file closes and reopens during a measurement.

In Measure G09 and later PVUs, the DISKFILE entity supports the use of OSS file pathnames in place of Guardian file names. It displays OSS file pathnames in DISKFILE reports and provides direct mapping between external structured records and the new OSSNAMES structured record output. Although many OSS file pathnames refer to a specific disk file, only one DISKFILE record exists for a file. Using any valid name retrieves the desired record. In reporting the record, all valid names that refer to the record are listed.

In Measure H01 and later PVUs, MEASCOM accepts ANSI SQL names in a DISKFILE entity specification.

Entity Specification Syntax for DISKFILE Entities

To describe a DISKFILE entity or set of DISKFILE entities:

```
DISKFILE entity-spec [,entity-spec] ...
```

DISKFILE

collects information about physical and logical access to a specified file.

entity-spec

is specified as:

```
{ * }
{ [[$device.]subvolume.]filename[:crvsn] }
{ [$device.]#tempfile }
{ "pathname" }
{ '{TABLE|INDEX} [[catalog.]schema.]object[ PARTITION partition]' }
{ 'SCHEMA [catalog.]schema'  }
{ 'CATALOG {catalog|*}'
```

where *catalog*, *schema*, *object* and *partition* are ANSI SQL identifiers. Omitted *catalog* and *schema* fields are resolved by the MEASCOM *catalog* and *schema* environment values.

Wildcard syntax is:

```
TABLE [[catalogC.]schemaS.]tableT - all partitions in tableT
INDEX [[catalogC.]schemaS.]indexI - all index partitions in indexI
SCHEMA [catalogC.]schemaS - all objects in schemaS
CATALOG catalogC - all objects in catalogC
CATALOG * - all objects visible on the node
```

*

measures local and remote access of all disk files.

$*device*

is the name of the volume (device) that contains the file to be measured. To indicate all volumes, use an asterisk (*). The default is the current default volume.

subvolume
: is the name of the subvolume that contains the file to be measured. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume.

filename
: is the name of the file to be measured. To indicate all files except temporary files, use an asterisk (*). Temporary files are specified separately.

:*CRVSN*
: in Measure G10 and later PVUs, is the timestamp, creation version serial number, or filename extension necessary to form a unique filename. Use this option to guarantee filename uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

#*tempfile*
: is the file identification number of a temporary file to be measured. To indicate all temporary files, use an asterisk (*).

"*pname*"
: can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

catalog, schema, object, partition
: are ANSI SQL identifiers in Measure H01 and later PVUs. When omitted, *catalog* and *schema* fields are resolved by the MEASCOM *catalog* and *schema* environment values. If the name space keywords TABLE or INDEX are omitted, TABLE is the assumed default.

DDL Record for DISKFILE Entities (Legacy Style)

This is the Legacy Style DDL record for DISKFILE entities. The fields included in BRIEF reports are in **boldface type**.

The DDL record for G-series DISKFILE entities is identical to the record for D-series DISKFILE entities except that counters are added to support direct bulk I/O and SQL actions. (Existing I/O counters such as DRIVER-INPUT-CALLS also include direct-bulk I/O operations.) In addition, new identifiers have been added for OSS file pathname support.

```
RECORD diskfile.  FILE is "diskfile" ENTRY-SEQUENCED.
                       .
                       .
                       .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                       .
                       .
                       .
* entity identification items:
 02 FCB-number                 type binary 16 unsigned.
 02 file-name.
    03 volume                  type character 8.
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 file-type                  type binary 16 unsigned.
 02 file-code                  type binary 16 unsigned.
* counter value items:
 02 starting-eof               type binary 64.
```

```
 02 ending-eof                  type binary 64.
 02 transient-opens             type binary 32 unsigned.
 02 open-qtime                  type binary 64.
 02 open-qlen-max               type binary 16 unsigned.
 02 driver-input-calls          type binary 32 unsigned.
 02 driver-output-calls         type binary 32 unsigned.
 02 cache-read-hits             type binary 32 unsigned.
 02 cache-write-hits            type binary 32 unsigned.
 02 cache write-cleans          type binary 32 unsigned.
 02 requests                    type binary 32 unsigned.
 02 requests-blocked            type binary 32 unsigned.
 02 lockwait-time               type binary 64.
 02 max-lockwait-time           type binary 64.
 02 lock-timeouts               type binary 32 unsigned.
 02 lock-bounces                type binary 32 unsigned.
 02 block-splits                type binary 32 unsigned.
 02 extent-allocations          type binary 32 unsigned.

* SMS changes:
 02 device-name                 type character 8.
 02 storage-pool                type character 8.
* New counters in G05:
 02 dbio-input-calls            type binary 32 unsigned.
 02 dbio-output-calls           type binary 32 unsigned.
* New counters in G08:
 02 SQL-inserts                 type binary 32 unsigned.
 02 SQL-updates                 type binary 32 unsigned.
 02 SQL-deletes                 type binary 32 unsigned.
 02 ending-rows                 type binary 32 unsigned.
 02 format                      type binary 16 unsigned.
* New identifiers for OSS file pathname support:
 02 file-name-mid.
    03 pathid                   type character 24.
    03 crvsn                    type character  6.
* New counters in G10:
 02 OSS-Block-Writes            type binary 32 unsigned.
 02 OSS-Block-Write-Bytes       type binary 64.
 02 OSS-Cache-Callbacks         type binary 32 unsigned.
 02 OSS-Callback-Writes         type binary 32 unsigned.
end
```

DDL Record for DISKFILE Entities (ZMS Style)

The ZMS style DDL record for DISKFILE entities is supported on Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsdfile-id.
 02 FCB-number                  type binary 16 unsigned.
 02 file-type                   type binary 16 unsigned.
 02 file-code                   type binary 16 unsigned.
 02 format                      type binary 16 unsigned.
 02 file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 device-name                 type character 8.
 02 storage-pool                type character 8.
 02 file-name-MID.
    03 PATHID                   type character 24.
    03 CRVSN                    type character 6.
 02 reserved-1                  type character 2.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsdfile-ctrs.
 02 starting-eof                type binary 64.
 02 ending-eof                  type binary 64.
 02 transient-opens             type binary 64.
 02 open-qtime                  type binary 64.
 02 driver-input-calls          type binary 64.
 02 driver-output-calls         type binary 64.
 02 cache-read-hits             type binary 64.
 02 cache-write-hits            type binary 64.
 02 cache write-cleans          type binary 64.
 02 requests                    type binary 64.
 02 requests-blocked            type binary 64.
 02 lockwait-time               type binary 64.
 02 max-lockwait-time           type binary 64.
 02 lock-timeouts               type binary 64.
 02 lock-bounces                type binary 64.
 02 block-splits                type binary 64.
 02 extent-allocations          type binary 64.
 02 DBIO-input-calls            type binary 64.
 02 DBIO-output-calls           type binary 64.
 02 SQL-inserts                 type binary 64.
 02 SQL-updates                 type binary 64.
 02 SQL-deletes                 type binary 64.
 02 starting-rows               type binary 64.
 02 ending-rows                 type binary 64.
 02 OSS-Block-Writes            type binary 64.
 02 OSS-Block-Write-Bytes       type binary 64.
 02 OSS-Cache-Callbacks         type binary 64.
 02 OSS-Callback-Writes         type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsdfile. FILE is "zmsdfile" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsdfile-ctrs.
 02 id                          type zmsdfile-id.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

FCB-NUMBER

File control block number of the measured file. Internal to the disk process.

FILE-NAME

Name of the measured file. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, FILE-NAME represents the physical file name that was specified when the disk file was opened. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, FILE-NAME represents a location-independent logical file name that was used when the disk file was opened. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

FILE-TYPE

One of these values, which indicates the type of file to be measured:

0	Unstructured
1	Relative
2	Entry-sequenced
3	Key-sequenced

FILE-CODE

Three-digit system file code, such as 100 for object files or 101 for edit files. For a complete list of file codes, see the *File Utility Program (FUP) Reference Manual*.

STARTING-EOF

EOF value of the file, in bytes, at measurement start time. This item is part of the structured DISKFILE record, but MEASCOM does not display it. Instead, MEASCOM uses this counter value as a base to compute the ratio of ENDING-EOF to STARTING-EOF.

Counter type: Snapshot.

ENDING-EOF

EOF value of the file, in bytes, at measurement stop time or measurement interval copy time. The REPORT RATE value does not affect the display of this value in MEASCOM. This item is displayed in its absolute value in bytes, followed by the ratio of ENDING-EOF over STARTING-EOF.

Counter type: Snapshot.

TRANSIENT-OPENS

Number of opens that occur during the measurement period.

Counter type: Incrementing.

OPEN-QTIME

The time for all the opens of this file (including transient opens).

Counter type: Queue.

OPEN-QLEN-MAX

(Legacy Style only) Maximum number of opens on this file.

Counter type: Max queue.

DRIVER-INPUT-CALLS

Number of read operations, including cache hits. The CACHE-READ-HITS counter counts cache hits on read operations.

Counter type: Incrementing.

DRIVER-OUTPUT-CALLS

Number of write operations, including cache hits. The CACHE-WRITE-HITS counter counts cache hits on write operations.

Counter type: Incrementing.

CACHE-READ-HITS

Number of times a read operation found the required block in cache, which saves a disk I/O. The DRIVER-INPUT-CALLS counter counts all read operations, both cache hits and misses.

Counter type: Incrementing.

CACHE-WRITE-HITS

Number of times a write operation found the required block in cache and saved a read I/O (to bring the block into cache). A write I/O can also be saved if the file has the buffered option set to ON. (That is, if multiple writes can occur to a dirty block before it is physically written.)

Counter type: Incrementing.

CACHE-WRITE-CLEANS

Number of times a write operation found the required block clean in cache, which saves a read operation to bring the block into cache.

Counter type: Incrementing.

REQUESTS

Number of requests for this file.

Counter type: Incrementing.

REQUESTS-BLOCKED

Number of times an I/O operation had to wait because the requested file or record was locked. LOCKWAITS is another term for this counter.

Counter type: Incrementing.

LOCKWAIT-TIME

The time spent waiting for locks.

The REQUESTS-BLOCKED counter counts the number of blocked requests—that is, the number of times requests are queued to locks. When the REPORT RATE value is on, this counter returns the average wait time (LOCKWAIT-TIME divided by REQUESTS-BLOCKED).

Counter type: Busy.

MAX-LOCKWAIT-TIME

Maximum wait time for each lock. Use this number as a guide to set the timeout value in the application.

Counter type: Max value.

LOCK-TIMEOUTS

Number of timeouts on locks (error 40). When a timeout value for a request expires, the file system (Enscribe and SQL/MP) sends a cancel request to the disk process. If the disk process finds the request still waiting for lock to be granted, the request is dequeued from the lockwait queue, and this counter is advanced.

Counter type: Incrementing.

LOCK-BOUNCES

Number of bounced locks (FELOCKED = error 73) returned to the file system. For SQL tables, a bounced lock is a result of the CONTROL TABLE command with the RETURN-IF-LOCKED control option. For Enscribe files, a bounced lock is a result of the SETMODE 4 function with Param.<15> = 1 (alternate lock mode).

Counter type: Incrementing.

BLOCK-SPLITS

Number of block splits during writes to the file.

Counter type: Incrementing.

EXTENT-ALLOCATIONS

Number of times a file extent is allocated.

Counter type: Incrementing.

DEVICE-NAME

Disk device on which the measured file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

STORAGE-POOL

Name of the SMF storage pool to which the measured disk is assigned (process name of the storage pool process).

DBIO-INPUT-CALLS

Number of direct bulk I/O read operations to a requesting processor. This counter is a subset of DRIVER-INPUT-CALLS.

DBIO-OUTPUT-CALLS

Number of direct bulk I/O write operations from a requesting processor. This counter is a subset of DRIVER-OUTPUT-CALLS.

SQL-INSERTS

The number of row insert operations performed on the SQL table.

Counter type: Incrementing.

SQL-UPDATES

The number of row update operations performed on the SQL table.

Counter type: Incrementing.

SQL-DELETES

The number of row delete operations performed on the SQL table.

Counter type: Incrementing.

STARTING-ROWS

Reserved for future use.

SQL-ENDING-ROWS

The number of rows in the SQL table at the end of a measurement interval. This counter is valid only if the table had zero rows at some time since the file open.

Counter type: Snapshot.

FILE-NAME-MID

FILE-NAME-MID has two subfields: PATHID and CRVSN.

- PATHID is an internal format representation of an OSS file or SQL/MX file. For files other than OSS or SQL/MX files, the field contains zeros.
- CRVSN is a creation version serial number that identifies a unique instance of an OSS or SQL/MX file. For other files, the field contains zeros.

OSS-BLOCK-WRITES

The number of block write operations issued by DP2 for the OSS File System.

Counter type: Incrementing.

OSS-BLOCK-WRITE-BYTES

The number of bytes written by DP2 as a result of OSS-BLOCK-WRITES.

Counter type: Accumulating.

OSS-CACHE-CALLBACKS

The number of cache callback requests sent by DP2 to the OSS file manager.

Counter type: Incrementing.

OSS-CALLBACK-WRITES

The number of OSS-BLOCK-WRITES received by DP2 as a result of OSS-CACHE-CALLBACKS.

Counter type: Incrementing.

Usage Notes for All DISKFILE Entities

- Both DISCOPEN and DISKFILE measure physical access to a file. When you decide when to use DISCOPEN and when to use DISKFILE, consider:
  - DISCOPEN creates a separate counter record for each file open. Opener processes are identified individually. If the same process opens a file more than once, multiple DISCOPEN records are created.
  - DISKFILE measures disk file access collectively for a file as a whole. DISKFILE creates one counter record per measured file and does not track individual opener processes.
  - For indirect opens, such as an open of an alternate-key file, DISCOPEN creates a separate record. DISKFILE does not create a separate record for indirect opens.
  - Both DISCOPEN and DISKFILE create a new record if a file closes and reopens during a measurement.
  - DISCOPEN provides a detailed view of disk file activity. DISCOPEN counters are especially useful for detecting potential problems with the structure of key-sequenced database files.
  - If you do not need a high level of detail, DISKFILE reduces the amount of data you must analyze while still letting you detect potential problems with file structure and contention.

  To measure logical access to a file, use the FILE entity type. For a description, see FILE (page 216).
- DISCOPEN and DISKFILE entities count requests to the storage driver software. These counts do not distinguish between requests that are satisfied out of cache (logical request only) and requests that are satisfied through an I/O transfer (physical I/O).

  In writes to mirrored disks, the request counts are incremented once even though two physical writes occurred.

Usage Notes for G-Series DISKFILE Entities

- The counters SQL-INSERTS, SQL-UPDATES, and SQL-DELETES are available starting with DP2 (T9053G08) and the G06.11 RVU. The counter SQL-ENDING-ROWS is available with DP2 (T9053G09) and the G06.12 RVU.
- The SQL-ENDING ROWS counter is of interest for SQL tables used to represent a queue. Following the initial open of a file, this value is not displayed and is reported in the external records as a -1 until the number of rows in the table has been zero at some time. Once the number of rows is equal to zero (for example, an empty queue), the counter tracks the number of rows in the file at the end of each interval.
- In Measure G08 and later PVUs, the DISKFILE entity report identifies whether the disk file is file format 1 or 2. A format 2 file is used when:
  - The maximum size of an unpartitioned file (or of any partition of a partitioned file) exceeds 2 GB - 1 MB.
  - A non-key-sequenced partitioned file has a maximum size equal to or greater than 4 GB.
  - A file's extent size exceeds 65535 pages.
  - A format 2 file is explicitly requested at creation.
- In Measure G09 and later PVUs (support of OSS file pathnames):
  - The directory information in the OSS PATHID is ignored for DISKFILE usage of OSS file pathnames.
  - Only one DISKFILE record exists for a file despite the number of accesses made.
  - Any valid pathname that leads to the file can be used.
  - The use of directory names to indicate a set of DISKFILE records is not supported.
  - If you must measure transient DISKFILE records for OSS files, specify them using the Guardian file name wild-card conventions; for example:
    - $*.*.*
    - $*device*.*.*
    - *.ZYQ*nnnnn*.
  - OSS PATHID and CRVSN information is not included in records returned by MEASREADACTIVE. To obtain OSS file pathnames that correspond to a particular DISKFILE record, use MEASLISTPNAME. This translation produces a warning that a CRVSN was not specified. However, you can ignore this warning because the translation is for a value from an active record.
- In Measure G11 and later PVUs, ANSI SQL names appear in displays.

Example

The report output for the DISKFILE entity includes the ANSI SQL name for files that have an ANSI SQL name.

```
Diskfile $DATA.ZSD12345.Z1234567                File Format x
ANSI SQL Name 'TABLE Catalog_12.Schema_45.Table_56 PARTITION Partition_78'
Device Name $DATA                                        Pool
Local CPU 0           File Type  Key Sequenced       File Code 550
Format Version:  H01  Data Version:  H01  Subsystem Version:  1
Local System \HURTS   From  12 Jan 2003, 19:35:28   For  46.9 Seconds
------------ File ------------------------------------------------------
Open-Qtime                  72.11 ms   Transient-Opens
Starting-EOF                           Ending-EOF
Extent-Allocations                     File-Growth-Ratio
------------ Requests --------------------------------------------------
Lockwait-Time                          Requests                        3 #
Max-Lockwait-Time                      Requests-Blocked
Lock-Timeouts                          Lock-Bounces
------------ Logical I/O -----------------------------------------------
```

```
Cache-Read-Hits                  2 #    Cache-Write-Hits
Block-Splits                            Cache-Write-Cleans
Driver-Input-Calls               2 #    Driver-Output-Calls
DBIO-Input-Calls                        DBIO-Output-Calls
------------ SQL ------------------------------------------------------------
SQL-Inserts
SQL-Updates                             SQL-Deletes
SQL-Starting-Rows                       SQL-Ending-Rows
------------ Open System Services -------------------------------------------
OSS-Cache-Callbacks                     OSS-Callback-Writes
OSS-Block-Writes                        OSS-Block-Write-Bytes
```

FILE

This entity measures the I/O operations performed by a user process on an explicitly opened file (logical file access).

In Measure G09 and later PVUs, the FILE entity type handles OSS file pathnames.

In Measure G11 and later PVUs, the FILE entity displays ANSI SQL names in FILE reports. In Measure H01 and later PVUs, MEASCOM accepts ANSI SQL names in FILE entity specifications.

Measure and OSS File Opens

The FILE entity includes a number of FILE relationships available to OSS. OSS FILE open types include:

OSS File Type	Description
OSS regular files	OSS disk files with names in the OSS hierarchical name space
OSS pipes	IPC mechanism for communication between processes
OSS FIFO	Named pipes, with names available in the OSS name space
OSS sockets	IPC mechanism using OSS AF_INET or OSS AF_UNIX sockets

The FILE entity does not record OSS opens of OSS directories, TTY devices, and /dev/null. For details on the aggregate counters used for these objects, see the PROCESS entity on page 272. OSS file instrumentation is generally at the system call level rather than the library call level. Existing Guardian TCP/IP sockets instrumentation is unchanged. Process file opens are recorded, but no socket level information is available.

The two significant differences between the support of OSS file opens in the FILE entity and that of Guardian file opens are:

- The timing of counter record creation
- Naming conventions used to identify OSS file open objects

Record Creation

OSS files are often opened within a parent process environment but accessed only in the context of a child process that inherits the open. Similarly, a child process might not actually use all file opens inherited from a parent process. To minimize the system impact of tracking OSS file opens in Measure, counter records for OSS file opens are initiated on first significant use of the file, not on the open of the file. These OSS APIs qualify as significant use:

accept()	open(O_TRUNC)	send()
fcntl(F_GETLK,F_SETLK,F_SETLKW)	read()	sendmsg()
fsync()	recv()	sendto()
ftruncate()	recvfrom()	write()
lseek()	recvmsg()	

For Guardian file opens, you commonly see file records that are not used by a process during a measurement. For OSS file opens, such opens do not produce a data record.

OSS Naming Conventions

A Measure FILE record identifies a file open object by its Guardian format name (*$vol.subvol.fname*). OSS file open objects might also have an OSS file pathname defined, such as /work/files/fname. An additional identifier for OSS AF_INET and AF_INET6 sockets, IP-ADDR, is an IP address-port combination specified in a quoted string with the port space-separated from the address (for example, "123.45.67.8 90" or "3FFE:1200:215:1 64").

Except for OSS regular files, OSS file open types do not have a valid Guardian file name available for use. For Measure use only, the OSS file system creates a Guardian file name for these file opens for configuration, identification, and selection. These names are only for Measure record selection and display. They cannot be used as targets of a FILE_OPEN_ call or otherwise applied in commands that use a file name as a parameter. To form the name, the OSS file system uses the device name of the process providing the resource followed by unique identifiers (*qual1*, *qual2*):

- The TCP/IP process name (user-defined) for AF_INET and AF_INET6 sockets
- The pipe server name ($ZPP*nn*) for pipes and ($ZPPNN) for FIFOs
- The local server ($ZPLS) for AF_UNIX sockets (pre-AF_UNIX R2)
- The local server ($ZLS*nn*) for unbound AF_UNIX sockets and ($ZLSNN) for bound AF_UNIX sockets (AF_UNIX R2)

In all cases, a CRVSN/timestamp is appended to the file name to form a unique file name modifier, represented as *cccccc* in the following list.

Depending on OSS file open type, the following file identifiers are present in the Measure file record. If OSS journaling is enabled for a measurement, only OSS file pathnames for regular files, FIFOs, and bound AF_UNIX sockets are journaled.

The pipe server and the local server (AF_UNIX R2) are really process groups, with one process per CPU. The last two bytes of the process name (*nn*) is the CPU number where the individual process resides. For example, the pipe server on a four-CPU system is a process group consisting of processes, $ZPP00, $ZPP01, $ZPP02 and $ZPP03. To create the illusion of a single server process for FIFOs and bound sockets, the File identifier that is displayed in the LIST FILE display

uses the process name, $ZPPNN, for the pipe server, and $ZLSNN for the local server for bound sockets. The actual process name (the one with the *nn*) is still shown as the Device Name in the LIST FILE display. Below is a sample LIST FILE display showing a bound socket:

```
> meascom;add ossfidfb;list file * (osssocket)
File Open $ZLSNN.Z00000.Z0001SQG:106899423347         Open Type OSSUNIXSTREAM
OSSPath: "/tmp/kenm/mysock"
Device Name $ZLS01            Device Type  0  (Process)      Subdevice Type  0
Opener 1,576    ($Z01H)       File Num 2       OSSPID:  64356356
Program  $OSS2.ZYQ00000.Z000161Y:929432249
OSSPath: "/tmp/kenm/server.exe"
Opener Device Name $OSS2
Format Version:  H04  Data Version:  H04  Subsystem Version:  0
Local System \MEASYOS From  14 Nov 2008, 14:50:33   For  19.7 Seconds
```

OSS File Open Type	File Identifier	
Regular files	$*VOL*.ZYQ*nnnnn*.Ziiiiiii:*cccccc*	OSSPath
FIFOs	$ZPPNN.Z*nnnnn*.Ziiiiiii:*cccccc*	OSSPath
Pipes	$ZPP*nn.qual1.qual2*:*cccccc*	No OSSPath or IP-Addr
AF_INET datagram sockets (unbound, unconnected or connected)	$ZTC*nn.qual1.qual2*:*cccccc*	No OSSPath or IP-Addr
AF_INET datagram sockets (bound)	$ZTC*nn.qual1.qual2*:*cccccc*	IP-Addr
AF_UNIX datagram sockets (unbound, unconnected, or connected) (pre-AF_UNIX R2)	$ZPLS.*qual1.qual2*:*cccccc*	No OSSPath or IP-Addr
AF_UNIX datagram sockets (bound) (pre-AF_UNIX R2)	$ZPLS.Z*nnnnn*.Z*iiiiiii*:*cccccc*	OSSPath
AF_UNIX datagram sockets (unbound, unconnected, or connected) (AF_UNIX R2)	$ZLS*nn.qual1.qual2:cccccc*	No OSSPath or IP-Addr
AF_UNIX datagram sockets (bound) (AF_UNIX R2)	$ZLSNN.Z*nnnnn.Ziiiiiii:cccccc*	OSSPath
AF_INET streams sockets (unbound or unconnected)	$ZTC*nn.qual1.qual2*:*cccccc*	No OSSPath or IP-Addr
AF_INET streams sockets (bound or connected)	$ZTC*nn.qual1.qual2*:*cccccc*	IP-Addr
AF_UNIX streams sockets (unbound or unconnected) (pre-AF_UNIX R2)	$ZPLS.*qual1.qual2*:*cccccc*	No OSSPath or IP-Addr
AF_UNIX streams sockets (bound or connected) (pre-AF_UNIX R2)	$ZPLS.Z*nnnnn*.Z*iiiiiii*:*cccccc*	OSSPath of named socket connection
AF_UNIX streams sockets (unbound or unconnected) (AF_UNIX R2)	$ZLS*nn.qual1.qual2:cccccc*	No OSSPath or IP-Addr
AF_UNIX streams sockets (bound or connected) (AF_UNIX R2)	$ZLSNN.Z*nnnnn.Ziiiiiii:cccccc*	OSSPath of named socket connection

IP-ADDR

Determining the significance of the IP-Addr value depends on the application. In general, a bound or connected stream socket in the server has its own address in IP-Addr. A bound or connected socket in the client process has the address of the server.

You cannot use IP-ADDR as a criterion for a measurement ADD FILE command, but you can select reports for display using the address for selection, as in LIST FILE *, IF IP-ADDR = "address port". Limited wild-card use is supported; for example:

```
{ "address port"  }
{ "address *"  }
{ "* port"   }
```

For more information about IP field values, see IP (page 230). Examples of use are in Command Examples: OSS File Opens (page 232).

ANSI SQL Naming Conventions

In Measure G11 and later PVUs, the FILE entity can provide ANSI SQL names in place of Guardian file names in its output. For complete details on SQL naming, see the *SQL/MX Reference Manual*.

Entity Specification Syntax

In Measure G09 and later PVUs, the FILE entity supports the use of OSS file pathnames in place of Guardian file names. It also displays OSS file pathnames in FILE reports and provides direct mapping between external structured records and OSSNAMES structured record output. OSS file pathnames might specify the name of the file being opened or specify the program file name of the process opening the file.

The G09 FILE entity specification syntax supports using OSS file pathnames for file name or opener-program-filename.

To describe a FILE entity:

```
FILE entity-spec [,entity-spec] ...
```

FILE

For D-series RVUs, collects information about operations on one or more logical files. The file can be local or remote. However, only local processes that access the file are measured.

For G-series RVUs, measures the I/O operations performed by a process on an explicitly open file (logical file access). The file can be local or remote. However, only local processes that access the file are measured.

entity-spec

is specified as:

```
filename [(opener)] [(file^open^type^list)]
```

where *filename* is one of the following:

```
{ * }
{ [[$device.]subvolume.]filename }
{ [$device.]#tempfile }
{ [$device.]#subdevice }
{ "pathname" }
{ '{TABLE|INDEX} [[catalog.]schema.]object' }
{ 'SCHEMA [[catalog.]schema'  }
{ 'CATALOG [{catalog|*}'  }
```

where *catalog*, *schema* and *object* are ANSI SQL identifiers. Omitted *catalog* and *schema* fields are resolved by the MEASCOM *catalog* and *schema* environment values.

Wildcard syntax is:

```
SCHEMA [catalogC.]schemaS - all objects in schemaS
CATALOG catalogC - all objects in catalogC
CATALOG * - all objects visible on the node
```

*

measures all files, temporary files, and subdevices, both local and remote, accessed by local processes.

file

is the name of the opened file to be measured, specified as:

```
{[[[\system.]$device.]subvolume.]filename[:CRVSN]        }
{[[\system.]$device.]#tempfile                           }
{[[\system.]$device.]#subdevice                          }
```

\system
: is the name of the system that contains the file to be measured. To indicate all systems, use an asterisk (*). The default is the current default system.

$device
: is a name of the volume (device) that contains the file to be measured. To indicate all devices, use an asterisk (*). The default is the current default device.

subvolume
: is the name of the subvolume that contains the file to be measured. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume).

filename
: is the name of the file to be measured. To indicate all files (except temporary files, which are specified separately), use an asterisk (*).

:CRVSN
: in Measure G10 and later PVUs, is the timestamp, creation version serial number, or filename extension necessary to form a unique filename. Use this option to guarantee filename uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

#tempfile
: is the file identification number of a temporary file to be measured. To indicate all temporary files, use an asterisk (*).

#subdevice
: is the name of a subdevice to be measured. To indicate all subdevices, use an asterisk (*).

opener
: is the process whose I/O operations on the specified file are measured. Specify *opener* as:

```
{ cpu, pin [ , filenum ]                        }
{ $pname [ , filenum ]                          }
{[[ $device.]subvolume.]filename [ , filenum ]  }
{ "pname" [ , filenum ]                         }
```

cpu
: is the number of the CPU on which the process is running.

pin
: is the process identification number of the process.

filenum
: is the file number of the process.

"pname"
: for D-series RVUs, is the process name or program file name of the process.

  for G-series and later RVUs, is the OSS file pathname of the program file name of the process.

"*pname*"
: can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

file^open^type

in Measure G10 and later, selects one, all, or several file open types for measurement. By default, only Guardian file opens are measured. Valid options include these literals:

ALLFILES	Selects all file open types for measurement (default for the ADD FILE and LIST commands); includes Guardian, OSS, and ANSI SQL files
GUARDIAN	Selects all Enscribe, process, and SQL file opens (current default for the ADD command)
ENSCRIBE	Selects Enscribe opens of unstructured, relative, entry-sequenced, and key-sequenced files
UNSTRUCT	Selects Enscribe opens of unstructured files or OSS regular files
RELFILE	Selects Enscribe opens of relative files
KEYFILE	Selects Enscribe opens of entry-sequenced files
SQLFILE	Selects SQL file opens
PROCFILE	Selects Enscribe opens of process and other nondisk files
OSS	Specifies all OSS opens are included
OSSDISK	Selects OSS opens of regular files
OSSFIFO	Selects OSS opens of named pipes
OSSPIPE	Selects OSS opens of unnamed pipes
OSSSOCKET	Selects OSS opens of all sockets
OSSINETSOCKET	Selects OSS opens of AF_INET sockets
OSSINETDGRAM	Selects OSS opens of AF_INET datagram sockets
OSSINETSTREAM	Selects OSS opens of AF_INET stream sockets
OSSUNIXSOCKET	Selects OSS opens of AF_UNIX sockets
OSSUNIXDGRAM	Selects OSS opens of AF_UNIX datagram sockets
OSSUNIXSTREAM	Selects OSS opens of AF_UNIX stream sockets

catalog, schema, object

in Measure H01 and later, are ANSI SQL identifiers. When omitted, *catalog* and *schema* fields are resolved by the MEASCOM *catalog* and *schema* environment values. If the name space keywords TABLE or INDEX are omitted, TABLE is the assumed default.

DDL Record for FILE Entities (Legacy Style)

This is the Legacy Style DDL record for FILE entities. The fields included in BRIEF reports are in **boldface type**. This record will not change after the G10 Measure PVU.

The DDL record for G-series FILE entities is identical to the record for D-series FILE entities except:

- Counters are added to measure direct bulk I/O operations.
- Longer byte-count fields are provided to reduce the possibility of field overflow. Each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see SYSTEM (page 128).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.
- Identifiers are added for OSS file pathnames support.

```
RECORD file. FILE is "file" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
 02 opener-cpu                  type binary 16 unsigned.

* entity identification items:
 02 opener-pin                  type binary 16 unsigned.
 02 file-number                 type binary 16 unsigned.
 02 file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 file-system-name            type character 8.
 02 file-type                   type binary 16 unsigned.
 02 device-type                 type binary 16 unsigned.

* counter value items:
 02 file-busy-time              type binary 64.
 02 reads                       type binary 32 unsigned.
 02 writes                      type binary 32 unsigned.
 02 updates-or-replies          type binary 32 unsigned.
 02 deletes-or-writereads       type binary 32 unsigned.
 02 info-calls                  type binary 32 unsigned.
 02 records-used                type binary 32 unsigned.
 02 records-accessed            type binary 32 unsigned.
 02 disc-reads                  type binary 32 unsigned.
 02 messages                    type binary 32 unsigned.
 02 message-bytes               type binary 32 unsigned.
 02 lock-waits                  type binary 32 unsigned.
 02 timeouts-or-cancels         type binary 32 unsigned.
 02 escalations                 type binary 32 unsigned.

* new entity identification items for D10:
 02 opener-processname          type character 8.
 02 opener-program-filename.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.

* SMS changes:
 02 device-name                 type character 8.
 02 opener-device-name          type character 8.
* New F40 counter value items:
 02 message-bytes-f              type binary 64.

* New counters in G05:
 02 dbio-reads                  type binary 32 unsigned.
 02 dbio-writes                 type binary 32 unsigned.
```

```
 02 dbio-read-bytes            type binary 64.
 02 dbio-write-bytes           type binary 64.

* New identifiers for OSS file pathname support:
 02 file-name-mid.
    03 pathid                  type character 24.
    03 crvsn                   type character  6.
 02 opener-osspid              type binary 32 unsigned.
 02 opener-program-fname-mid.
    03 pathid                  type character 24.
    03 crvsn                   type character  6.

* New identifier for G10:
 02 File-open-type             type binary 16 unsigned.
 02 IP.
    03 family                  type binary 16 unsigned.
    03 port                    type binary 16 unsigned.
    03 IP-addr                 type character 16.

* New counters for G10:
 02 read-bytes                 type binary 64.
 02 write-bytes                type binary 64.
 02 OSS-cache-reads            type binary 64.
 02 OSS-cache-writes           type binary 64.
 02 OSS-cache-read-bytes       type binary 64.
 02 OSS-cache-write-bytes      type binary 64.
 02 OSS-block-reads            type binary 32 unsigned.
 02 OSS-block-read-bytes       type binary 64.
 02 OSS-flow-controls          type binary 32 unsigned.
 02 misc-calls                 type binary 32 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for FILE Entities (ZMS Style)

The ZMS style DDL record for FILE entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsfile-id.
 02 opener-pin                 type binary 16 unsigned.
 02 file-number                type binary 16 unsigned.
 02 file-type                  type binary 16 unsigned.
 02 device-type                type binary 16 unsigned.
 02 file-name.
    03 volume                  type character 8.
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 file-system-name           type character 8.
 02 opener-processname         type character 8.
 02 opener-program-filename.
    03 volume                  type character 8.
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 device-name                type character 8.
 02 opener-device-name         type character 8.
 02 file-name-MID.
    03 PATHID                  type character 24.
    03 CRVSN                   type character 6.
 02 reserved-1                 type character 6.
 02 opener-osspid              type binary 32 unsigned.
```

```
 02 opener-program-fname-MID.
    03 PATHID                    type character 24.
    03 CRVSN                     type character 6.
 02 file-open-type               type binary 16 unsigned.
 02 IP.
    03 family                    type binary 16 unsigned.
    03 port                      type binary 16 unsigned.
    03 IP-addr                   type character 16.
 02 reserved-2                   type character 4.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsfile-ctrs.
 02 file-busy-time               type binary 64.
 02 reads                        type binary 64.
 02 writes                       type binary 64.
 02 updates-or-replies           type binary 64.
 02 deletes-or-writereads        type binary 64.
 02 info-calls                   type binary 64.
 02 records-used                 type binary 64.
 02 records-accessed             type binary 64.
 02 disc-reads                   type binary 64.
 02 messages                     type binary 64.
 02 message-bytes                type binary 64.
 02 lock-waits                   type binary 64.
 02 timeouts-or-cancels          type binary 64.
 02 escalations                  type binary 64.
 02 DBIO-reads                   type binary 64.
 02 DBIO-writes                  type binary 64.
 02 DBIO-read-bytes              type binary 64.
 02 DBIO-write-bytes             type binary 64.
 02 read-bytes                   type binary 64.
 02 write-bytes                  type binary 64.
 02 OSS-cache-reads              type binary 64.
 02 OSS-cache-writes             type binary 64.
 02 OSS-cache-read-bytes         type binary 64.
 02 OSS-cache-write-bytes        type binary 64.
 02 OSS-block-reads              type binary 64.
 02 OSS-block-read-bytes         type binary 64.
 02 OSS-flow-controls            type binary 64.
 02 misc-calls                   type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsfile. FILE is "zmsfile" ENTRY-SEQUENCED.
 02 hdr                          type zmsheader.
 02 ctr                          type zmsfile-ctrs.
 02 id                           type zmsfile-id.
end
```

OPENER-CPU

(Legacy Style only) Number of the CPU on which the process that opened the file is executing.

OPENER-PIN

Process identification number of the process that opened the file.

FILE-NUMBER

Number of the file.

FILE-NAME

Name of the measured file. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, FILE-NAME represents the physical file name that was specified when the file was opened. The VOLUME subfield gives the device name of the physical volume on which the primary disk file is located. If a primary disk file has associated partitions or alternate key files, all activity is recorded under the primary file name.

For SMF files, FILE-NAME represents a location-independent logical file name that was used when the file was opened. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

FILE-SYSTEM-NAME

Name of the system from the complete file specification.

FILE-TYPE

File type, from the access control block:

1	Relative file
2	Entry-sequenced file
3	Key-sequenced file
0	Any other type of file

DEVICE-TYPE

One of these values, which indicates the type of device associated with the file:

0	Process
1	Operator
2	$RECEIVE
3	Disk
4	Tape drive
5	Printer
6	Terminal
7	Communication line
8	SCSI

FILE-BUSY-TIME

For waited I/O, the FILE-BUSY-TIME counter is the elapsed time spent executing waited I/O requests. OSS file opens count file busy time for waited requests after the first significant file operation on the file. This counter is started by the file system when the message for the I/O request (for example, READ or WRITE) is sent to the server. The counter is stopped after the reply is received and the process runs again to process the reply. Thus, the counter includes execution time of the requester, execution time of the server (for example, DP2 disk process), any physical I/O time (for example, disk read), and possibly the execution time of higher priority processes in both CPUs, and so forth. Requests sent during the file open and close and requests sent to the OSS name server are not counted in the FILE-BUSY-TIME counter, but are counted in the PROCESS entity.

For nowait I/O, the FILE-BUSY-TIME counter is the execution time to initiate nowait I/O requests to the server and any elapsed time the process actually spends waiting for nowait requests to complete in AWAITIO[X].

Counter type: Busy.

READS

For D-series RVUs, number of calls to file-system read operations such as READX and OSS file system read APIs such as read() or recv(). For OSS, the total includes OSS-CACHE-READS. For the Enscribe file system, the total includes non-DBIO and DBIO operations. This counter includes both user and system calls to these procedures.

For G-series RVUs, number of calls to the READ, READLOCK, READUPDATE, and READUPDATELOCK procedures. This counter includes both user and system calls to these procedures. For G05 and later RVUs, this counter includes direct bulk I/O read operations.

Counter type: Incrementing.

WRITES

For D-series RVUs, number of calls to file-system write operations such as WRITEX, and OSS File System write APIs such as write() or send(). For OSS, the total includes OSS-CACHE-WRITES. For the Enscribe file system, the total includes non-DBIO and DBIO operations. This counter includes both user and system calls to WRITE.

For G-series RVUs, number of calls to the WRITE procedure. This counter includes both user and system calls to WRITE. For G05 and later RVUs, this counter includes direct bulk I/O write operations.

Counter type: Incrementing.

UPDATES-OR-REPLIES

Number of calls to the WRITEUPDATE or WRITEUPDATEUNLOCK procedure with a buffer length greater than zero (update operations) or the number of calls to the REPLY procedure (reply operations). This counter includes both user and system calls to these procedures.

Interpretation of this counter depends on the type of file being measured. Update operations are performed on disk and tape files. Reply operations are performed on terminals or processes.

Counter type: Incrementing.

DELETES-OR-WRITEREADS

Number of calls to the WRITEUPDATE or WRITEUPDATEUNLOCK procedure with a buffer length of 0 (delete operations) or the number of calls to the WRITEREAD procedure (writeread operations). This counter includes both user and system calls to these procedures.

Interpretation of this counter depends on the type of file being measured. Delete operations are performed on disk and tape files. Writeread operations are performed on terminals or processes.

Counter type: Incrementing.

INFO-CALLS

Number of calls to these procedures that resulted in a message being dispatched to a process (IOP or other) to obtain the information requested:

FILEINFO	FILE_GETINFOBYNAME_	DEVICEINFO
FILERECINFO	FILE_GETINFOLISTBYNAME_	DEVICEINFO2
FILE_GETINFO_	FILEGETINFO_LIST	

For OSS, the number of fstat(), fstatvfs(), fpathconf(), getsockname(), getpeername(), getsockopt(), and sockatmark() API calls for this file.

This counter includes both user and system calls to these procedures.

Counter type: Incrementing.

RECORDS-USED

Number of records returned to the SQL executor on reads, inserts, writes, updates, and deletes. The value of this counter is zero for Enscribe files.

Counter type: Incrementing.

RECORDS-ACCESSED

Number of records read by the disk process or file system to perform the operations. Number of records accessed is greater than or equal to the number of records used. The ratio of RECORDS-USED to RECORDS-ACCESSED indicates the selectivity of the statement. The value of this counter is zero for Enscribe files.

Counter type: Incrementing.

DISC-READS

Number of physical disk reads performed.

Counter type: Incrementing.

MESSAGES

Number of messages sent for this open, including the OPEN messages. Because file labels might exceed the 30-KB file-message limit, multiple OPEN messages might be sent.

For OSS file opens, MESSAGES is the total number of messages sent or received from server processes and the OSS Name Server for this file. It does not include the OPEN message, which is tracked in the PROCESS entity. For OSS sockets, MESSAGES and MESSAGE-BYTES reflect OSS Name Server activity only.

Counter type: Incrementing.

MESSAGE-BYTES

Number of message bytes sent and received for this file open.

For OSS, counts the total number of message data bytes sent or received for MESSAGES from server processes and the OSS Name Server for this file.

This counter does not include bytes transferred by direct bulk I/O. The DBIO-READ-BYTES and DBIO-WRITE-BYTES counters measure direct bulk I/O transfers.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the MESSAGE-BYTES-F field is a 64-bit version of MESSAGE-BYTES.

Counter type: Accumulating.

LOCK-WAITS

Number of times a call waited for a lock request. For OSS, counts whether a fcntl (,F_SETLKW) lock request waited for another lock.

Counter type: Incrementing.

TIMEOUTS-OR-CANCELS

Number of timeouts or cancels issued against this OPEN.

Counter type: Incrementing.

ESCALATIONS

Number of times a lock escalated to a file-level lock.

Counter type: Incrementing.

OPENER-PROCESSNAME

Name of the opener process.

OPENER-PROGRAM-FILENAME

Name of the opener program file. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, OPENER-PROGRAM-FILENAME represents the physical program file name of the file opener. The VOLUME subfield gives the device name of the physical volume on which the program file is located.

For SMF files, OPENER-PROGRAM-FILENAME represents a location-independent logical file name of the file opener. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

DEVICE-NAME

Disk device on which the measured file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

OPENER-DEVICE-NAME

Disk device on which the opener program file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the OPENER-PROGRAM-FILENAME VOLUME subfield.

MESSAGE-BYTES-F

(G-series, Legacy Style only) Same as MESSAGE-BYTES but accommodates larger values (64 bits rather than 32 bits).

DBIO-READS

Number of direct bulk I/O read operations to the measured file. This counter is a subset of READS.

Counter type: Incrementing.

DBIO-WRITES

Number of direct bulk I/O write operations to the measured file. This counter is a subset of WRITES.

Counter type: Incrementing.

DBIO-READ-BYTES

Number of bytes transferred by direct bulk I/O read operations.

Counter type: Incrementing.

DBIO-WRITE-BYTES

Number of bytes transferred by direct bulk I/O write operations.

Counter type: Incrementing.

FILE-NAME-MID

FILE-NAME-MID has two subfields:

- PATHID is an internal format representation of an OSS file or SQL/MX file. For other files, the field contains zeros.
- CRVSN is a creation version serial number that identifies a unique instance of an OSS or SQL/MX file. For other files, the field contains zeros.

OPENER-OSSPID

If the opener process runs in the OSS environment, this field contains the OSS Process ID. If this process is not an OSS process, this field contains zeros.

OPENER-PROGRAM-FNAME-MID

OPENER-PROGRAM-FNAME-MID has two subfields: PATHID and CRVSN. PATHID is an internal format representation of an OSS file name. If these files are not OSS files, this field contains zeros. CRVSN is the creation version serial number that identifies a unique instance of an OSS file. For other files, this field contains zeros.

FILE-OPEN-TYPE

Identifier for the file open type added for measurement or selected for list display reporting. Possible values are:

ALLFILES	%B0000000111111111
GUARDIAN	%B1111111111111111
ENSCRIBE	%B1111111011111111
UNSTRUCT	%B1111111011100001
RELFILE	%B1111111011100010
ENTRYFILE	%B1111111011100100
KEYFILE	%B1111111011101000
SQLFILE	%B1111111111100000
PROCFILE	%B1111111011110000
OSS	%B0000000011100000
OSSDISK	%B0111111011100000
OSSFIFO	%B1101111011100000
OSSPIPE	%B1011111011100000
OSSSOCKET	%B1110000011100000
OSSUNIXSOCKET	%B1110011011100000
OSSUNIXDGRAM	%B1110111011100000
OSSUNIXSTREAM	%B1111011011100000
OSSINETSOCKET	%B1111100011100000
OSSINETDGRAM	%B1111101011100000
OSSINETSTREAM	%B1111110011100000

IP

An IP address value consisting of three subfields: FAMILY, PORT, and IP address. Determining the significance of the IP-Addr value depends on the application. In general, a bound or connected stream socket in the server has its own address in IP-ADDR. A bound or connected socket in the client process has the address of the server.

The FAMILY field specifies whether you are using IPv4 or IPv6 addressing. The possible values of this field are 2 for IPv4 or 26 for IPv6.

The PORT field contains the port number.

The IP-ADDR field is interpreted differently for IPv4 and IPv6. If the FAMILY field specifies IPv4 addressing--if the value of FAMILY is 2--then the first four bytes of the IP-addr field contain hexadecimal values for numerals, representing the four parts of the dotted decimal form of the IP address. If the FAMILY field specifies IPv6 addressing--if the value of FAMILY is 26--then all 16 bytes of the IP-addr field are used to accommodate the hexadecimal values corresponding to numerals in the IP address.

In ADD FILE commands, you can specify an IP address in a quoted string with the port space-separated from the address (for example, "123.45.67.8 90" or "3FFE:1200:215:1 64").

READ-BYTES

Number of bytes returned to a user application data buffer. Available for all file open types.

Counter type: Accumulating.

WRITE-BYTES

Number of bytes written from a user application buffer. Available for all file open types.

Counter type: Accumulating.

OSS-CACHE-READS

For OSS regular files only, the number of READS performed using OSS cache.

Counter type: Incrementing.

OSS-CACHE-WRITES

For OSS regular files only, the number of WRITES performed using OSS cache.

Counter type: Incrementing.

OSS-CACHE-READ-BYTES

For OSS regular files only, the number of bytes read by OSS-CACHE-READS.

Counter type: Accumulating.

OSS-CACHE-WRITE-BYTES

For OSS regular files only, the number of bytes written by OSS-CACHE-WRITES.

Counter type: Accumulating.

OSS-BLOCK-READS

For OSS regular files only, the number of block read requests to DP2 issued to fill OSS cache. A block read call can return more than one block.

Counter type: Incrementing.

OSS-BLOCK-READ-BYTES

For OSS regular files only, the number of bytes read to cache as a result of OSS-BLOCK-READS.

Counter type: Accumulating.

OSS-FLOW-CONTROLS

For OSS sockets and pipes, this counter reports the number of times a WRITE operation was blocked.

Counter type: Incrementing.

MISC-CALLS

For OSS, this counter totals operations that increase FILE-BUSY-TIME but are not READS, WRITES, or INFO-CALLS. APIs that fall in this category are ioctl(), fsync(), ftruncate(), lseek(), and fcntl(), and socket APIs connect(), setsockopt(), accept(), and shutdown().

Counter type: Incrementing.

Usage Notes for All FILE Entities

- FILE counters are useful for detecting potential problems in database access.
- The Measure subsystem tracks only local entities. Therefore, the FILE entity type includes I/O operations performed by local user processes only. The files being accessed can be local, remote, or both, depending on the measurement configuration.
- The FILE entity type measures all partitions of a partitioned file together, as a single entity. To measure partitions individually, use the DISCOPEN entity type.
- The FILE entity type cannot measure files, such as alternate-key files, that are not explicitly opened by your application.
- Remote Data Facility and TMF I/O operations are not reported under the FILE entity because these operations are performed by direct interface with the disk process.

Usage Notes for G-Series FILE Entities

- In Measure G11 and later PVUs, FILE records all ANSI SQL file opens.
- In G-series RVUs, SUBSYSTEM-VERSION has a value of zero.
- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field MESSAGE-BYTES-F collects the same data as the 32-bit field MESSAGE-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

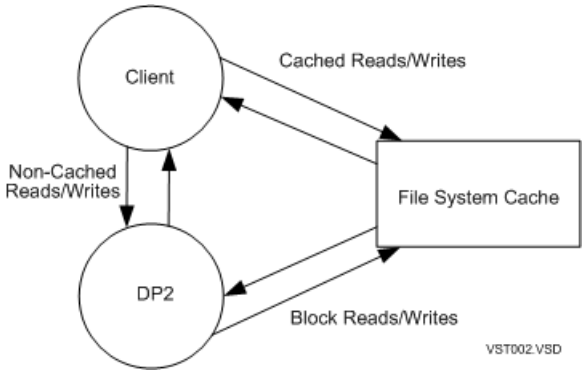
  The 32-bit fields are currently active and continue to return values. If no field overflow exists, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

  Convert your applications to use the 64-bit fields. The 32-bit fields might be deactivated in a future release.

  In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST DEVICE BY INPUT-BYTES, not LIST DEVICE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.
- With the Measure G09 PVU support for OSS file pathnames, file records are available for opens of OSS files only through the Guardian file system. With the Measure G10 PVU, the FILE entity records opens by the OSS file system.

- A read or write operation on an OSS regular file can be cached or noncached:
  - Noncached operations result in messages sent directly to DP2 and transfer only as much information as the client specifies.
  - Cached operations utilize the file-system cache in the client's CPU and can result in a block read/write between the cache and DP2.

    Block reads or writes occur on cache block boundaries (except for the last block of the file) where a cache block is 4 KB. Block read operations try to read ahead to anticipate any need for the information in the immediate future.



- The FILE entity counters for READS and WRITES count all logical file I/O operations, including reads and writes using OSS cache. They do not include block operations to replenish cache. Therefore, on a cache miss, six counters are incremented:

READS	OSS-CACHE-READS	OSS-BLOCK-READS
READ-BYTES	OSS-CACHE-READ-BYTES	OSS-BLOCK-READ-BYTES

- The FILE entity does not contain a counter for OSS block writes or block write bytes because a cache write is not necessarily triggered by the owning file. These counters are added to the DISKFILE entity.
- The OSS file system uses an internal algorithm to evaluate the effectiveness of cache file operations. Under some circumstances, caching is turned off by the system for a file open, so cache counters are not incremented.

Usage Notes for H-Series and J-Series FILE Entities

- In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.
- In H-series and J-series RVUs, SUBSYSTEM-VERSION for ZMSFILE records is provided by the Guardian or OSS file system, depending on the type of open represented by the ZMSFILE record.

Command Examples: OSS File Opens

These command usage examples illustrate the use of file identifiers in selecting Measure file reports.

OSS Opens of Disk Files

File opens of the same disk file share the same Guardian format file name and OSS file pathname.

To select file opens of the same disk file, use either its Guardian format file name or OSS file pathname:

```
LIST FILE $VOL.SUBVOL.FILE:CRVSN

LIST FILE "/a/b/c"
```

The Guardian format name can be masked to select disk file opens with the same device name or with the same device name and subvolume name:

```
LIST FILE $VOL.*.*

LIST FILE $VOL.SUBVOL.*
```

The Measure file open type selects OSS file opens of OSS regular files and Guardian files opened via /G:

```
LIST FILE * (OSSDISK)

LIST FILE $VOL.*.* (OSSDISK)
```

OSS Opens of FIFOs

OSS file opens of the same OSS FIFO are selected by the Guardian-format file name or OSS file pathname. A LISTGNAME command on a FIFO file pathname returns the Guardian device name in the form of $ZPPNN. Measure records display the file open name in the same format ($ZPPNN.Z*nnnnn*.Z*iiiiiii*:CRVSN) for all FIFO opens. However, in the device name field of the record, the NN is always replaced with the CPU number of the FIFO opener. If more than one opener in different CPUs is active, the $ZPP device name differs. A LIST FILE by FIFO pathname returns all instances:

```
LIST FILE $ZPPNN.*.*:ccccccccc      (ccccccccc is the CRSVN)
LIST FILE "/a/b/fifoz"
```

The Measure file open type can be used to select OSS file opens of OSS FIFOs:

```
LIST FILE * (OSSFIFO)
LIST FILE $ZPPNN.*.* (OSSFIFO)
```

In an Expand network environment, if an OSS file open attaches to a FIFO located on a remote system but the remote is not running G06.17 or later, no Measure file record is created. For example:

```
>list file $ZPPNN.*.* (OSSFIFO)
File Open $ZPPNN.Z00001.Z0007HRX:106589078765          Open Type OSSFIFO
OSSPath: "/home/software/markr/x"
Device Name $ZPP04            Device Type  0  (Process)      Subdevice Type  0
Opener 4,397                  File Num 6       OSSPID:  269223152
Program  $ROOT.ZYQ00000.Z0000HJM:874302705
```

OSS Opens of Pipes

OSS opens of the same pipe have the same Guardian format file name but have no OSS file pathname. Both files used to form the pipe (read/write) share the same Guardian format file name. Measure records are created by file open of the pipe, not its creation.

To select OSS file opens of the same pipe, use its Guardian format filename:

```
LIST FILE $ZPPnn.*.*:ccccccccc (nn =CPU number, ccccccccc = CRSVN))
```

The Guardian format name is masked to select OSS pipe and FIFO opens with the same device name:

```
LIST FILE $ZPPnn.*.*     (nn =CPU number)
```

The Measure file open type selects OSS file opens of OSS pipes:

```
LIST FILE * (OSSPIPE)
```

OSS Opens of AF_UNIX Sockets

Mask the Guardian format name or use the Measure file open type to select OSS socket opens of OSS AF_UNIX sockets:

```
LIST FILE $ZPLS.*.*              (pre-AF_UNIX R2)
LIST FILE $ZLSNN.*.*             (AF_UNIX R2 - bound sockets)
LIST FILE $ZLSnn.*.*             (AF_UNIX R2 - unbound sockets)
LIST FILE * (OSSUNIXSOCKET)
LIST FILE * (OSSUNIXSTREAM)
LIST FILE * (OSSUNIXDGRAM)
```

OSS Opens of AF_UNIX Sockets Using socketpair()

OSS opens of AF_UNIX sockets due to socketpair() share the same Guardian format file name but have no OSS file pathname. Both socket opens created by socketpair() share the same Guardian format file name, which can select the shared OSS socket opens:

```
LIST FILE $ZPLS.*.*     (pre-AF_UNIX R2)
LIST FILE $ZLSnn.*.*    (AF_UNIX R2 - unbound sockets, nn =CPU number)
```

OSS Opens of AF_UNIX Stream Sockets Using socket() or accept()

OSS opens of OSS AF_UNIX stream sockets participating in the socket connections represented by the server socket address all share the same Guardian format file name and OSS file pathname of the listening socket. This includes the sockets involved in the bind(), connect(), and accept() calls. OSS opens of unbound and unconnected AF_UNIX stream sockets have unique Guardian format file names and no OSS file pathname.

To select OSS socket opens of the same AF_UNIX socket stream connection, use the Guardian format file name or OSS file pathname of the listening socket:

```
LIST FILE $ZPLS.xxxxxx.xxxxxxxx:CRVSN
LIST FILE $ZLSNN.*.*          (AF_UNIX R2 - bound sockets)
LIST FILE "/a/b/socketf"
```

OSS Opens of AF_UNIX Datagram Sockets

Bound AF_UNIX datagram socket opens have unique Guardian format names and an OSS file pathname. Unbound AF_UNIX datagram socket opens have unique Guardian format names and no OSS file pathname.

To select OSS opens of bound datagram sockets use their bound OSS file pathname:

```
LIST FILE "/a/b/socketf"
```

OSS Opens of OSS AF_INET and AF_INET6 Sockets

To select opens of AF_INET and IF_INET6 sockets, mask the Guardian format name or use the Measure file open type:

```
LIST FILE $ZTC0.*.*

LIST FILE * (OSSINETSOCKET)

LIST FILE * (OSSINETSTREAM)

LIST FILE * (OSSINETDGRAM)
```

OSS Opens of OSS AF_INET and AF_INET6 Stream Sockets

OSS opens of AF_INET and AF_INET6 stream sockets involved in the socket connection (using the same bound socket address) all share the IP address of the listening socket. This includes the sockets involved in the bind(), connect() and accept() calls. If the IP address of the listening socket is 0.0.0.0 (bound to multiple IP addresses), the IP address of an accepted socket is its own socket address.

OSS opens of AF_INET and AF_INET6 stream sockets involved in a socket connection have unique Guardian format names, except accepted AF_INET or AF_INET6 stream sockets, which share their Guardian format name with the listening AF_INET or AF_INET6 stream socket.

To select OSS socket opens of the same AF_INET or AF_INET6 socket stream connection, use the IP address of the listening socket. If the IP address of the listening socket is 0.0.0.0 (bound to multiple IP addresses), the IP address of an accepted socket is its own socket address.

```
LIST FILE $ZTC0.*.*, if IP-ADDR = "1.2.3.4 555"

LIST FILE *, if IP-ADDR = "1.2.3.4 *"   IP address = 1.2.3.4, any port number

LIST FILE *, if IP-ADDR = "* 37"        IP address = any IP address, port 37

LIST FILE *, if IP-ADDR = "1.2.3.4 37"  IP address = 1.2.3.4, port 37

LIST FILE *, if IP-ADDR = "1070::800:2006:4178 37"  IPv6 address, port 37
```

OSS Opens of OSS AF_INET and AF_INET6 Datagram Sockets

Bound AF_INET or AF_INET6 datagram socket opens have unique Guardian format names and use the socket address as IP address. Unbound AF_INET or AF_INET6 datagram socket opens have unique Guardian format names but no IP address.

OSS opens of an OSS AF_INET or AF_INET6 datagram socket can be selected using its bound socket address:

```
LIST FILE *, if IP-ADDR = "1.2.3.4 *"   IP address = 1.2.3.4, any port number

LIST FILE *, if IP-ADDR = "* 37"        IP address = any IP address, port 37

LIST FILE *, if IP-ADDR = "1.2.3.4 37"  IP address = 1.2.3.4, port 37

LIST FILE *, if IP-ADDR = "1070::800:2006:4178 37"  IPv6 address, port 37
```

For more information about IPv6 addresses, see the *TCP/IPv6 Configuration and Management Manual*.

Example

The report output for the FILE entity includes the ANSI SQL name for files that have an ANSI SQL name. Here is an example:

```
File Open $DATA.ZSD12345.Z1234567                    Open Type
ANSI SQL Name 'TABLE Catalog_12.Schema_45.Table_56'
Device Name                  Device Type  3  (Disk)
Opener 0,285                 File Num
Program
Opener Device Name $SYSTEM
Format Version:  H01  Data Version:  H01  Subsystem Version:  1
Local System \HURTS   From  12 Jan 2003, 19:35:28   For  46.8 Seconds
------------ Requests ------------------------------------------------------
File-Busy-Time                        Disc-Reads
Reads                                 Writes
Deletes-or-Writereads                 Updates-or-Replies
Timeouts-Or-Cancels                   Info-Calls
Misc-Calls
------------ Logical I/O ---------------------------------------------------
Messages                      15 #    Message-Bytes              47,558 #
Read-Bytes                            Write-Bytes
DBIO-Reads                            DBIO-Writes
DBIO-Read-Bytes                       DBIO-Write-Bytes
Lock-Waits                            Escalations
------------ SQL -----------------------------------------------------------
Records-Accessed                      Records-Used
------------ Open System Services ------------------------------------------
OSS-Cache-Reads                       OSS-Cache-Writes
OSS-Cache-Read-Bytes                  OSS-Cache-Write-Bytes
OSS-Block-Reads                       OSS-Flow-Controls
OSS-Block-Read-Bytes
```

LINE

The LINE entity type provides information about one or more communication lines.

Entity Specification Syntax for LINE Entities

To describe LINE entities:

NOTE: You can use a D-series measurement application to measure G-series LINE entities if the application specifies all LINE entities (ADD LINE *) or specifies only *$device* or *$device (cpu)*. If an application specifies *controller*, *channel*, or *unit*, you must modify the entity identifiers to measure G-series LINE entities.

```
LINE entity-spec
```

LINE

collects information about one or more communication lines.

entity-spec

is specified as:

```
{ *                                                    }
{ $device [ ( cpu [ , channel [ , ctrl [ , unit ] ] ] ) ]}
{ $line                                          }
{ cpu [ ( %Htrackid [, clip [, line ] ] ) ] [ ( type [, subtype ] ) ] }
```

*

measures all communication lines in all CPUs.

$*device*

(D-series) is the device name of the line to be measured. To indicate all lines, use an asterisk (*).

cpu

is the number of the CPU in which the line to be measured is configured. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

channel

(D-series) is the channel number of the line to be measured. The default is all channels.

ctrl

(D-series) is the controller number of the line to be measured. The default is all controllers.

unit

(D-series) is the unit number of the line to be measured. The default is all units.

$*line*

(G-series) is the device name of the line to be measured. Use an asterisk (*) to indicate all lines.

%Htrackid

(G-series) is the identifier of a specific SWAN concentrator enclosure of three CLIPs. (Each CLIP controls two lines.) To indicate all TRACKIDs, use an asterisk (*).

The TRACKID is recorded in the SEEPROM of the controller. The TRACKID is usually printed on an external label on the enclosure as well.

clip

(G-series) is 1, 2, or 3 to indicate a specific CLIP within a SWAN concentrator enclosure, or it is 1, 2, 3, 4, 5, or 6 to indicate a specific CLIP within a SWAN 2 concentrator enclosure. To indicate all CLIPs, use an asterisk (*).

line

(G-series) is 0 or 1 to indicate a specific line managed by a CLIP in a SWAN concentrator. To indicate all lines, use an asterisk (*).

type

(G-series) is a specific line type. (To list the valid values for *type*, use SCF LISTDEV.) Specifying *type* selects only lines of a particular protocol, such as type 61 (X25AM). To indicate all types, use an asterisk (*). The default is all types.

subtype

(G-series) provides further information about the line type. (Use SCF LISTDEV to see the valid values for *subtype*.) *subtype* is subsystem specific in value. The default is all subtypes that apply to *type*.

DDL Record for LINE Entities (Legacy Style)

This is the Legacy Style DDL record for LINE entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

The DDL record for G-series LINE entities is identical to the record for D-series LINE entities except:

- Different entity identification fields are used. The CTRL, UNIT, and CHANNEL fields are no longer used.
- Longer byte-count fields are provided to reduce the possibility of field overflow. In G-series RVUs, each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Notes for G-Series LINE Entities (page 243). In H-series and J-series RVUs, all byte-count fields accommodate 64 bits.
- The ERROR field can signal a field overflow in a 32-bit byte-count field.

```
RECORD line. FILE is "line" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 device-name                  type character 8.
 02 logical-device               type binary 16 unsigned.
 02 ctrl                         type binary 16 unsigned.
 02 unit                         type binary 16 unsigned.
 02 device-type                  type binary 16 unsigned.
 02 device-subtype               type binary 16 unsigned.
* counter value items:
 02 read-busy-time               type binary 64.
```

```
 02 write-busy-time            type binary 64.
 02 requests                   type binary 32 unsigned.
 02 input-bytes                type binary 32 unsigned.
 02 output-bytes               type binary 32 unsigned.
 02 input-data-bytes           type binary 32 unsigned.
 02 output-data-bytes          type binary 32 unsigned.
 02 reads                      type binary 32 unsigned.
 02 writes                     type binary 32 unsigned.
 02 retries                    type binary 32 unsigned.
 02 transactions               type binary 32 unsigned.
 02 response-time              type binary 64.
* New entity identification item for D10:
 02 channel                    type binary 16 unsigned.
* F40 new entity identification items:
 02 trackid                   type character 6.
 02 clip                      type binary 16 unsigned.
 02 line                      type binary 16 unsigned.
* F40 new counter value items:
 02 input-bytes-f             type binary 64.
 02 output-bytes-f            type binary 64.
 02 input-data-bytes-f        type binary 64.
 02 output-data-bytes-f       type binary 64.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for LINE Entities (ZMS Style)

The ZMS style DDL record for LINE entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsline-id.
 02 device-name                type character 8.
 02 logical-device             type binary 32 unsigned.
 02 device-type                type binary 16 unsigned.
 02 device-subtype             type binary 16 unsigned.
 02 pin                        type binary 16 unsigned.
 02 trackid                    type character 6.
 02 clip                       type binary 16 unsigned.
 02 line                       type binary 16 unsigned.
 02 reserved-1                 type character 4.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsline-ctrs.
 02 read-busy-time             type binary 64.
 02 write-busy-time            type binary 64.
 02 requests                   type binary 64.
 02 input-bytes                type binary 64.
 02 output-bytes               type binary 64.
 02 input-data-bytes           type binary 64.
 02 output-data-bytes          type binary 64.
 02 reads                      type binary 64.
 02 writes                     type binary 64.
 02 retries                    type binary 64.
 02 transactions               type binary 64.
 02 response-time              type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsline. FILE is "zmsline" ENTRY-SEQUENCED.
 02 hdr                          type zmsheader.
 02 ctr                          type zmsline-ctrs.
 02 id                           type zmsline-id.
end
```

PIN

Process identification number of the primary process.

DEVICE-NAME

Device name of the measured line.

LOGICAL-DEVICE

Logical device number of the measured line.

CTRL

(Legacy Style only) For D-series, controller number of the measured line. For G-series RVUs, not used; returns zero.

UNIT

(Legacy Style only) For D-series, unit number of the measured line. For G-series RVUs, not used; returns zero.

DEVICE-TYPE

Device type, such as 54 for an ATP6100 line. For a list of device type values for communications lines, see the *System Generation Manual*.

DEVICE-SUBTYPE

Additional identifier for DEVICE-TYPE. For a list of subtype values for communications lines, see the *System Generation Manual*.

READ-BUSY-TIME

The time spent reading from the communications line. This counter measures the time from the instruction that starts the read to the interrupt that signals the end of the I/O operation. Any process can run during this time.

For SNAX and AM6520 lines, this counter is always close to 100 percent busy because polling protocols constantly have a read outstanding.

Counter type: Busy.

WRITE-BUSY-TIME

The time spent writing to the communications line. This counter measures the time from the instruction that starts the write to the interrupt that signals the end of the I/O operation. Any process can run during this time.

Counter type: Busy.

REQUESTS

Number of requests received by the I/O process for the communications line. To determine how long the requests were queued before being read by the I/O process, measure the I/O process and examine the RECV-QTIME counter of the PROCESS entity. After reading a request, the I/O process places it on an internal queue.

Counter type: Incrementing.

INPUT-BYTES

Number of bytes read from the communication line.

For SNAXLINK lines, INPUT-BYTES is the total number of bytes received from a SNAXLINK read unit. INPUT-BYTES includes overhead for the request/response header (RH), request/response unit (RU), transmission header (TH), and HDLC frame (3 + RU + 6 + approximately 5 bytes, assuming only one PIU has arrived in the frame).

For all other lines, INPUT-BYTES is the total number of bytes received from a read unit. INPUT-BYTES includes the RH + RU + TH + SDLC frame overhead (3 + RU + 6 + 6 bytes).

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the communication line.

For SNAXLINK lines, OUTPUT-BYTES is the total number of bytes sent on the SNAXLINK write unit. OUTPUT-BYTES includes overhead for the request/response header (RH), request/response unit (RU), transmission header (TH), and HDLC frame (3 + RU + 6 + approximately 6 bytes, assuming only one PIU is being sent in the frame).

PIUs can be sent to the SNAXLINK unit in blocked format. Therefore, the overhead cannot be calculated exactly.

For all other lines, OUTPUT-BYTES is the total number of bytes sent on the write unit. OUTPUT-BYTES includes the RH + RU + TH + SDLC frame overhead (3 + RU + 6 + 6 bytes).

Counter type: Accumulating.

INPUT-DATA-BYTES

Number of data bytes read from the communication line. This counter includes only user data. Data required by line protocol is ignored.

For SNAX lines, this counter includes only request/response header (RH) and RU data counts (3 + RU bytes) of RUs destined for an OPEN logical unit or a logical unit in a pass-through session.

All INPUT-DATA-BYTES events are recorded on the write unit of the line.

The INPUT-DATA-BYTES count is always less than the INPUT-BYTES count.

Counter type: Accumulating.

OUTPUT-DATA-BYTES

Number of data bytes written to the communication line. This counter includes only user data. Data required by line protocol is ignored.

For SNAX lines, this counter includes only the request/response header (RH) and RU data counts (3 + RU bytes) of RUs being sent on the line.

The OUTPUT-DATA-BYTES count is always less than the OUTPUT-BYTES count.

Counter type: Accumulating.

READS

Number of read operations (from the communications line to memory) performed by the I/O process. Because the I/O process modifies this counter before the I/O operation, this counter includes both successful and unsuccessful operations. The RETRIES counter counts the number of I/O operations retries due to failures.

Counter type: Incrementing.

WRITES

Number of write operations (from memory to the communications line) performed by the I/O process. Because the I/O process modifies this counter before the I/O operation, this counter

includes both successful and unsuccessful operations. The RETRIES counter counts the number of I/O operations retries due to failures.

Counter type: Incrementing.

RETRIES

Number of times the I/O process retried an I/O operation because of an error on the communications line.

Counter type: Incrementing.

TRANSACTIONS

Number of terminal transactions performed by the I/O process for all measured subdevices on the line. A transaction is a read from a terminal followed immediately by a write to the terminal. Terminal response is the interval between the end of the read and the beginning of the write. RESPONSE-TIME measures the time spent on the transactions.

For SNAX lines, this counter is always zero.

Counter type: Incrementing.

RESPONSE-TIME

The time that the I/O process spent on terminal response for all measured subdevices on the line. A transaction is a read from a terminal followed immediately by a write to the terminal. Terminal response is the interval between the end of the read and the beginning of the write. TRANSACTIONS counts the number of transactions.

For SNAX lines, this counter is always zero.

Counter type: Response.

CHANNEL

(Legacy Style only) For D-series, channel number of the line. For G-series RVUs, not used; returns zero.

TRACKID

The 6-byte identifier returned by the 3880 controller SEEPROM and visible externally on the unit.

CLIP

The number of counter ID control blocks available for new counter records in this processor at the end of the reported measurement interval.

LINE

The specific line within a CLIP on a 3880 SWAN concentrator: either 0 or 1.

INPUT-BYTES

Number of bytes read from the communication line.

For SNAXLINK lines, INPUT-BYTES is the total number of bytes received from a SNAXLINK read unit. INPUT-BYTES includes overhead for the request/response header (RH), request/response unit (RU), transmission header (TH), and HDLC frame (3 + RU + 6 + approximately 5 bytes, assuming only one PIU has arrived in the frame).

For all other lines, INPUT-BYTES is the total number of bytes received from a read unit. INPUT-BYTES includes the RH + RU + TH + SDLC frame overhead (3 + RU + 6 + 6 bytes).

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the INPUT-BYTES-F field is a 64-bit version of INPUT-BYTES.

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the communications line.

For SNAXLINK lines, OUTPUT-BYTES is the total number of bytes sent on the SNAXLINK write unit. OUTPUT-BYTES includes overhead for the request/response header (RH), request/response unit (RU), transmission header (TH), and HDLC frame (3 + RU + 6 + approximately 6 bytes, assuming only one PIU is being sent in the frame).

PIUs can be sent to the SNAXLINK unit in blocked format. Therefore, the overhead cannot be calculated exactly.

For all other lines, OUTPUT-BYTES is the total number of bytes sent on the write unit. OUTPUT-BYTES includes the RH + RU + TH + SDLC frame overhead (3 + RU + 6 + 6 bytes).

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the OUTPUT-BYTES-F field is a 64-bit version of OUTPUT-BYTES.

Counter type: Accumulating.

INPUT-DATA-BYTES

Number of data bytes read from the communications line. This counter includes only user data. Data required by line protocol is ignored.

All INPUT-DATA-BYTES events are recorded on the write unit of the line. The INPUT-DATA-BYTES count is always less than the INPUT-BYTES count.

For SNAX lines, this counter includes only request/response header (RH) and RU data counts (3 + RU bytes) of RUs destined for an OPEN logical unit or a logical unit in a pass-through session.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the INPUT-DATA-BYTES-F field is a 64-bit version of INPUT-DATA-BYTES.

Counter type: Accumulating.

OUTPUT-DATA-BYTES

Number of data bytes written to the communications line. This counter includes only user data. Data required by line protocol is ignored.

The OUTPUT-DATA-BYTES count is always less than the OUTPUT-BYTES count.

For SNAX lines, this counter includes only the request/response header (RH) and RU data counts (3 + RU bytes) of RUs being sent on the line.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the OUTPUT-DATA-BYTES-F field is a 64-bit version of OUTPUT-DATA-BYTES.

Counter type: Accumulating.

INPUT-BYTES-F

(G-series, Legacy Style only) Same as INPUT-BYTES but accommodates larger values (64 bits rather than 32).

OUTPUT-BYTES-F

(G-series, Legacy Style only) Same as OUTPUT-BYTES but accommodates larger values (64 bits rather than 32).

INPUT-DATA-BYTES-F

(G-series, Legacy Style only) Same as INPUT-DATA-BYTES but accommodates larger values (64 bits rather than 32).

OUTPUT-DATA-BYTES-F

(G-series, Legacy Style only) Same as OUTPUT-DATA-BYTES but accommodates larger values (64 bits rather than 32).

Usage Notes for G-Series LINE Entities

- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field INPUT-BYTES-F collects the same data as the 32-bit field INPUT-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. If no field overflow exists, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

  Convert your applications to use the 64-bit fields. The 32-bit fields might be deactivated in a future RVU.

  In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST LINE BY INPUT-BYTES, not LIST LINE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.
- SUBSYSTEM-VERSION for ZMSLINE records is provided by the specific product subsystem of the line represented by each ZMSLINE record; for example, SNAX/XF, X25AM, or ENVOY. In G-series RVUs, the value of SUBSYSTEM-VERSION is zero.
- The G09 PVU of Measure supports up to six CLIPs in a SWAN concentrator.

Usage Notes for H-Series and J-Series LINE Entities

- In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.
- The SUBSYSTEM-VERSION is set to 1 for the first H-series and J-series version of the subsystem.

NETLINE

The NETLINE entity type provides information about network communications lines.

Entity Specification Syntax for NETLINE Entities

To describe NETLINE entities:

```
NETLINE entity-spec
```

NOTE: You can measure G-series network communication lines with a D-series measurement application if it specifies all NETLINE entities (ADD NETLINE *) or only *$device* or *$device (cpu)*. If the application specifies *controller*, *channel*, or *unit*, you must modify the entity identifiers to measure G-series NETLINE entities.

NETLINE

collects information about one or more network communications lines.

entity-spec

is specified as:

```
{ *                                                          }
{ $device [ ( cpu [ , channel [ , ctrl [ , unit ] ] ] ) ]}
{ $line                                                 }
{ cpu [ ( %Htrackid [, clip [, line ] ] ) ] [ ( type [, subtype ] ) ] }
```

*

measures all network communications lines in all CPUs.

$*device*

(D-series) is the device name of the line to be measured. Use an asterisk (*) to indicate all network lines.

cpu

is the number of the CPU in which the measured line is configured. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

channel

(D-series) is the channel number of the line to be measured. The default is all channels.

ctrl

(D-series) is the controller number of the line to be measured. The default is all controllers.

unit

(D-series) is the unit number of the line to be measured. The default is all units.

$*line*

is the device name of the line to be measured. To indicate all lines, use an asterisk (*). The default is all lines.

%Htrackid

is the identifier of a specific 3880 SWAN concentrator enclosure of three CLIPs. (Each CLIP controls two lines.) To indicate all TRACKIDs, use an asterisk (*).

The TRACKID is recorded in the SEEPROM of the controller. It is usually printed on an external label on the enclosure as well.

clip

is 1, 2, or 3 to indicate a specific CLIP within a SWAN concentrator enclosure, or it is 1, 2, 3, 4, 5, or 6 to indicate a specific CLIP within a SWAN 2 concentrator enclosure. To indicate all CLIPS, use an asterisk (*).

line

is 0 or 1 to indicate a specific line managed by a CLIP in a 3880 SWAN concentrator. To indicate all lines, use an asterisk (*).

type

is a type number to indicate a particular protocol, such as 63 (Expand line handler). To list valid values for line types, use SCF LISTDEV. To indicate all line types, use an asterisk (*).

subtype

is a subtype number, such as 3 (FOX line handler). To see valid subtype numbers, use SCF LISTDEV. The default is all subtypes that apply to *type*.

DDL Record for NETLINE Entities (Legacy Style)

This is the Legacy Style DDL record for NETLINE entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

The DDL record for G-series NETLINE entities is identical to the record for D-series NETLINE entity, except:

- Different entity identification fields are used. The CHANNEL, CTRL, and UNIT fields are no longer used.
- Longer byte-count fields are provided to prevent field overflow. Each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Notes for G-Series NETLINE Entities (page 251).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.

```
RECORD netline. FILE is "netline" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 device-name                  type character 8.
 02 logical-device               type binary 16 unsigned.
 02 ctrl                         type binary 16 unsigned.
 02 unit                         type binary 16 unsigned.
 02 device-type                  type binary 16 unsigned.
 02 device-subtype               type binary 16 unsigned.
* counter value items:
 02 write-busy-time              type binary 64.
 02 read-busy-time               type binary 64.
 02 requests                     type binary 32 unsigned.
 02 reads                        type binary 32 unsigned.
 02 writes                       type binary 32 unsigned.
 02 l2in-bytes                   type binary 32 unsigned.
 02 l2out-bytes                  type binary 32 unsigned.
 02 din4-bytes                   type binary 32 unsigned.
 02 dout4-bytes                  type binary 32 unsigned.
 02 cin4-bytes                   type binary 32 unsigned.
 02 cout4-bytes                  type binary 32 unsigned.
 02 u64-bytes                    type binary 32 unsigned.
 02 u128-bytes                   type binary 32 unsigned.
 02 u256-bytes                   type binary 32 unsigned.
 02 u512-bytes                   type binary 32 unsigned.
 02 u1024-bytes                  type binary 32 unsigned.
 02 u2048-bytes                  type binary 32 unsigned.
 02 u4096-bytes                  type binary 32 unsigned.
 02 o4095-bytes                  type binary 32 unsigned.
* new entity identification item for D10:
```

```
 02 channel                    type binary 16 unsigned.
* F40 new entity identification items:
 02 TrackID                    type character 6.
 02 Clip                       type binary 16 unsigned.
 02 Line                       type binary 16 unsigned.
* F40 new counter value items:
 02 l2in-bytes-f               type binary 64.
 02 l2out-bytes-f              type binary 64.
 02 din4-bytes-f               type binary 64.
 02 dout4-bytes-f              type binary 64.
 02 cin4-bytes-f               type binary 64.
 02 cout4-bytes-f              type binary 64.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for NETLINE Entities (ZMS Style)

The ZMS style DDL record for NETLINE entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsnline-id.
 02 device-name                 type character 8.
 02 logical-device              type binary 32 unsigned.
 02 device-type                 type binary 16 unsigned.
 02 device-subtype              type binary 16 unsigned.
 02 pin                         type binary 16 unsigned.
 02 trackid                     type character 6.
 02 clip                        type binary 16 unsigned.
 02 line                        type binary 16 unsigned.
 02 reserved-1                  type character 4.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsnline-ctrs.
 02 write-busy-time             type binary 64.
 02 read-busy-time              type binary 64.
 02 requests                    type binary 64.
 02 reads                       type binary 64.
 02 writes                      type binary 64.
 02 l2in-bytes                  type binary 64.
 02 l2out-bytes                 type binary 64.
 02 din4-bytes                  type binary 64.
 02 dout4-bytes                 type binary 64.
 02 cin4-bytes                  type binary 64.
 02 cout4-bytes                 type binary 64.
 02 u64-bytes                   type binary 64.
 02 u128-bytes                  type binary 64.
 02 u256-bytes                  type binary 64.
 02 u512-bytes                  type binary 64.
 02 u1024-bytes                 type binary 64.
 02 u2048-bytes                 type binary 64.
 02 u4096-bytes                 type binary 64.
 02 o4095-bytes                 type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsnline. FILE is "zmsnline" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsnline-ctrs.
```

```
 02 id                          type zmsnline-id.
end
```

PIN

Process identification number of the primary process.

DEVICE-NAME

Device name of the measured line.

LOGICAL-DEVICE

Logical device number of the measured line.

CTRL

(Legacy Style only) For D-series, controller number of the measured line. For G-series RVUs, not used; returns zero.

UNIT

(Legacy Style only) For D-series, unit number of the measured line. For G-series RVUs, not used; returns zero.

DEVICE-TYPE

A value of 63, which indicates an Expand line.

DEVICE-SUBTYPE

Additional information about the device type:

0	Single-line handler attached to a controller
1	Multiline path handler
2	Multiline line attached to a controller
3	FOX line handler
5	Single-line handler attached to a 6100 subsystem
6	Multiline line attached to a 6100 subsystem

WRITE-BUSY-TIME

The time spent writing to the network communications line. This counter measures the time from the instruction that starts the write to the interrupt that signals the end of the I/O operation. Any process can run during this time.

Counter type: Busy.

READ-BUSY-TIME

The time spent reading data from the network communications line. This counter includes the time spent waiting for input to the line. The counter measures the time from the instruction that starts the read to the interrupt that signals the end of the I/O operation.

Because active, full-duplex lines usually have a read continue outstanding, expect a high value for this counter. Any process can run during this time.

Counter type: Busy.

REQUESTS

The sum of all SYSTEM LINKS handled by this path plus the sum of all SYSTEM SENT FORWARDs (PASSTHRU REQUESTS) handled by this path.

LINKS is a message-level number. SENT, RECEIVED, SENT FORWARD, and RECEIVE FORWARD are packet counters. One message can be several packets.

REQUESTS counts both message-level events (LINKS) and packet-level events (SENT FORWARD) in the same counter. Requests are the number of message quick cells (MQCs) this line handler receives that are sent to the remote system.

Counter type: Incrementing.

READS

Number of read operations (from the network communications line to memory) performed by the line-handler process.

Counter type: Incrementing.

WRITES

Number of write operations (from memory to the network communications line) performed by the line-handler process.

Counter type: Incrementing.

L2IN-BYTES

Number of Level 2 bytes read by the network line handler. This counter includes both user data and data associated with line protocol.

For non-CSS type controllers, this counter contains DIN4-BYTES and CIN4-BYTES. For CSS type controllers, this counter also contains CLB (CAP to CLIP) protocol and CLIP TRACE blocks.

The total number of Level 2 bytes on the network line is this counter plus L2OUT-BYTES.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the L2IN-BYTES-F field is a 64-bit version of L2IN-BYTES.

Counter type: Accumulating.

L2OUT-BYTES

Number of Level 2 bytes written by the network line handler. This counter includes both user data and data associated with line protocol.

For non-CSS type controllers, this counter contains DOUT4-BYTES and COUT4-BYTES. For CSS type controllers, this counter also contains CLB protocol bytes.

The total number of Level 2 bytes on the network line is this counter plus L2IN-BYTES.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the L2OUT-BYTES-F field is a 64-bit version of L2OUT-BYTES.

Counter type: Accumulating.

DIN4-BYTES

Number of Level 4 data bytes read by the network line handler. This counter includes only user data and, on D-series, message system protocol bytes. The data associated with line protocol is ignored.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the DIN4-BYTES-F field is a 64-bit version of DIN4-BYTES.

Counter type: Accumulating.

DOUT4-BYTES

Number of Level 4 data bytes written by the network line handler. This counter includes only user data and, on D-series, message system protocol bytes. The data associated with line protocol is ignored.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the DOUT4-BYTES-F field is a 64-bit version of DOUT4-BYTES.

Counter type: Accumulating.

CIN4-BYTES

Number of Level 4 protocol bytes and NCP requests bytes (ACK, NAK, ENQ, PCHG, NCPM, CONN, and CAN) read by the network line handler for an Expand line.

X.25 does not update this counter.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the CIN4-BYTES-F field is a 64-bit version of CIN4-BYTES.

Counter type: Accumulating.

COUT4-BYTES

Number of Level 4 protocol bytes and NCP request bytes (ACK, NAK, ENQ, PCHG, NCPM, CONN, and CAN) written by the network line handler for an Expand line.

X.25 does not update this counter.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

For G-series RVUs, the COUT4-BYTES-F field is a 64-bit version of COUT4-BYTES.

Counter type: Accumulating.

U64-BYTES

Number of messages shorter than 64 bytes after compression. The size includes protocol bytes.

Counter type: Incrementing.

U128-BYTES

Number of messages from 64 through 127 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

U256-BYTES

Number of messages from 128 through 255 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

U512-BYTES

Number of messages from 256 through 511 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

U1024-BYTES

Number of messages from 512 through 1023 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

U2048-BYTES

Number of messages from 1024 through 2047 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

U4096-BYTES

Number of messages from 2048 through 4095 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

O4095-BYTES

Number of messages longer than 4095 bytes long after compression. The size includes protocol bytes.

Counter type: Incrementing.

CHANNEL

(Legacy Style only) For D-series, channel number of the measured line. For G-series RVUs, not used; returns zero.

TRACKID

The 6-byte identifier returned by the 3880 controller SEEPROM and labeled externally on the enclosure.

CLIP

The number of counter ID control blocks available for new counter records in this processor at the end of the reported measurement interval.

LINE

The specific line within a CLIP on a SWAN controller: either 0 or 1.

L2IN-BYTES-F

(G-series, Legacy Style only) Same as L2IN-BYTES but accommodates larger values (64 bits rather than 32).

L2OUT-BYTES-F

(G-series, Legacy Style only) Same as L2OUT-BYTES but accommodates larger values (64 bits rather than 32).

DIN4-BYTES-F

(G-series, Legacy Style only) Same as DIN4-BYTES but accommodates larger values (64 bits rather than 32).

DOUT4-BYTES-F

(G-series, Legacy Style only) Same as DOUT4-BYTES but accommodates larger values (64 bits rather than 32).

CIN4-BYTES-F

(G-series, Legacy Style only) Same as CIN4-BYTES but accommodates larger values (64 bits rather than 32).

COUT4-BYTES-F

(G-series, Legacy Style only) Same as COUT4-BYTES but accommodates larger values (64 bits rather than 32).

Usage Notes for All NETLINE Entities

- In general, the data collected by the Measure L2 counters is very close to the actual line byte counts. However, line problems or CLIP traces can alter the Measure byte counts.

  For CSS-type controllers, the information gathered by Measure is from the controllers to the line handler. Measure counters are not advanced when retransmission of data occurs because of line problems. SCF line statistics count events that cause retransmissions (timeouts and FCS errors). To get detailed information about line problems, use a datascope.

  When you take CLIP traces, the trace data is included in the L2IN-BYTES counter. Similarly, the CIN4-BYTES and COUT4-BYTES counters occasionally include NCP NETMAP packets.

- For descriptions of networks and network protocols, see the *Expand Configuration and Management Manual*, the *Expand Network Management and Troubleshooting Manual*, and the *X25AM Configuration and Management Manual*.

Usage Notes for G-Series NETLINE Entities

- SUBSYSTEM-VERSION for ZMSNLINE records is provided by Expand.
- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field L2IN-BYTES-F collects the same data as the 32-bit field L2IN-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. If no field overflow exists, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

  Convert your applications to use the 64-bit fields. The 32-bit fields might be deactivated in a future RVU.

  In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST LINE BY INPUT-BYTES, not LIST LINE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

  The Measure G09 and later PVUs support up to six CLIPs in a SWAN concentrator.

Usage Notes for H-Series and J-Series NETLINE Entities

In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.

OPDISK

The OPDISK entity type provides information about one or more optical disk systems on the local system.

The optical disk system is an enclosure (familiarly called a jukebox) that contains one or more disk drives and multiple disk cartridges stored in cells. Each cartridge has two sides, and each side is a volume.

G-series, H-series, and J-series RVUs do not support optical disks. However, the G02 and later Measure PVUs include an OPDISK DDL record for future use. Specifying an OPDISK entity in a RVU that does not support the entity does not cause an error message but also does not affect the measurement configuration.

Optical disk records from D-series data can be formatted and displayed.

Entity Specification Syntax for OPDISK Entities

To describe OPDISK entities in D-series RVUs:

```
OPDISK entity-spec
```

OPDISK

collects information about one or more optical disks in the local system.

entity-spec

is specified as:

```
{ *                                                          }
{ $device [ $vol ] [ ( cpu [ , chan [ , ctrl [ , unit] ] ] ) ]}
```

*

measures all optical disks on the system.

$*device*

is the device name of the optical disk to be measured. Use an asterisk (*) to indicate all devices.

$*vol*

is the name of the optical disk volume to be measured. Use an asterisk (*) to indicate all volumes.

cpu

is the number of the CPU in which the optical disk is configured. The default is all CPUs.

chan

is the channel number of the optical disk to be measured. The default value is all channels.

ctrl

is the controller number of the optical disk to be measured (0-31). The default is all controllers.

unit

is the unit number of the optical disk to be measured. The default is all units.

DDL Record for OPDISK Entities (Legacy Style)

This is the Legacy Style DDL record for OPDISK entities.

The fields included in BRIEF reports are in **boldface type**.

```
RECORD opdisk.  FILE is "opdisk" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 device-name                  type character 8.
 02 logical-device               type binary 16 unsigned.
 02 ctrl                         type binary 16 unsigned.
 02 unit                         type binary 16 unsigned.
 02 device-type                  type binary 16 unsigned.
 02 device-subtype               type binary 16 unsigned.
 02 volume-name                  type character 8.
 02 cell                         type binary 16 unsigned.
 02 side                         type binary 16 unsigned.
* counter value items:
 02 request-qtime                type binary 64.
 02 request-qlen-max             type binary 16 unsigned.
 02 requests                     type binary 32 unsigned.
 02 read-busy-time               type binary 64.
 02 write-busy-time              type binary 64.
 02 seek-busy-time               type binary 64.
 02 reads                        type binary 32 unsigned.
 02 writes                       type binary 32 unsigned.
 02 seeks                        type binary 32 unsigned.
 02 input-bytes                  type binary 32 unsigned.
 02 output-bytes                 type binary 32 unsigned.
* new entity identification item for D10:
 02 channel                      type binary 16 unsigned.
 02 servernet                    type binary 16 unsigned
                                 redefines channel.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for OPDISK Entities (ZMS Style)

The ZMS style DDL record for OPDISK entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsodisk-id.
 02 pin                          type binary 16 unsigned.
 02 device-type                  type binary 16 unsigned.
 02 device-subtype               type binary 16 unsigned.
 02 servernet                    type binary 16 unsigned.
 02 device-name                  type character 8.
 02 logical-device               type binary 32 unsigned.
 02 GMS.
    03 group                     type binary 32 unsigned.
    03 module                    type binary 32 unsigned.
    03 slot                      type binary 32 unsigned.
 02 SCSI-id                      type binary 64.
 02 config-name                  type character 64.
 02 adapter-name                 type character 64.
 02 SAC-name                     type character 64.
 02 volume-name                  type character 8.
 02 cell                         type binary 16 unsigned.
```

```
 02 side                          type binary 16 unsigned.
 02 reserved-1                    type character 4.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsodisk-ctrs.
 02 request-qtime                 type binary 64.
 02 requests                      type binary 64.
 02 reads                         type binary 64.
 02 writes                        type binary 64.
 02 input-bytes                   type binary 64.
 02 output-bytes                  type binary 64.
 02 read-qbusy-time               type binary 64.
 02 read-qtime                    type binary 64.
 02 write-qbusy-time              type binary 64.
 02 write-qtime                   type binary 64.
 02 device-qbusy-time             type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsodisk. FILE is "zmsodisk" ENTRY-SEQUENCED.
 02 hdr                           type zmsheader.
 02 ctr                           type zmsodisk-ctrs.
 02 id                            type zmsodisk-id.
end
```

PIN

Process identification number of the optical disk process.

DEVICE-NAME

Name of the optical disk enclosure.

LOGICAL-DEVICE

Logical device number of the optical disk enclosure.

CTRL

(Legacy Style only) Controller number of the device.

UNIT

(Legacy Style only) Unit number of the device.

DEVICE-TYPE

A value of 30 (optical disk device).

DEVICE-SUBTYPE

An additional identifier for DEVICE-TYPE. For a complete list of possible subtype values for optical disks, see the *System Generation Manual*.

VOLUME-NAME

Name of an optical disk volume. An optical disk volume is one side of an optical disk cartridge.

CELL

Number of a storage cell in an optical disk enclosure.

SIDE

Number that identifies a side of an optical disk cartridge.

REQUEST-QTIME

The time that requests spent on the optical disk process internal queue (includes any currently active requests).

When reading a request, the optical disk process places it on the internal queue. After processing the request, the optical disk process removes the request from the internal queue. To determine the length of the optical disk process external queue (requests waiting to be read by the optical disk process), measure the optical disk process and examine the RECV-QTIME counter of the PROCESS entity.

Counter type: Queue.

REQUEST-QLEN-MAX

(Legacy Style only) Maximum number of items on the queue described by the REQUEST-QTIME counter.

Counter type: Max queue.

REQUESTS

Number of requests received by the optical disk process.

To determine how long the requests were queued before being read by the optical disk process, measure the optical disk process and examine the RECV-QTIME counter of the PROCESS entity.

Counter type: Incrementing.

READ-BUSY-TIME

(Legacy Style only) The time spent reading from the disk. This counter measures the time from the instruction that starts the read operation to the interrupt that ends the operation. It includes the time spent reading data and positioning the disk heads to read the data.

Counter type: Busy.

WRITE-BUSY-TIME

(Legacy Style only) The time spent writing to the disk. This counter measures the time from the instruction that starts the write operation to the interrupt that ends the operation. This counter includes the time spent writing data and positioning the disk heads to write the data.

Counter type: Busy.

SEEK-BUSY-TIME

(Legacy Style only) No longer used; returns zeros. All seek operations are implicit in read and write operations.

READS

Number of read operations from device to memory performed by the optical disk process. In addition to programmatic read operations, internal operations such as reading file labels also modify this counter.

Counter type: Incrementing.

WRITES

Number of write operations from memory to device performed by the optical disk process. In addition to programmatic write operations, internal operations such as writing volume labels also modify this counter.

Counter type: Incrementing.

SEEKS

(Legacy Style only) No longer used; returns zeros. All seek operations are implicit in read and write operations.

INPUT-BYTES

Number of bytes read from the optical disk. In addition to programmatic read operations, internal operations such as reading file labels also modify this counter.

Because the optical disk process modifies this counter before the I/O operation, this counter might not be accurate if the read fails.

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the optical disk. In addition to programmatic write operations, internal operations such as writing volume labels also modify this counter.

Because the optical disk process modifies this counter before the I/O operation, the counter might not be accurate if the write fails.

Counter type: Accumulating.

CHANNEL

(Legacy Style only) Channel number of the device.

Usage Notes for All OPDISK Entities

- OPDISK does not measure optical disk volumes that have had no activity even if the REPORT ZERO-REPORTS attribute is set to INCLUDE.
- SUBSYSTEM-VERSION for ZMSODISK records are reported as zero.

OSSCPU

The OSSCPU entity provides information about the Open System Services elements that operate in each processor of a system, including:

- POSIX extended segment (PXS)
- OSS file system cache
- File manager
- Pipe pool
- Pipe server

OSSCPU entities can only be used on systems running the Measure G10 PVU or later.

Entity Specification Syntax for OSSCPU Entities

To describe an OSSCPU entity:

```
ADD OSSCPU osscpu-spec
```

osscpu-spec

is specified as:

```
{ * }
{ cpu } [ , cpu ] ...
```

*

measures the OSS elements for all processes on the local system.

cpu

is the CPU on which the OSS elements be measured exist. Use a comma-separated list to specify multiple CPU numbers. Use an asterisk (*) to indicate all CPUs. The default is all CPUs.

DDL Record for OSSCPU Entities (Legacy Style)

This is the Legacy Style DDL record for OSSCPU entities. This record will not change after the G10 Measure PVU.

Fields included in BRIEF reports are in **boldface type**.

```
RECORD osscpu.  FILE is "osscpu" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
 *Entity Identification items:
 02 PXS-bytes-allowed          type binary 64.
 02 Cache-block-size           type binary 32 unsigned.
 02 Cache-blocks-allowed       type binary 32 unsigned.
 02 PP-block-size              type binary 32 unsigned.
 02 PP-blocks-allowed          type binary 32 unsigned.

 * Counter value items:
 02 PXS-ending-bytes           type binary 64.
 02 PXS-failures               type binary 32 unsigned.
 02 FS-direct-reads            type binary 64.
 02 FS-direct-read-bytes       type binary 64.
 02 FS-direct-writes           type binary 64.
 02 FS-direct-write-bytes      type binary 64.
 02 FS-cache-reads             type binary 64.
 02 FS-cache-read-bytes        type binary 64.
 02 FS-cache-writes            type binary 64.
 02 FS-cache-write-bytes       type binary 64.
 02 FS-cache-valid-qtime       type binary 64.
 02 FS-cache-valid-qmax        type binary 16 unsigned.
 02 FS-cache-active-qtime      type binary 64.
 02 FS-cache-active-qmax       type binary 16 unsigned.
 02 FS-cache-dirty-qtime       type binary 64.
 02 FS-cache-dirty-qmax        type binary 16 unsigned.
 02 FS-cache-RD-read-reqs      type binary 64.
 02 FS-cache-WT-read-reqs      type binary 64.
 02 FS-cache-read-blocks       type binary 64.
 02 FS-cache-write-reqs        type binary 64.
 02 FS-cache-write-blocks      type binary 64.
 02 FS-prefetch-blocks         type binary 64.
 02 FS-prefetch-used           type binary 64.
 02 FS-reread-stolen-blks      type binary 64.
 02 FS-map-failures            type binary 32 unsigned.
 02 FM-cache-infos             type binary 32 unsigned.
 02 FM-callbacks               type binary 32 unsigned.
```

```
 02 FM-callback-WT-reqs          type binary 32 unsigned.
 02 FM-callback-WT-blocks        type binary 64.
 02 FM-stolen-blk-WT-reqs        type binary 64.
 02 PP-block-inuse-qtime         type binary 64.
 02 PP-block-inuse-qmax          type binary 16 unsigned.
 02 PP-alloc-failures            type binary 32 unsigned.
 02 PS-gettime-reqs              type binary 32 unsigned.
 02 PS-settime-reqs              type binary 32 unsigned.
 02 LCL                          occurs 16 times.
  03 PS-proxy-reads              type binary 32 unsigned.
  03 PS-proxy-writes             type binary 32 unsigned.
  03 PS-proxy-read-bytes         type binary 64.
  03 PS-proxy-write-bytes        type binary 64.
  03 PS-select-rcvd              type binary 32 unsigned.
  03 PS-select-sent              type binary 32 unsigned.
  03 PS-select-ready             type binary 32 unsigned.
 02 REM                          occurs 1 times.
  03 PS-proxy-reads              type binary 32 unsigned.
  03 PS-proxy-writes             type binary 32 unsigned.
  03 PS-proxy-read-bytes         type binary 64.
  03 PS-proxy-write-bytes        type binary 64.
  03 PS-select-rcvd              type binary 32 unsigned.
  03 PS-select-sent              type binary 32 unsigned.
  03 PS-select-ready             type binary 32 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for OSSCPU Entities (ZMS Style)

The ZMS style DDL record for OSSCPU entities is supported in Measure G11 and later PVUs.

Fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsosscp-id.
 02 PXS-bytes-allowed            type binary 64.
 02 Cache-block-size             type binary 32 unsigned.
 02 Cache-blocks-allowed         type binary 32 unsigned.
 02 PP-block-size                type binary 32 unsigned.
 02 PP-blocks-allowed            type binary 32 unsigned.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsosscp-ctrs.
 02 PXS-ending-bytes             type binary 64.
 02 PXS-failures                 type binary 64.
 02 FS-direct-reads              type binary 64.
 02 FS-direct-read-bytes         type binary 64.
 02 FS-direct-writes             type binary 64.
 02 FS-direct-write-bytes        type binary 64.
 02 FS-cache-reads               type binary 64.
 02 FS-cache-read-bytes          type binary 64.
 02 FS-cache-writes              type binary 64.
 02 FS-cache-write-bytes         type binary 64.
 02 FS-cache-valid-qtime         type binary 64.
 02 FS-cache-active-qtime        type binary 64.
 02 FS-cache-dirty-qtime         type binary 64.
 02 FS-cache-RD-read-reqs        type binary 64.
 02 FS-cache-WT-read-reqs        type binary 64.
 02 FS-cache-read-blocks         type binary 64.
 02 FS-cache-write-reqs          type binary 64.
 02 FS-cache-write-blocks        type binary 64.
```

```
 02 FS-prefetch-blocks          type binary 64.
 02 FS-prefetch-used            type binary 64.
 02 FS-reread-stolen-blks       type binary 64.
 02 FS-map-failures             type binary 64.
 02 FM-cache-infos              type binary 64.
 02 FM-callbacks                type binary 64.
 02 FM-callback-WT-reqs         type binary 64.
 02 FM-callback-WT-blocks       type binary 64.
 02 FM-stolen-blk-WT-reqs       type binary 64.
 02 PP-block-inuse-qtime        type binary 64.
 02 PP-alloc-failures           type binary 64.
 02 PS-gettime-reqs             type binary 64.
 02 PS-settime-reqs             type binary 64.
 02 LCL                         occurs 16 times.
  03 PS-proxy-reads            type binary 64.
  03 PS-proxy-writes           type binary 64.
  03 PS-proxy-read-bytes       type binary 64.
  03 PS-proxy-write-bytes      type binary 64.
  03 PS-select-rcvd            type binary 64.
  03 PS-select-sent            type binary 64.
  03 PS-select-ready           type binary 64.
  03 LS-sends                  type binary 64.
  03 LS-recvs                  type binary 64.
  03 LS-queues                 type binary 64.
  03 LS-send-bytes             type binary 64.
  03 LS-recv-bytes             type binary 64.
  03 LS-queue-bytes            type binary 64.
  03 LS-awakes                 type binary 64.
  03 LS-selects                type binary 64.
 02 REM                         occurs 1 times.
  03 PS-proxy-reads            type binary 64.
  03 PS-proxy-writes           type binary 64.
  03 PS-proxy-read-bytes       type binary 64.
  03 PS-proxy-write-bytes      type binary 64.
  03 PS-select-rcvd            type binary 64.
  03 PS-select-sent            type binary 64.
  03 PS-select-ready           type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsosscp. FILE is "zmsosscp" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsosscp-ctrs.
 02 id                          type zmsosscp-id.
end
```

PXS-BYTES-ALLOWED

The maximum size in bytes allowed for the POSIX extended segment (PXS).

CACHE-BLOCK-SIZE

The size of an OSS file-system cache block in bytes.

CACHE-BLOCKS-ALLOWED

The max number of blocks configured for the OSS file-system cache.

PP-BLOCK-SIZE

The size of an OSS pipe pool block in bytes.

PP-BLOCKS-ALLOWED

The maximum number of blocks allowed for the OSS pipe pool.

PXS-ENDING-BYTES

The current number of bytes allocated within the POSIX extended segment (PXS).

Counter type: Snapshot.

PXS-FAILURES

The number of allocation failures encountered for the POSIX extension segment (PXS). An allocation failure can occur because of a lack of free space or a lack of contiguous free space. If the problem persists, user applications might need to be migrated to another CPU.

Counter type: Incrementing.

FS-DIRECT-READS

The number of OSS file system noncached disk read requests to DP2.

Counter type: Incrementing.

FS-DIRECT-READ-BYTES

The total number of bytes read by FS-DIRECT-READS on the CPU.

Counter type: Accumulating.

FS-DIRECT-WRITES

The number of OSS file system noncached write requests to DP2.

Counter type: Incrementing.

FS-DIRECT-WRITE-BYTES

The number of bytes written by FS-DIRECT-WRITES without using OSS file-system cache.

Counter type: Accumulating.

FS-CACHE-READS

The number of application reads from the OSS file-system cache occurring on the CPU.

Counter type: Incrementing.

FS-CACHE-READ-BYTES

The number of bytes read from FS-CACHE-READS.

Counter type: Accumulating.

FS-CACHE-WRITES

The number of application writes processed using data written to the OSS file-system cache.

Counter type: Incrementing.

FS-CACHE-WRITE-BYTES

The number of bytes written in FS-CACHE-WRITES.

Counter type: Accumulating.

FS-CACHE-VALID-QTIME

The amount of time the OSS file-system cache blocks are valid. Valid cache for open files includes active cache blocks and inactive (unowned) file cache blocks for recently closed files. Valid and active cache counters do not reflect cache blocks stolen by the NonStop memory manager. See FS-REREAD-STOLEN-BLKS (page 262).

Counter type: Queue.

FS-CACHE-VALID-QMAX

(Measure G10 PVU only) The maximum value for cache blocks valid.

Counter type: Max Queue.

FS-CACHE-ACTIVE-QTIME

The amount of time OSS file-system cache blocks were present for an open file.

Counter type: Queue.

FS-CACHE-ACTIVE-QMAX

(Measure G10 PVU only) The maximum number for active cache blocks in use.

Counter type: Max Queue.

FS-CACHE-DIRTY-QTIME

The amount of time OSS file-system cache blocks were dirty.

Counter type: Queue.

FS-CACHE-DIRTY-QMAX

(Measure G10 PVU only) The maximum value for dirty cache blocks.

Counter type: Max Queue.

FS-CACHE-RD-READ-REQS

The number of block read requests to DP2 from the OSS file-system cache to satisfy an application read API call. A block read logical operation can return multiple blocks.

Counter type: Incrementing.

FS-CACHE-WT-READ-REQS

The number of block read requests to DP2 from the OSS file-system cache to satisfy an application write API call. A block read logical operation can return multiple blocks.

Counter type: Incrementing.

FS-CACHE-READ-BLOCKS

The number of blocks read in block reads from DP2 to the OSS file-system cache.

Counter type: Accumulating.

FS-CACHE-WRITE-REQS

The number of block writes to DP2 from the OSS file-system cache.

Counter type: Incrementing.

FS-CACHE-WRITE-BLOCKS

The number of blocks written to DP2 due to FS-CACHE-WRITE-REQS from the OSS file-system cache.

Counter type: Accumulating.

FS-PREFETCH-BLOCKS

The number of blocks prefetched by the OSS file-system cache.

Counter type: Incrementing.

FS-PREFETCH-USED

The number of prefetched blocks actually used by the application.

Counter type: Incrementing.

FS-REREAD-STOLEN-BLKS

The number of data blocks that had to be reread into the OSS file system cache because the NonStop memory manager took the memory pages on which those cache data blocks were located.

Counter type: Incrementing.

FS-MAP-FAILURES

The number of times the OSS file-system cache failed to obtain a map buffer for prefetching cache blocks. The map buffer is a cache I/O buffer that is used for I/O transfers with DP2. A map failure can be a symptom of performance degradation. To remedy the failure, move applications.

Counter type: Incrementing.

FM-CACHE-INFOS

The number of OSS file manager cache info requests received from DP2. The OSS file manager allocates PXS and disk cache segments and acts as an agent for DP2 and the NonStop memory manager for cache management requests.

Counter type: Incrementing.

FM-CALLBACKS

The number of cache callback requests received by the OSS File Manager from DP2.

Counter type: Incrementing.

FM-CALLBACK-WT-REQS

The number of cache write requests issued by the OSS file manager because of DP2 cache callbacks.

Counter type: Incrementing.

FM-CALLBACK-WT-BLOCKS

The number of blocks written back by the OSS file manager as a result of FM-CALLBACK-WT-REQS.

Counter type: Accumulating.

FM-STOLEN-BLK-WT-REQS

The number of times OSS cache block writes were performed to clean dirty memory pages so those memory pages could be stolen by the NonStop memory manager.

Counter type: Incrementing.

PP-BLOCK-INUSE-QTIME

The amount of time the OSS pipe pool blocks were in use.

Counter type: Queue.

PP-BLOCK-INUSE-QMAX

(Measure G10 PVU only) The maximum number of blocks in use by the OSS pipe pool.

Counter type: Max Queue.

PP-ALLOC-FAILURES

The number of times an OSS pipe could not be resized or allocated.

Counter type: Incrementing.

PS-GETTIME-REQS

The number of gettime requests received by an OSS pipe server process from an OSS name server.

Counter type: Incrementing.

PS-SETTIME-REQS

The number of settime requests received by an OSS pipe server process from an OSS name server.

Counter type: Incrementing.

LCL, REM

OSS pipe server requests from CPU-remote and system-remote pipe server processes. Measurement is divided into seven counters:

PS-PROXY-READS

PS-PROXY-WRITES

PS-PROXY-READ-BYTES

PS-PROXY-WRITE-BYTES

PS-SELECT-RCVD

PS-SELECT-SENT

PS-SELECT-READY

LS-SENDS

LS-SEND-BYTES

LS-RECVS

LS-RECV-BYTES

LS-QUEUES

LS-QUEUE-BYTES

LS-AWAKES

LS-SELECTS

These counters occur 17 times, once for each CPU in the system and once for remote systems.

PS-PROXY-READS

The number of proxy reads performed by an OSS pipe server process.

Counter type: Incrementing.

PS-PROXY-WRITES

The number of proxy writes performed by an OSS pipe server process.

Counter type: Incrementing.

PS-PROXY-READ-BYTES

The number of proxy read bytes read by an OSS pipe server process.

Counter type: Accumulating.

PS-PROXY-WRITE-BYTES

The number of proxy write bytes written by an OSS pipe server process.

Counter type: Accumulating.

PS-SELECT-RCVD

The number of select requests received by an OSS pipe server process from other CPUs or systems.

Counter type: Incrementing.

PS-SELECT-SENT

The number of select ready requests sent to an OSS pipe server process from other CPUs or systems.

Counter type: Incrementing.

PS-SELECT-READY

The number of select ready requests received by an OSS pipe server process from other CPUs or systems.

Counter type: Incrementing.

LS-SENDS

The number of SEND messages performed by the LS2 process on behalf of satellite sockets in CPU *n* where *n* is the LCL array index. Socket APIs,`write()`, `send()`, `sendto()` and `sendmsg()`, update this counter when the client is using a satellite socket (the socket is mastered in a different CPU).

NOTE: Processes can be spawned in other CPUs where the descendent inherits the file descriptors (opened files) of the parent. A master (or original) socket is one created via a call to socket(), socketpair() or accept(). A satellite socket is created when a process that has created a socket spawns a process in another CPU.

Counter type: Incrementing.

LS-SEND-BYTES

The number of bytes received in SEND requests from satellite sockets in CPU *n* where *n* is the LCL array index. Socket APIs,`write()`, `send()`, `sendto()` and `sendmsg()`, update this counter when the client is using a satellite socket (the socket is mastered in a different CPU). (See Note under LS-SENDS (page 264).)

Counter type: Accumulating.

LS-RECVS

The number of RECEIVE operations performed by the LS2 process on behalf of satellite sockets in CPU *n* where *n* is the LCL array index. Socket APIs,`read()`, `recv()`, `recvfrom()` and `recvmsg()`, update this counter when the client is using a satellite socket (the socket is mastered in a different CPU). (See Note under LS-SENDS (page 264).)

Counter type: Incrementing.

LS-RECV-BYTES

The number of bytes returned in response to RECEIVE requests from CPU *n* where *n* is the LCL array index. Socket APIs,`read()`, `recv()`, `recvfrom()` and `recvmsg()`, update this counter when the client is using a satellite socket (the socket is mastered in a different CPU). (See Note under LS-SENDS (page 264).)

Counter type: Accumulating.

LS-QUEUES

The number of QUEUE operations performed by the LS2 process. QUEUE requests may be received from socket clients or they may be received from LS2 processes in CPU *n* where *n* is the LCL array index. Socket APIs,`write()`, `send()`, `sendto()` and `sendmsg()`, update this counter when the sending and receiving sockets are mastered in different CPUs. (See Note under LS-SENDS (page 264).)

Counter type: Incrementing.

LS-QUEUE-BYTES

The number of bytes received in QUEUE requests from CPU *n* where *n* is the LCL array index. Socket APIs,`write()`, `send()`, `sendto()` and `sendmsg()`, update this counter when the sending and receiving sockets are mastered in different CPUs. (See Note under LS-SENDS (page 264).)

Counter type: Accumulating.

LS-AWAKES

The number of AWAKE operations performed on behalf of processes in CPU *n* where *n* is the LCL array index. Socket APIs,`connect()`, `accept()`, `write()`, `send()`, `sendto()`, `sendmsg()`, `read()`, `recv()`, `recvfrom()`, `recvmsg()`, `shutdown()`, and `close()` update this counter.

Counter type: Incrementing.

LS-SELECTS

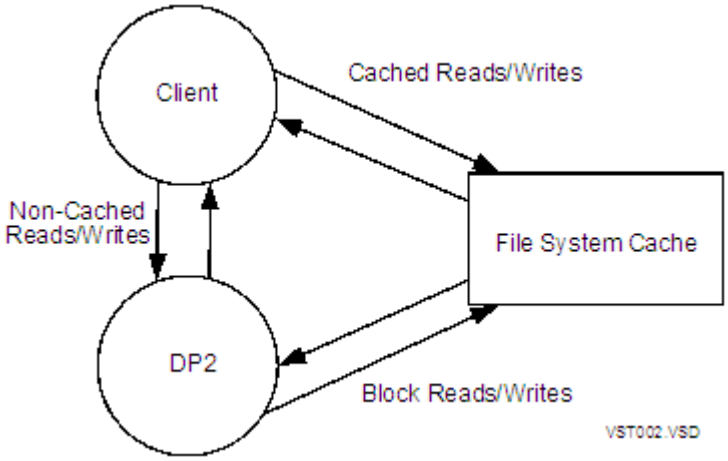
The number of SELECT operations performed by the LS2 process to CPU *n* where *n* is the LCL array index. Socket APIs,`select()`, update this counter. In addition, the procedures, `FILE_COMPLETE_()` and `FILE_COMPLETE_SET_()` update this counter.

Counter type: Incrementing.

Usage Notes for OSSCPU Entities

- A read or write operation on a regular file can be sent directly to DP2 or satisfied via the OSS File System cache.
  - Noncached operations (FS-DIRECT-READS, FS-DIRECT-WRITES) result in messages to DP2 and transfer only as much information as specified by the client.
  - Cached operations (FS-CACHE-READS, FS-CACHE-WRITES) utilize the File System Cache in the client's CPU and can result in a block read/write (FS-CACHE-RD-READ-REQS, FS-CACHE-WT-READ-REQS, FS-CACHE-WRITE-REQS) between the cache and DP2.

    Block reads and writes occur on cache block boundaries (except for the last cache block of the file) where a cache block is 4 KB. Block read operations try to read ahead (FS-PREFETCH-BLOCKS) to anticipate any need for the information in the immediate future. The OSS file system controls cache use and might, for internal reasons, select to bypass its use on a particular file open.



- When a pipe is remote from the sender or receiver, the pipe server in the CPU where the pipe was created acts as an agent for the remote clients. Write and read requests result in messages to the pipe server in the CPU where the pipe was created. The pipe server, in turn, performs proxy pipe reads and writes to the pipe buffer on behalf of the remote client.
- Select() messages between client and pipe server and between pipe servers, occur only when the client calling select() is running in a CPU other than where the pipe was created:
  1. The client sends a select request to the pipe server that manages the pipe.
  2. The pipe server replies immediately.
  3. When the client's read, write, or both can succeed, the pipe server sends a select ready request to the pipe server in the client's CPU.
  4. That pipe server awakens the client, and the select() call completes.
- You can calculate the OSS cache miss rate as:

```
OSS cache miss rate = (FS-Cache-RD-Read-Reqs +
                       FS-Cache-WT-Read-Reqs +
                       FS-Cache-Write-Reqs)

-------------------------------------------------------------

(FS-Cache-Reads + FS-Cache-Writes)
```

OSSNS

The OSSNS entity provides information about the operation and performance of OSS Name Server processes. There is one record for the primary and backup processes configured for each OSS Name Server defined on a system. Any OSS operation involving an OSS file pathname uses the Name Server.

OSSNS entities can only be used on systems running the Measure G10 PVU or later.

Entity Specification Syntax for OSSNS Entities

To describe an OSSNS entity:

```
ADD OSSNS ossns-spec [, ossns-spec ] ...
```

ossns-spec

is specified as:

```
{ *                                                    }
{ pid                                                  }
{ $process-name [(pid)]                                }
{ [[$device.]subvolume.]filename[(pid)]   }
```

*

measures the OSS elements for all processes on the local system.

pid

is the CPU on which the OSS elements to be measured exist. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

process-name

is the name of a specific process to measure.

DDL Record for OSSNS Entities (Legacy Style)

This is the Legacy Style DDL record for OSSNS entities. This record will not change after the G10 Measure PVU.

Fields included in BRIEF reports are in **boldface type**.

```
RECORD ossns.  FILE is "ossns" ENTRY-SEQUENCED.
                       .
                       .
                       .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                       .
                       .
                       .
 * Entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                    type character 8.
 02 program-file-name.
    03 volume                       type character 8.
    03 subvol                       type character 8.
    03 filename                     type character 8.
 02 IC-entries                    type binary 32 unsigned.
 02 LC-entries                    type binary 32 unsigned.

 *Counter value items:
 02 RR-processed                  type binary 32 unsigned.
 02 RR-redir-sent                 type binary 32 unsigned.
 02 RR-redir-processed            type binary 32 unsigned.
 02 IC-lookups                    type binary 32 unsigned.
 02 IC-hits                       type binary 32 unsigned.
 02 IC-dirty-qtime                type binary 64.
 02 LC-lookups                    type binary 32 unsigned.
 02 LC-hits                       type binary 32 unsigned.
 02 LC-inuse-qtime                type binary 64.
 02 Checkpoint-reqs               type binary 32 unsigned.
 02 Checkpoint-blks               type binary 32 unsigned.
 02 Sem-waits                     type binary 32 unsigned.
 02 Sem-wait-qtime                type binary 64.
 02 Sem-wait-qlen-max             type binary 16 unsigned.
 02 DP2-DD-reqs                   type binary 32 unsigned.
 02 DP2-messages                  type binary 32 unsigned.
 02 DP2-msg-qtime                 type binary 64.
 02 DP2-msg-qlen-max              type binary 16 unsigned.
 02 PS-messages                   type binary 32 unsigned.
```

```
 02 PS-msg-qtime                type binary 64.
 02 PS-msg-qlen-max             type binary 16 unsigned.
 02 LS-messages                 type binary 32 unsigned.
 02 LS-msg-qtime                type binary 64.
 02 LS-msg-qlen-max             type binary 16 unsigned.
 02 gettime-reqs                type binary 32 unsigned.
 02 settime-reqs                type binary 32 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for OSSNS Entities (ZMS Style)

The ZMS style DDL record for OSSNS entities is supported in Measure G11 and later PVUs.

Fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsossns-id.
 02 pin                         type binary 16 unsigned.
 02 reserved-1                  type character 6.
 02 process-name                type character 8.
 02 program-file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 IC-entries                  type binary 32 unsigned.
 02 LC-entries                  type binary 32 unsigned.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsossns-ctrs.
 02 RR-processed                type binary 64.
 02 RR-redir-sent               type binary 64.
 02 RR-redir-processed          type binary 64.
 02 IC-lookups                  type binary 64.
 02 IC-hits                     type binary 64.
 02 IC-dirty-qtime              type binary 64.
 02 LC-lookups                  type binary 64.
 02 LC-hits                     type binary 64.
 02 LC-inuse-qtime              type binary 64.
 02 Checkpoint-reqs             type binary 64.
 02 Checkpoint-blks             type binary 64.
 02 sem-waits                   type binary 64.
 02 sem-wait-qtime              type binary 64.
 02 DP2-DD-reqs                 type binary 64.
 02 DP2-messages                type binary 64.
 02 DP2-msg-qtime               type binary 64.
 02 PS-messages                 type binary 64.
 02 PS-msg-qtime                type binary 64.
 02 LS-messages                 type binary 64.
 02 LS-msg-qtime                type binary 64.
 02 gettime-reqs                type binary 64.
 02 settime-reqs                type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsossns. FILE is "zmsossns" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsossns-ctrs.
02 id                           type zmsossns-id.
end
```

IC-ENTRIES

The configured number of inode cache records available on an OSS name server.

Counter type: Snapshot.

LC-ENTRIES

The configured number of link cache records available on an OSS name server.

Counter type: Snapshot.

RR-PROCESSED

The number of name resolution requests processed by the OSS name server under measurement.

Counter type: Incrementing.

RR-REDIR-SENT

The number of name resolution requests redirected by an OSS name server.

Counter type: Incrementing.

RR-REDIR-PROCESSED

The number of redirected name resolution requests processed by an OSS name server.

Counter type: Incrementing.

IC-LOOKUPS

The number of inode cache lookups performed by an OSS name server.

Counter type: Incrementing.

IC-HITS

The number of inode cache hits encountered by an OSS name server. To derive inode cache misses, subtract IC-HITS from IC-LOOKUPS.

Counter type: Incrementing.

IC-DIRTY-QTIME

The amount of time inode cache entries were dirty on an OSS name server.

Counter type: Queue.

LC-LOOKUPS

The number of link cache lookups performed by an OSS name server.

Counter type: Incrementing.

LC-HITS

The number of link cache hits encountered by an OSS name server. Link cache misses can be derived by subtracting LC-HITS from LC-LOOKUPS.

Counter type: Incrementing.

LC-INUSE-QTIME

The amount of time link cache entries were in use on an OSS name server. Link cache expands beyond the configured number of entries if needed. If expanded, the name server process eventually scales link cache resources back to the configured size when demand lessens.

Counter type: Queue.

CHECKPOINT-REQS

The number of logical checkpoint requests sent by an OSS name server. Checkpoint requests are assembled into blocks in the primary so that multiple requests are sent to the backup in each message.

Counter type: Incrementing.

CHECKPOINT-BLKS

The number of checkpoint blocks sent by an OSS name server to its backup.

Counter type: Incrementing.

SEM-WAITS

The number of semaphore waits an OSS name server encounters in its processing threads.

Counter type: Incrementing.

SEM-WAIT-QTIME

The amount of time spent in a semaphore wait condition by OSS name server processing threads. A name server is a multithreaded process. A semaphore wait is specific to a thread processing a name resolution request.

Counter type: Queue.

SEM-WAIT-QLEN-MAX

(Measure G10 PVU only) The maximum number of resolution request semaphore waits on an OSS name server since it began execution.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max Queue.

DP2-DD-REQS

The number of data definition requests sent to DP2 by an OSS name server. Data definition operations are initiated by these procedures:

creat()	link()	mkdir()	chmod()	utime()	bind()
open()	unlink()	rmdir()	chown()	rename()	

Counter type: Incrementing.

DP2-MESSAGES

The number of messages an OSS name server sent to DP2. The total includes Data Definition calls, which modify the file label, as well as other calls that do not alter the disk entry.

Counter type: Incrementing.

DP2-MSG-QTIME

The accumulated service time spent in a thread waiting while DP2 messages were outstanding on an OSS name server.

Counter type: Queue.

DP2-MSG-QLEN-MAX

(Measure G10 PVU only) The maximum number of DP2 messages outstanding for an OSS name server since it began execution.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max Queue.

PS-MESSAGES

The number of messages an OSS name server sent to pipe server processes. The pipe server runs in each CPU ($ZPP*nn*, where *nn* = CPU number) and provides pipe services for applications using OSS pipes and FIFOs.

Counter type: Incrementing.

PS-MSG-QTIME

The amount of time pipe server messages from an OSS name server were outstanding.

Counter type: Queue.

PS-MSG-QLEN-MAX

(Measure G10 PVU only) The maximum number of concurrent pipe server messages outstanding on an OSS name server since it began execution.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max Queue.

LS-MESSAGES

The number of messages an OSS name server sent to the local server process. The OSS local server ($ZPLS for pre-AF_UNIX R2; $ZLS*nn*, where *nn* is the CPU number, or $ZLSNN for AF_UNIX R2) is the transport provider for AF_UNIX sockets.

Counter type: Incrementing.

LS-MSG-QTIME

The amount of time spent waiting on completion of local server messages by an OSS name server.

Counter type: Queue.

LS-MSG-QLEN-MAX

(Measure G10 PVU only) The maximum number of concurrent local server messages outstanding on an OSS name server since it began execution.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max Queue.

GETTIME-REQS

The number of gettime requests sent by the OSS name server to a pipe server or local server.

Counter type: Incrementing.

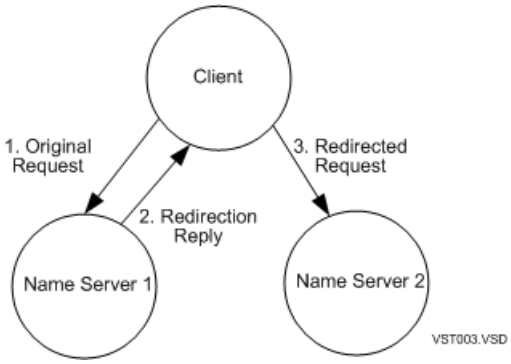
SETTIME-REQS

The number of settime requests sent by the OSS name server to a pipe server or local server.

Counter type: Incrementing.

Usage Notes for OSSNS Entities

- Fields that report received or redirected resolution requests reflect the capacity of an OSS name server to route requests to the appropriate resource. When a request is sent to a name server, the name server might be able to completely process the request, or it might require the services of another name server. In the latter case, the first name server replies to the client with a redirection reply, which causes the file-system library running in the client to compose and send a new redirected request to the second name server.



- Inode and link cache counters report on cache performance for OSS Fileset Catalog accesses. The name server OSS Fileset Catalog consists of three Guardian disk files:

PXINODE	A key-sequenced file containing one record for each directory, regular file, FIFO, and AF_UNIX socket.
PXLINK	A key-sequenced file containing one record for each file name in the fileset.
PXLOG	An unstructured file used to ensure catalog integrity after a failure. PXLOG is used only when the fileset is configured with BUFFERED NONE.

To reduce the frequency of access to the PXINODE and PXLINK files, name servers cache records from these files in memory. Separate caches are used for each file. Cache size is reported in IC-ENTRIES and LC-ENTRIES. The size of the caches can be changed individually. Both the primary and backup name server processes have PXINODE caches. Only the primary process has a PXLINK cache. The PXLINK cache not only keeps copies of PXLINK records, it also remembers unsuccessful name lookups. If the name is looked up again, it is not necessary to reread the PXLINK file to discover again that the name does not exist.

PROCESS

The PROCESS entity type provides information about one or more processes on a local system. In Measure G09 and later PVUs, the PROCESS entity type handles OSS file pathnames.

Entity Specification Syntax for PROCESS Entities

To describe a PROCESS entity:

```
PROCESS entity-spec
```

PROCESS

collects information about one or more processes on the local system.

entity-spec

is specified as:

```
{ *                                                    }
{ cpu,pin                                              }
```

```
{ SYSTEM-PROCESSES                                  }
{ $process-name [ ( pid ) ]                         }
{ [[$device.]subvolume.]filename[:CRVSN][ ( pid ) ]}
{ "pname" [ ( pid ) ]                               }
```

*

measures all processes on the local system.

cpu

is the CPU on which the process to be measured is running. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

pin

is the process identification number of the process to be measured. To indicate all processes, use an asterisk (*). The default is all processes.

SYSTEM-PROCESSES

measures all processes installed using the SYSGEN program.

$*process-name*

is the name of the process to be measured.

pid

is the process identifier of the process to be measured. Specify pid as one of:

```
{ cpu,pin          }
{ SYSTEM-PROCESSES }
```

cpu

is the number of the CPU on which the process to be measured is running. Use an asterisk (*) to specify all CPUs.

pin

is the process identification number of the process to be measured. Use an asterisk (*) to specify all process identification numbers.

$*device*

is the volume (device) on which the object file of the process to be measured is located. Use an asterisk (*) to indicate all volumes. The default is the current default volume.

subvolume

is the subvolume in which the object file of the process to be measured is located. Use an asterisk (*) to indicate all subvolumes.

filename[:CRVSN]

is the name of the object file of an executing process to be measured. Use an asterisk (*) to indicate all files (except temporary files).

In Measure G09 and later PVUs, CRVSN is the timestamp, creation version serial number, or file name extension necessary to form a unique file name. To get the CRVSN for a file, see the Measure report or use the LISTGNAME command.

"*pname*"

can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

DDL Record for PROCESS Entities (Legacy Style)

This is the Legacy Style DDL record for PROCESS entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

The DDL record for G-series PROCESS entities is identical to the record for D-series PROCESS entities except:

- Longer byte-count fields are provided to reduce the possibility of field overflow. Each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Notes for G-Series PROCESS Entities (page 291).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.
- Counters are added to support operating system changes in memory handling. The memory handling changes also affect the interpretation of the existing CPU-BUSY-TIME, READY-TIME, and PRES-PAGES-QTIME counters.
- In Measure G09 and later PVUs, several identifiers are added for OSS file pathname support.

```
RECORD process. FILE is "process" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 priority                     type binary 16 unsigned.
* counter value items:
 02 cpu-busy-time                type binary 64.
 02 ready-time                   type binary 64.
 02 mem-qtime                    type binary 64.
 02 dispatches                   type binary 32 unsigned.
 02 page-faults                  type binary 32 unsigned.
 02 pres-pages-qtime             type binary 64.
 02 pres-pages-max               type binary 16 unsigned.
 02 ext-segs-qtime               type binary 64.
 02 ext-segs-max                 type binary 16 unsigned.
 02 vsems                        type binary 32 unsigned.
 02 recv-qtime                   type binary 64.
 02 recv-qlen-max                type binary 16 unsigned.
 02 messages-sent                type binary 32 unsigned.
 02 sent-bytes                   type binary 32 unsigned.
 02 returned-bytes               type binary 32 unsigned.
 02 messages-received            type binary 32 unsigned.
 02 received-bytes               type binary 32 unsigned.
 02 reply-bytes                  type binary 32 unsigned.
 02 lcb-allocations              type binary 32 unsigned.
 02 mqc-allocations              type binary 32 unsigned
```

```
                                redefines lcb-allocations.
 02 lcb-alloc-failures         type binary 32 unsigned.
 02 mqc-alloc-failures         type binary 32 unsigned
                               redefines lcb-alloc-failures.
 02 lcbs-inuse-qtime           type binary 64.
 02 mqcs-inuse-qtime           type binary 64
                               redefines lcbs-inuse-qtime.
 02 max-lcbs-inuse             type binary 16 unsigned.
 02 max-mqcs-inuse             type binary 16 unsigned
                               redefines max-lcbs-inuse.
 02 checkpoints                type binary 32 unsigned.
 * new fields for entity identification:
 02 userid.
    03 group                   type binary 8 unsigned.
    03 user                    type binary 8 unsigned.
 02 creatorid.
    03 group                   type binary 8 unsigned.
    03 user                    type binary 8 unsigned.
* fields for TNS/R specific counters:
 02 accel-busy-time            type binary 64.
 02 tns-busy-time              type binary 64.
 02 comp-traps                 type binary 32 unsigned.
 02 program-accelerated        type binary 16 unsigned.
* new entity identification items for D10
 02 ancestor-cpu               type binary 16 unsigned.
 02 ancestor-pin               type binary 16 unsigned.
 02 ancestor-sysname           type character 8.
 02 ancestor-process-name      type character 8.
* Native Mode busy time:
 02 tnsr-busy-time             type binary 64.
* Native Mode process:
 02 tnsr-process               type binary 16 unsigned.
* new entity identification items and counters for D25
 02 hometerm-sysname           type character 8.
 02 hometerm-name.
    03 device                  type character 8.
    03 subdevice               type character 8.
    03 qualifier               type character 8.

 02 page-size-bytes            type binary 16 unsigned.
 02 alloc-seg-calls            type binary 32 unsigned.

 02 UCL-qtime                  type binary 64.
 02 UCL-max                    type binary 16 unsigned.

 02 UCL-lock-qtime             type binary 64.
 02 UCL-lock-max               type binary 16 unsigned.

02 file-open-calls             type binary 32 unsigned.
02 info-calls                  type binary 32 unsigned.
* D30 changes:
 02 begin-trans                type binary 32 unsigned.
 02 abort-trans                type binary 32 unsigned.

* SMS changes:
 02 device-name                type character 8.

* F40 new counter value items:
 02 sent-bytes-f               type binary 64.
 02 returned-bytes-f           type binary 64.
 02 received-bytes-f           type binary 64.
 02 reply-bytes-f              type binary 64.
* New counters for G04
 02 pres-pages-start           type binary 32 unsigned.
 02 pres-pages-end             type binary 32 unsigned.
```

```
 02 abs-segs-qtime              type binary 64.
 02 abs-segs-qlen-max           type binary 16 unsigned.
 02 abs-segs-start              type binary 32 unsigned.
 02 abs-segs-end                type binary 32 unsigned.
* New counters for G08:
 02 msgs-sent-qtime             type binary 64.
 02 msgs-sent-qlen-max          type binary 16 unsigned.
 02 sent-cbytes                 type binary 64.
 02 returned-cbytes             type binary 64.
 02 received-cbytes             type binary 64.
 02 reply-cbytes                type binary 64.
* New identifiers for OSS file pathname support:
 02 osspid                      type binary 32 unsigned.
 02 program-file-name-MID.
    03 PATHID                   type character 24.
    03 CRVSN                    type character 6.
* New identifiers for NetBatch Job Control:
 02 GMOM.
    03 GMOM-node                type binary 16 unsigned.
    03 GMOM-cpu                 type binary 16 unsigned.
    03 GMOM-pin                 type binary 16 unsigned.
    03 GMOM-jobid               type binary 16 unsigned.
 02 GMOM-full-id                type binary 64
                                     redefines GMOM.
 02 GMOM-sysname                type character 8.
 02 GMOM-process-name           type character 8.
* New counters for G10:

 02 OSS-tty-reads                 type binary 32 unsigned.
 02 OSS-tty-writes                type binary 32 unsigned.
 02 OSS-tty-read-bytes            type binary 64.
 02 OSS-tty-write-bytes           type binary 64.
 02 OSS-tty-wait-time             type binary 64.
 02 OSS-dev-null-ops              type binary 64.
 02 OSSNS-DD-calls                type binary 32 unsigned.
 02 OSSNS-requests                type binary 32 unsigned.
 02 OSSNS-message-bytes           type binary 64.
 02 OSSNS-wait-time               type binary 64.
 02 OSSNS-redirects               type binary 32 unsigned.
 02 launches                      type binary 32 unsigned.
 02 launch-wait-time              type binary 64.
 02 open-close-wait-time          type binary 64.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for PROCESS Entities (ZMS Style)

The ZMS style PROCESS DDL record is supported in Measure G11 and later PVUs.

ID Fields DDL Definition

```
DEFINITION zmsproc-id.
 02 pin                         type binary 16 unsigned.
 02 priority                    type binary 16 unsigned.
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 group                    type binary 8 unsigned.
 02 process-name                type character 8.
 02 program-file-name.
    03 volume                   type character 8.
```

```
    03 subvol                  type character 8.
    03 filename                type character 8.
 02 osspid                     type binary 32 unsigned.
 02 ancestor-cpu               type binary 16 unsigned.
 02 ancestor-pin               type binary 16 unsigned.
 02 ancestor-sysname           type character 8.
 02 ancestor-process-name      type character 8.
 02 device-name                type character 8.
 02 program-file-name-MID.
    03 PATHID                  type character 24.
    03 CRVSN                   type character 6.
 02 reserved-1                 type character 2.
 02 GMOM.
    03 GMOM-node               type binary 16 unsigned.
    03 GMOM-cpu                type binary 16 unsigned.
    03 GMOM-pin                type binary 16 unsigned.
    03 GMOM-jobid              type binary 16 unsigned.
 02 GMOM-full-id               type binary 64
                                    redefines GMOM.
 02 GMOM-sysname               type character 8.
 02 GMOM-process-name          type character 8.
 02 program-accelerated        type binary 16 unsigned.
 02 native-process             type binary 16 unsigned.
 02 system-process             type binary 16 unsigned.
 02 ipus                       type binary 16 unsigned.
 02 reserved-2                 type character 2.
 02 hometerm-sysname           type character 8.
 02 hometerm-name.
    03 device                  type character 8.
    03 subdevice               type character 8.
    03 qualifier               type character 8.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsproc-ctrs.
 02 cpu-busy-time              type binary 64.
 02 ready-time                 type binary 64.
 02 mem-qtime                  type binary 64.
 02 dispatches                 type binary 64.
 02 page-faults                type binary 64.
 02 pres-pages-qtime           type binary 64.
 02 ext-segs-qtime             type binary 64.
 02 vsems                      type binary 64.
 02 recv-qtime                 type binary 64.
 02 messages-sent              type binary 64.
 02 sent-bytes                 type binary 64.
 02 returned-bytes             type binary 64.
 02 messages-received          type binary 64.
 02 received-bytes             type binary 64.
 02 reply-bytes                type binary 64.
 02 mqc-allocations            type binary 64
 02 mqc-alloc-failures         type binary 64
 02 mqcs-inuse-qtime           type binary 64
 02 checkpoints                type binary 64.
 02 comp-traps                 type binary 64.
 02 native-busy-time           type binary 64.
 02 accel-busy-time            type binary 64.
 02 tns-busy-time              type binary 64.
 02 alloc-seg-calls            type binary 64.
 02 file-open-calls            type binary 64.
 02 info-calls                 type binary 64.
 02 begin-trans                type binary 64.
 02 abort-trans                type binary 64.
 02 pres-pages-start           type binary 32 unsigned.
```

```
 02 pres-pages-end              type binary 32 unsigned.
 02 Abs-segs-qtime              type binary 64.
 02 Abs-segs-start              type binary 32 unsigned.
 02 Abs-segs-end                type binary 32 unsigned.
 02 Msgs-sent-qtime             type binary 64.
 02 Sent-cbytes                 type binary 64.
 02 Returned-cbytes             type binary 64.
 02 Received-cbytes             type binary 64.
 02 Reply-cbytes                type binary 64.
 02 OSS-tty-reads               type binary 64.
 02 OSS-tty-writes              type binary 64.
 02 OSS-tty-read-bytes          type binary 64.
 02 OSS-tty-write-bytes         type binary 64.
 02 OSS-tty-wait-time           type binary 64.
 02 OSS-dev-null-ops            type binary 64.
 02 OSSNS-DD-calls              type binary 64.
 02 OSSNS-requests              type binary 64.
 02 OSSNS-message-bytes         type binary 64.
 02 OSSNS-wait-time             type binary 64.
 02 OSSNS-redirects             type binary 64.
 02 launches                    type binary 64.
 02 launch-wait-time            type binary 64.
 02 open-close-wait-time        type binary 64.
 02 ipu-switches                type binary 64.
 02 ipu-num                     type binary 32.
 02 ipu-num-prev                type binary 32.
end
```

DDL Record Description Fields

```
RECORD zmsproc. FILE is "zmsproc" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsproc-ctrs.
 02 id                          type zmsproc-id.
end
```

PIN

Process identification number of the measured process.

PROCESS-NAME

Name of the process.

PROGRAM-FILE-NAME

Object file of the measured process. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, PROGRAM-FILE-NAME represents the physical file name that was specified when the process was opened. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, PROGRAM-FILE-NAME represents a location-independent logical file name that was used when the process was opened. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

PRIORITY

Creation priority of the measured process. Changing a process priority using the ALTPRI command does not affect this value.

CPU-BUSY-TIME

The time that the CPU spent executing the measured process:

- For D-series RVUs, this counter is not advanced for the sending process when LDONE queuing is in use.
- For measurements on systems running G05 or later RVUs, this counter includes any page-defaulting activity performed on behalf of this process.
- For pre-G05.00 G-series RVUs, all page-faulting is charged to the memory manager process $VIRTUAL.

For TNS/R servers, CPU-BUSY-TIME is the sum of ACCEL-BUSY-TIME, TNS-BUSY-TIME, and TNSR-BUSY-TIME. For TNS/E servers, CPU-BUSY-TIME is the sum of ACCEL-BUSY-TIME, TNS-BUSY-TIME, and NATIVE-BUSY-TIME

NOTE: The CPU-BUSY-TIME counter is enhanced for more precise measurement of processor utilization. Measurements on systems running D-series and G-series RVUs might show slightly higher utilization than comparable measurements on systems running C-series RVUs. The difference appears mainly on systems that run at low utilization levels.

Related Counter	Description	Page
CPU CPU-BUSY-TIME	Similar, but is the busy time for the entire processor.	161
PROCESS CPU-NUM	Identifies the CPU executing the process	

Counter type: Busy.

READY-TIME

The time that the measured process spent on the ready list or executing.

To keep the measurement overhead to a minimum, this counter is not turned off during certain system activities. This counter includes:

- Time required to dispatch the process.
- Time spent in software interrupts that are invoked asynchronously to the executing process.
- Time spent waiting for a page fault to be satisfied by the memory manager. (D-series)
- Time spent in message system routines. (D-series)
- Time spent in low-level message system routines when the process receives data or a message and the data transfer is not complete. The process is then suspended without being taken off the ready list timer. For example, a process might remain in the ready timer, not executing, while waiting for completion of a transfer of ServerNet or FOX data. (G-series)

Due to changes in memory handling, the results of the READY-TIME counter for G05 and later are not directly comparable to earlier measurements. Previously, the ready list timer for a process remained active while waiting for a page fault even if the memory manager subsequently stalled waiting on I/O for the needed page. As of G05, each process handles its own page fault requests. If a request stalls while waiting for an I/O completion for the needed page, the ready timer is turned off.

Counter type: Busy.

NOTE: Measure data can exhibit a combination of high PROCESS.READY-TIME and CPU.CPU-QTIME while having low PROCESS.CPU-BUSY-TIME and low overall processor utilization. Prior to G05, this was due to the suspension of PROCESS.CPU-BUSY-TIME while waiting for completion of a page fault. However, in all RVUs, a process can also be suspended while waiting for incomplete message system transfers to attempt completion. In either case, the PROCESS.READY-TIME and CPU.CPU-QTIME counters continue to accumulate. Although this message system behavior is rare, its occurrence might signal congested FOX or ServerNet/FX networks, possibly the result of paths in the network being marked as down, underlying hardware problems in the network, or data overload.

MEM-QTIME

The time that the measured process spent waiting on page faults. A page fault does not cause the system to remove the process from the ready list.

Related Counter	Description	Page
PROCESS PAGE-FAULTS	Counts the number of page faults generated by the measured process	280
CPU MEM-QTIME	Measures the time that all processes in a given processor spent waiting on page faults	162
CPU CPU-QTIME	Measures the time that all processes spent on the ready list, including time spent waiting on page faults	161

Counter type: Queue.

DISPATCHES

Number of times the process was selected from the ready list and executed by the CPU identified by the CPU-NUM field. You can expect one dispatch for each I/O operation, plus additional dispatches caused by the process being preempted by higher priority processes.

Counter type: Incrementing.

PAGE-FAULTS

Number of page faults generated by the process.

Counter type: Incrementing.

PRES-PAGES-QTIME

For D-series RVUs, the time that pages owned by the process spent in main memory.

For G-series RVUs, the time that pages sponsored by the measured process spent in main memory. For a description of sponsoring, see Usage Notes for G-Series PROCESS Entities (page 291).

Counter type: Queue.

PRES-PAGES-MAX

(Legacy Style only) Maximum number of items on the queue described by the PRES-PAGES-QTIME counter.

Counter type: Max queue.

EXT-SEGS-QTIME

The time that extended segments were allocated to the process through the SEGMENT_ALLOCATE_ procedure. This counter includes both user and system calls to this procedure.

The system does not use SEGMENT_ALLOCATE_ when allocating segments for process code and data stack segments.

Counter type: Queue.

EXT-SEGS-MAX

(Legacy Style only) Maximum number of items on the queue described by the EXT-SEGS-QTIME counter.

Counter type: Max queue.

VSEMS

Number of times the process waited for a resource in use by another process, where the resource is protected by the semaphore facility.

Counter type: Incrementing.

RECV-QTIME

The time that messages spent waiting on the process message input queue.

For user processes, the message input queue is $RECEIVE.

Counter type: Queue.

RECV-QLEN-MAX

(Legacy Style only) Maximum number of items on the queue described by the RECV-QTIME counter.

Counter type: Max queue.

MESSAGES-SENT

Number of messages sent by the linker process. User I/O operations (except on $RECEIVE) and calls to some system procedures increment this counter.

Counter type: Incrementing.

SENT-BYTES

Number of message bytes sent by the linker process. The message bytes are also included in the RECEIVED-BYTES counter of the listener process that received the message.

This counter does not include bytes transferred by direct bulk I/O. For DBIO byte counts, see the measurements attributed to the linker process in the DBIO-WRITE-BYTES counter of the FILE entity type.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

The G-series SENT-BYTES-F field is a 64-bit version of SENT-BYTES.

Counter type: Accumulating.

RETURNED-BYTES

Number of message bytes received by the linker process. The message bytes are also included in the REPLY-BYTES counter of the listener process that sent the message.

This counter does not include bytes transferred by direct bulk I/O. For DBIO byte counts, see the measurements attributed to the linker process in the DBIO-READ-BYTES counter of the FILE entity type.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

The G-series RETURNED-BYTES-F field is a 64-bit version of RETURNED-BYTES.

Counter type: Accumulating.

MESSAGES-RECEIVED

Number of messages the process has read from its message input queue.

For user processes, the message input queue is $RECEIVE.

These operations do not advance this counter:

- Messages received by a process due to SIGNALTIMEOUT and SIGNALPROCESSTIMEOUT
- The sending process when LDONE queuing is in use
- Timeout notifications (TLEs)

The counter is not advanced for the sending process when LDONE queuing is in use, nor do timeout notifications (TLEs) cause the MESSAGES-RECEIVED counter to be advanced.

Counter type: Incrementing.

RECEIVED-BYTES

Number of message bytes received by the listener process. The message bytes are also included in the SENT-BYTES counter of the linker process that sent the message.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

The G-series RECEIVED-BYTES-F field is a 64-bit version of RECEIVED-BYTES.

Counter type: Accumulating.

REPLY-BYTES

Number of message bytes sent by a listener process in reply to a message from a linker process. The message bytes are also included in the RETURNED-BYTES counter of the linker process that received the message.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, this is a 64-bit counter.

The G-series REPLY-BYTES-F field is a 64-bit version of REPLY-BYTES.

Counter type: Accumulating.

LBC-ALLOCATIONS

Redefined. See MQC-ALLOCATIONS.

MQC-ALLOCATIONS

Number of MQCs currently allocated to the process.

Counter type: Incrementing.

LCB-ALLOC-FAILURES

Redefined. See MQC-ALLOC-FAILURES.

MQC-ALLOC-FAILURES

Number of MQC requests that failed.

Counter type: Incrementing.

LCBS-INUSE-QTIME

Redefined. See MQCS-INUSE-QTIME.

MQCS-INUSE-QTIME

The time for which MQCs have been allocated to the process.

Counter type: Queue.

MAX-LCBS-INUSE

(Legacy Style only) Redefined. See MAX-MQCS-INUSE.

MAX-MQCS-INUSE

(Legacy Style only) Maximum number of MQCs on the queue.

Counter type: Max queue.

CHECKPOINTS

Number of calls to the CHECKPOINT procedure. This counter includes both user and system calls to this procedure.

Many system processes do their own checking rather than calling the CHECKPOINT procedure. These processes do not increment this counter.

Counter type: Incrementing.

USERID

Process accessor ID (PAID) of the process.

CREATORID

Creator access ID (CAID). This field identifies the creator of the process. It is divided into two subfields: GROUP and USER.

ACCEL-BUSY-TIME

The time that the CPU was busy executing accelerated code in this process. This counter applies only to measurements taken on a TNS/R or TNS/E system.

For H-series, J-series, and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter of the CPU record reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy.

NATIVE-BUSY-TIME

(ZMS style only) The time that the CPU was busy executing native code in this process.

For H-series, J-series, and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter of the CPU record reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy

TNS-BUSY-TIME

The time that the CPU was busy executing TNS code in this process. This counter applies only to measurements taken on a TNS/R or TNS/E system.

For H-series, J-series, and later RVUs, this counter value is calculated, returned, and displayed only if a PROCESSH measurement is active on the CPU. Otherwise, the counter is not displayed, and its value in the returned record is zero. The counter is calculated based on the number of PROCESSH samples observed in the applicable code region, and so for more accurate numbers, you can increase the frequency of PROCESSH samples. The PROCESSH-SAMPLES counter of the CPU record reports the current sampling frequency.

Effective with Measure H04, J02, and later PVUs, if the PROCESSH sample count is unchanged from the start to the end of a measurement interval, the ACCEL-BUSY-TIME, NATIVE-BUSY-TIME, and TNS-BUSY-TIME fields will not be displayed by MEASCOM even if a PROCESSH measurement is active. For the LISTALL command, this means some intervals might display those fields and others might not.

Counter type: Busy.

COMP-TRAPS

Number of times a compatibility trap occurred during execution of this process. A compatibility trap is a data misalignment event, an unexpected transition to or from accelerated or unaccelerated code, or a relative segment 2 or 3 problem. See the *EPTRACE Manual* for further details.

Counter type: Incrementing.

PROGRAM-ACCELERATED

A flag that is set to nonzero if the UC (user code) or UL (user library) portions of the program have been accelerated.

ANCESTOR-CPU

Number of the CPU on which the ancestor process resides.

ANCESTOR-PIN

Process identification number of the ancestor process.

ANCESTOR-SYSNAME

Name of the system on which the ancestor process resides.

ANCESTOR-PROCESS-NAME

Name of the ancestor process. The report header displays the ancestor information. If this information cannot be obtained, the string “unknown” appears.

TNSR-BUSY-TIME

(Legacy Style only) The amount of time that the processor was busy executing native code for this process. This counter applies only to measurements taken on a TNS/R system. This counter is derived from other counters:

TNSR-BUSY-TIME = CPU-BUSY-TIME - (ACCEL-BUSY-TIME + TNS-BUSY-TIME)

Counter type: Busy.

TNSR-PROCESS

A flag set for processes executing only in native mode.

HOMETERM-SYSNAME

The home terminal system name associated with the process. This identifier appears only in the data record, not in the PROCESS report.

HOMETERM-NAME

Home terminal associated with the process, identified by subcomponent (device name, subdevice name, and qualifier name). This identifier appears only in the data record, not in the PROCESS report.

PAGE-SIZE-BYTES

Number of bytes in the memory page frame of the CPU being measured. This value is typically 4096 for TNS/R architectures and 2048 for other architectures.

ALLOC-SEG-CALLS

Number of successful calls made to the SEGMENT_ALLOCATE_ or ALLOCATESEGMENT procedures.

Counter type: Incrementing.

LOCK-PAGES-QTIME

No longer used.

LOCK-PAGES-MAX

No longer used.

UCL-QTIME

For G10 and later Measure PVUs, zero. For the nearest equivalent, see PRES-PAGES-QTIME (page 280), which includes user code and data frames, as well as system code and data frames that this process has caused to be present in memory.

For G09 and earlier Measure PVUs, the time that user code and library pages were mapped. This counter is a subset of (not an addition to) the PRES-PAGES-QTIME counter.

With REPORT RATE OFF, this counter is the total queue time for the memory category. With REPORT RATE ON, this counter is the average number of UCL pages.

You can get process data page information by subtracting UCL-QTIME (total code and library pages) from PRES-PAGES-QTIME (the total of all pages mapped for the process.

Counter type: Queue.

UCL-MAX

(Legacy Style only)

For G10 and later Measure PVUs, zero.

For G09 and earlier Measure PVUs, Maximum number of items on the queue described by the UCL-QTIME counter.

Counter type: Max queue.

UCL-LOCK-QTIME

No longer used.

UCL-LOCK-MAX

No longer used.

FILE-OPEN-CALLS

Number of calls to the FILE_OPEN_ or OPEN procedures. Counts all opens including opens of regular files, pipes, FIFOs, AF_INET sockets, AF_UNIX sockets, directories, /dev/null, and TTYs. Includes implicit opens due to the fork(),exec*() family of process creation APIs.

Counter type: Incrementing.

INFO-CALLS

Number of calls to these procedures that resulted in a message being sent to a process (IOP or other) to obtain the information requested:

FILEINFO	DEVICEINFO2	getsockopt()
FILERECINFO	access()	pathconf()
FILE_GETINFO_	fpathconf()	select()
FILE_GETINFOBYNAME_	fstat()	sockatmark()
FILE_GETINFOLISTBYNAME_	fstatvfs()	stat()
FILEGETINFO_LIST	getpeername()	statvfs()
DEVICEINFO	getsockname()	

Counter type: Incrementing.

BEGIN-TRANS

Number of times the process began a TMF transaction.

Counter type: Incrementing.

ABORT-TRANS

Number of times the process invokes the TMF ABORTTRANSACTION procedure, thus causing a TMF transaction to be rolled back.

Counter type: Incrementing.

DEVICE-NAME

Disk device on which the program file is located. For SMF files, this field provides the physical location that corresponds to the logical program file name. For NonStop files, this field is the same as the PROGRAM-FILE-NAME VOLUME subfield.

SENT-BYTES-F

For G-series RVUs, same as SENT-BYTES but accommodates larger values (64 bits rather than 32).

RETURNED-BYTES-F

For G-series RVUs, same as RETURNED-BYTES but accommodates larger values (64 bits rather than 32).

RECEIVED-BYTES-F

For G-series RVUs, same as RECEIVED-BYTES but accommodates larger values (64 bits rather than 32).

REPLY-BYTES-F

For G-series RVUs, same as REPLY-BYTES but accommodates larger values (64 bits rather than 32).

PRES-PAGES-START

Number of pages in main memory sponsored by the measured process at the start of the measurement interval.

Counter type: 32-bit snapshot.

PRES-PAGES-END

Number of pages in main memory sponsored by the measured process at the end of the measurement interval.

Counter type: 32-bit snapshot.

ABS-SEGS-QTIME

The time that segment table entries were in use for absolute addressable segments on behalf of the measured process.

Counter type: Queue.

ABS-SEGS-QLEN-MAX

(Legacy) Maximum number of items on the queue described by ABS-SEGS-QTIME. In H-series and J-series RVUs, the value is always 1.

Counter type: Max Queue.

ABS-SEGS-START

Number of items on the queue described by ABS-SEGS-QTIME at the start of the measurement interval.

Counter type: snapshot.

ABS-SEGS-END

Number of items on the queue described by ABS-SEGS-QTIME at the end of the measurement interval.

Counter type: snapshot.

MSGS-SENT-QTIME

The time (in microseconds) that messages sent by the process were outstanding.

Counter type: Queue.

MSGS-SENT-QLEN-MAX

(Legacy) Maximum number of items on the queue described by the MSGS-SENT-QTIME counter. In H-series and J-series RVUs, the value is always 1.

Counter type: Max queue.

SENT-CBYTES

Number of protocol control bytes sent by the message system on behalf of this process. This counter is updated when the process is acting as a linker process. The protocol control bytes are not included in the counter SENT-BYTES.

Counter type: Accumulating.

RETURNED-CBYTES

Number of protocol control bytes returned by the message system on behalf of this process. This counter is updated when the process is acting as a linker process. The protocol control bytes are not included in the counter RETURNED-BYTES.

Counter type: Accumulating.

RECEIVED-CBYTES

Number of protocol control bytes received by the message system on behalf of this process. This counter is updated when the process is acting as a listener process. The protocol control bytes are not included in the counter RECEIVED-BYTES.

Counter type: Accumulating.

REPLY-CBYTES

Number of protocol control bytes sent by the message system on behalf of this process in reply to a message from a linker process. This counter is updated when the process is acting as a listener process. The protocol control bytes are not included in the counter REPLY-BYTES.

Counter type: Accumulating.

OSSPID

If the process runs in the OSS environment, this field is the OSS Process ID. For a non-OSS process, the OSSPID contains zeros.

PROGRAM-FILE-NAME_MID

This field consists of two subfields: PATHID and CRVSN. For the OSS file set, PATHID is an internal representation of an OSS file name for the specified PROGRAM-FILE-NAME. For non-OSS files, this field contains zeros. CRVSN is the creation version serial number that identifies a unique instance of an OSS file. For other files, this field contains zeros.

OSS-TTY-WRITES

The number of WRITE operations performed by the process to all OSS file opens of /dev/tty.

Counter type: Incrementing.

OSS-TTY-READS

The number of READ operations performed by the process to all OSS file opens of /dev/tty.

Counter type: Incrementing.

OSS-TTY-WRITE-BYTES

The number of bytes written by the process to all files opened as /dev/tty.

Counter type: Accumulating.

OSS-TTY-READ-BYTES

The number of bytes read by the process from all files opened as /dev/tty.

Counter type: Accumulating.

OSS-TTY-WAIT-TIME

Amount of time the process waits on requests to all files opened as /dev/tty.

Counter type: Busy.

OSS-DEV-NULL-OPS

The number of reads and writes to all files opened as /dev/null.

Counter type: Incrementing.

OSSNS-DD-CALLS

The number of data definition requests sent to OSS Name Servers. These APIs increment this counter:

bind()	open(, O_CREAT)	mkfifo()	utime()
chmod()	link()	rename()	
chown()	mkdir()	rmdir()	
creat()	mknod()	unlink()	

Counter type: Incrementing.

OSSNS-REQUESTS

The number of requests sent to OSS Name Servers. Requests can be fileinfo calls, data definition requests, directory operations, or resolution requests.

Counter type: Incrementing.

OSSNS-MESSAGE-BYTES

The number of message data bytes sent and received for requests to OSS name servers.

Counter type: Accumulating.

OSSNS-REDIRECTS

The number of redirected requests sent to OSS name servers.

Counter type: Incrementing.

OSSNS-WAIT-TIME

Amount of time the process waits on requests to all OSS name servers.

Counter type: Busy.

LAUNCHES

The number of calls made by the process to process creation procedures. For Guardian process creation, the counter includes unsuccessful creation attempts.

Counter type: Incrementing.

LAUNCH-WAIT-TIME

Amount of time the process waits on requests to PROCESS_CREATE_ or the fork()/exec*() family of process creation APIs.

Counter type: Busy.

OPEN-CLOSE-WAIT-TIME

Amount of time the process waits on the open and close requests. For OSS processes, wait time accumulates for file open and close operations, excluding name server operations. Name server operations are counted in OSSNS-BUSY-TIME.

Counter type: Busy.

GMOM-NODE

The Expand node number of the GMOM process if this process is part of a NetBatch job and the GMOM process is remote; otherwise zero.

GMOM-CPU

The processor number of the GMOM process if this process is part of a NetBatch job; otherwise zero.

GMOM-PIN

The PIN of the GMOM process if this process is part of a NetBatch job; otherwise zero.

GMOM-JOBID

The job ID of the NetBatch job that initiated this process, assigned by the indicated GMOM; otherwise zero.

GMOM-FULL-ID

A 64-bit redefinition of the individual GMOM fields as a single value. Zero indicates that the process is not part of a NetBatch job stream.

GMOM-SYSNAME

The Expand system name of the GMOM node if the GMOM process is remote; otherwise spaces.

GMOM-PROCESS-NAME

The name of the GMOM process that initiated the NetBatch job that initiated this process; otherwise spaces.

SYSTEM-PROCESS

Reserved for future use.

IPUS

In J01 and later Measure PVUs, indicates the number of IPUs in the CPU. The field is not displayed by MEASCOM, but determines whether the IPU-SWITCHES field is displayed.

In H03 and later H-series Measure PVUs, this field is 1, and is not displayed by MEASCOM.

IPU-SWITCHES

An accumulating counter that indicates the total number of times that a process switched IPUs while executing. On a non-NSMA system, this is always 0 (zero).

Counter type: Accumulating.

IPU-NUM

In J03 and later J-series Measure PVUs, the number of the IPU on which the process was executing when the sample was taken..

In H05 and later H-series Measure PVUs, this field is 0 (zero), and is not displayed by MEASCOM.

Counter type: Snapshot.

IPU-NUM-PREV

In J03 and later J-series Measure PVUs, if an IPU switch took place during the measurement interval, IPU-NUM-PREV displays number of the IPU on which the process was executing before the most recent switch. If no IPU switch occurred during the measurement interval, this field is not displayed.

In H05 and later H-series Measure PVUs, this field is 0 (zero), and is not displayed by MEASCOM.

Counter type: Snapshot.

Usage Note for All PROCESS Entities

For information on how processes are dispatched, when they are executed, and so on, see the appropriate system description manual.

Usage Notes for G-Series PROCESS Entities

- The 64-bit byte-count fields ending in -F collect the same data as existing 32-bit byte-count fields. For example, the 64-bit field RETURNED-BYTES-F collects the same data as the 32-bit field RETURNED-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. (A field overflow is indicated by a value of -1 in both the field that has overflowed and in the ERROR field for the measured entity.) However, you should convert your applications to use the 64-bit fields. The 32-bit fields might be deactivated in a future RVU.

  In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST PROCESS BY RETURNED-BYTES, not LIST PROCESS BY RETURNED-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

- As of the G05 RVU, changes in memory handling have introduced the concept of sponsored and unsponsored memory pages. Pages are sponsored in memory by the process that causes them to be present (usually, but not always, the process that defines the segment). If a process that is the sponsor of shared pages terminates, the pages become temporarily unsponsored. The next sharing process to reference an unsponsored page becomes the sponsor for that page.

  The PRES-PAGES-QTIME counter provides information about pages sponsored by the measured process.

- For a discussion on the different types of message system transfer protocols and concepts, (pre-push, post-pull, linker and listener), see the *NonStop S-Series Server Description Manual*.

Usage Notes for H-Series and J-Series PROCESS Entities

- In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.
- Counters that report maximum numbers (counters that have MAX in their names) are valid only in Legacy style. In H-series and J-series RVUs, the value of any such counter is 1.
- For Measure PVUs H02, H03, H04, J01, and J02, the ZMS style PROCESS displays report a subsystem version of 2.
- For H05 and later H-series Measure PVUs and J03 and later J-series Measure PVUs, the ZMS style PRCOESS displays report a subsystem version of 3.

PROCESSH

The PROCESSH entity type provides information about the relative execution time of one or more code ranges within a program.

You can measure code ranges by specifying procedure names and address ranges or by specifying a code-space category (user code, user library, system code, or system library).

If you use a wildcard in a file name that applies to multiple disk files (such as $SYSTEM.WORK.*), you must specify a code-space category rather than a procedure name and address range.

You do not need to explicitly add the PROCESS entity for the PROCESSH entity. When you add the PROCESSH entity to a measurement, the corresponding PROCESS entity is automatically added for the measured process. You can use the PROCESS entity to get more information about

the measured process with two exceptions: the ALLTIME and ALLINTR measurements. Although Measure creates a PROCESS record for each of these measurements, these PROCESS records are just placeholders because these two measurements do not involve measuring any specific processes.

NOTE: In H-series and J-series RVUs, the ALLINTR option for PROCESSH sampling is disabled. On systems running H-series and J-series RVUs, all interrupt handlers are individual and separate Interrupt/ Auxiliary Processes and so can be sampled and measured individually.

In Measure G09 and later PVUs, the PROCESSH entity supports the use of OSS file pathnames in place of Guardian file names. The G09 PROCESSH displays OSS file pathnames in PROCESSH reports and provides direct mapping between external structured records and the new OSSNAMES structured record output.

Entity specification syntax for G09 PVU PROCESSH entities differs from the syntax for D-series and pre-G09 PVU entities. The DDL records also differ.

Entity Specification Syntax for PROCESSH Entities

To describe a PROCESSH entity:

```
PROCESSH entity-spec
```

PROCESSH

collects information about code ranges within one or more processes.

entity-spec

is one of:

```
{ process-spec ( code-file-spec ) [, ( code-file-spec ) ] … }
{ process-spec [, process-spec ] … }
{ ALLTIME | ALLINTR cpu ( code-file-spec )
   [, ( code-space code-file ) ] …  }
```

process-spec

is a process specification:

```
{ *                                                  }
{ cpu,pin                                            }
{ $process-name [ ( pid ) ]                          }
{ [[$device.]subvolume.]filename[:CRVSN][ ( pid ) ]}
{ "pname" [ ( pid ) ] }
```

where

*

measures all processes on the system.

cpu

is the number of the CPU on which the process to be measured is running. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

pin
is the process identification number of the process to be measured. To indicate all processes, use an asterisk (*). The default is all processes.

$*process-name*
is the name of the process to be measured.

pid
is the process identifier of the process to be measured. Specify *pid* as one of: { cpu,pin }

cpu
is the number of the CPU on which the process to be measured is running. To specify all CPUs, use an asterisk (*). The default is all CPUs.

pin
is the process identification number of the process to be measured. To specify all process identification numbers, use an asterisk (*). The default is all processes.

$*device*
is the device (volume) on which the object file of the process to be measured is located. To indicate all volumes, use an asterisk (*). The default is the current default volume.

subvolume
is the subvolume in which the object file of the process to be measured is located. To indicate all subvolumes, use an asterisk (*).

filename
is the name of the object file of the process to be measured. To indicate all files (except temporary files), use an asterisk (*).

If *filename* is specified, code-space and code-file parameters are required for TNS code. Otherwise these parameters are accepted but ignored, so as not to break any existing scripts.

:*CRVSN*
in Measure G10 and later PVUs, is the timestamp or creation version serial number or file-name extension, necessary to form a unique file name. Use this option to guarantee file-name uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

pname
can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and are expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

code-file-spec
{ [code-space] code-file] }

code-space

is a code-space specification. In G12 and later PVUs, *code-space* is required only for TNS and accelerated TNS code; otherwise this parameter is accepted but ignored.

For accelerated and TNS code, the code-space specification is:

Space	Specification	Accelerated	TNS
User code	UC[.n], where n is in the range:	0-31	0-15
User library	UL[.n], where n is in the range:	0-31	0-15
System code	SC[.n], where n is in the range:	0-31	0
System library	SL[.n], where n is in the range:	0-31	0-31

If n is not specified, its value defaults to all code spaces of the specified type.

For TNS systems, the number of code spaces for types UC and UL is cumulative and cannot total more than 32. Therefore, the maximum number of UL-type code spaces allowed is 16 (0 through 15) only if the number of UC-type code spaces specified is in the range 0 through 15 range. If the number of UC-type code spaces exceeds 16 (falls in the range 16 through 31 range), the maximum number of UL-type codes spaces is 31 minus UC.n. For example, if UC.n equals 25, the maximum UL.n you can specify is 6; that is, 31 - 25 = 6.

For native code, the code-space specification is:

Space	Specification
User code	UCR
User library	ULR
System code	SCR
System library	SLR

code-file

is the either the Guardian file name of a TNS object file, an EDIT file, or an Executable and Linking Format (ELF) object file (TNS/R file code 700 or TNS/E file code 800), or for a G09 or later Measure PVU, the OSS file pathname, a "pname", that designates one of the same set of file types that can be specified via a Guardian file name.

If code-file is an object file, MEASFH examines the procedure names and addresses and adds each procedure to the configuration. If code-file is an EDIT file, it must contain a set of name-address tuples, listed in ascending order of address, formatted as:

code-range-name code-range-address

where

code-range-name

is a code range (procedure) name of 1 through 1024 alphanumeric characters, circumflexes (^), hyphens (-), or underscores (_). The first character can be a letter, a circumflex, or an underscore. Except for TNS and accelerated code, the first character can also be a dollar sign ($).

code-range-address

In an EDIT (code 101) file describing TNS code, *code-range-address* is an octal address in the range 0 through 177777 or an address offset.

For EDIT (code 101) files describing TNS/R or TNS/E code, code-range-address is a valid virtual address for the systems, in hexadecimal, or an address offset. To indicate an offset from the previous code-range-address, precede the octal or hexadecimal number with a plus sign (+). The last code-range-address without a plus sign is added to the offset to yield the effective address.

The first line of the EDIT file must contain an object file name specifier consisting of the keyword OBJECT followed by a Guardian file name or OSS pathname. A Guardian file name example is OBJECT $A.B.MYAPP and an OSS pathname example is OBJECT "/a/b/myapp".

ALLTIME

specifies measurement of the execution time for all processes, including interrupts.

In PVUs earlier than G12, if you specify ALLTIME, you must specify *code-space* as SC.n or SL.n for TNS code-file measurements. In G12 and later PVUs, specify *code-space* only for TNS and accelerated code. (If you specify a *code-space* when it is not required, the parameter is accepted but ignored.)

ALLINTR

specifies measurement of the execution time for all system interrupts.

In PVUs earlier than G12, if you specify ALLINTR, you must specify *code-space* as SC.n or SL.n for TNS code-file measurements. In G12 and later PVUs, specify *code-space* only for TNS and accelerated code. (If you specify a *code-space* when it is not required, the parameter is accepted but ignored.)

This option is disabled in H-series and J-series RVUs. See Usage Notes for H-Series and J-Series PROCESS Entities (page 291).

cpu

is the number of the CPU on which the process to be measured is running. To indicate all CPUs, use an asterisk (*). There is no default value.

DDL Record for PROCESSH Entities (Legacy Style)

This is the Legacy Style DDL record for PROCESSH entities. This record will not change after the G10 Measure PVU.

```
RECORD processh.  FILE is "processh" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 priority                     type binary 16 unsigned.
* code range identifiers and value:
 02 process-busy-samples         type binary 32 unsigned.
 02 code-space                   type binary 16 unsigned.
 02 code-space-busy-samples      type binary 32 unsigned.
```

```
 02 code-range                  type character 32.
 02 code-range-busy-samples     type binary 32 unsigned.
* new entity identification field:
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
* fields for Liberty specific counters as of C30:
 02 accel-busy-samples          type binary 32 unsigned.
 02 tns-busy-samples           type binary 32 unsigned.
* new entity identification items for D10:
 02 ancestor-cpu                type binary 16 unsigned.
 02 ancestor-pin                type binary 16 unsigned.
 02 ancestor-sysname            type character 8.
 02 ancestor-process-name       type character 8.
* Native Mode busy samples:
 02 tnsr-busy-samples           type binary 32 unsigned.
* SMS changes:
 02 device-name                 type character 8.
* new identifiers for OSS file pathname support:
 02 osspid                      type binary 32 unsigned.
 02 program-file-name-mid.
    03 pathid                   type character 24.
    03 crvsn                    type character 6.
 02 object-device-name    type character 8.
 02 object-file-name-MID.
    03 PATHID                   type character 24.
    03 CRVSN                    type character 6.
 02 reserved-1                  type character 6.
 02 code-range-flags            type binary 16 unsigned.
 02 code-range-length           type binary 16 unsigned.
 02 code-range                  type character 1024.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for PROCESSH Entities (ZMS Style)

The ZMS style PROCESSH DDL record is supported in Measure G11 and later PVUs.

ID Fields DDL Definition

```
DEFINITION zmsproch-id.
 02 pin                         type binary 16 unsigned.
 02 priority                    type binary 16 unsigned.
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 process-name                type character 8.
 02 program-file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 osspid                      type binary 32 unsigned.
 02 ancestor-cpu                type binary 16 unsigned.
 02 ancestor-pin                type binary 16 unsigned.
 02 ancestor-sysname            type character 8.
 02 ancestor-process-name       type character 8.
 02 device-name                 type character 8.
```

```
 02 program-file-name-MID.
    03 PATHID                    type character 24.
    03 CRVSN                     type character 6.
 02 code-space                   type binary 16 unsigned.
 02 object-device-name           type character 8.
 02 object-file-name-MID.
    03 PATHID                    type character 24.
    03 CRVSN                     type character 6.
 02 code-range-flags             type binary 16 unsigned.
 02 code-range-length            type binary 16 unsigned.
 02 reserved-1                    type character 6.
 02 code-range                   type character 1024.
 end
```

Counter Fields DDL Definition

```
DEFINITION zmsproch-ctrs.
 02 process-busy-samples         type binary 32 unsigned.
 02 code-space-busy-samples      type binary 32 unsigned.
 02 code-range-busy-samples      type binary 32 unsigned.
 02 accel-busy-samples           type binary 32 unsigned.
 02 tns-busy-samples             type binary 32 unsigned.
 02 native-busy-samples          type binary 32 unsigned.
end
```

DDL Record Description Fields

```
RECORD zmsproch. FILE is "zmsproch" ENTRY-SEQUENCED.
 02 hdr                          type zmsheader.
 02 ctr                          type zmsproch-ctrs.
 02 id                           type zmsproch-id.
end
```

PIN

Process identification number of the process.

PROCESS-NAME

Name of the process.

PROGRAM-FILE-NAME

Name of the object file the process is executing. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, PROGRAM-FILE-NAME represents the physical file name that was specified when the process was executed. The VOLUME subfield gives the device name of the physical volume on which the program file is located.

For SMF files, PROGRAM-FILE-NAME represents a location-independent logical file name that was used when the process was executed. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

PRIORITY

Priority of the process.

PROCESS-BUSY-SAMPLES

Number of times the sampling operation found the process executing.

CODE-SPACE

One of these code space identifiers:

0	UC.n	(user code—TNS or accelerated)
1	UCR	(user code—TNS/R native)
2	UL.n	(user library—TNS or accelerated)
3	ULR	(user library—TNS/R native)
4	SC.0	(system code—TNS or accelerated)
5	SCR	(system code—TNS/R native)
6	SL.n	(system library—TNS or accelerated)
7	SLR	(system library—TNS/R native)

CODE-SPACE-BUSY-SAMPLES

Number of times the sampling operation found the measured code space executing.

Counter type: Sampling.

CODE-RANGE

Name of the measured code range.

CODE-RANGE-BUSY-SAMPLES

Number of times the sampling operation found the measured code range executing.

For TNS/R and TNS/E systems running TNS and accelerated code, CODE-RANGE-BUSY-SAMPLES=ACCEL-BUSY-SAMPLES + TNS-BUSY-SAMPLES.

For TNS/R and TNS/E systems running TNS/R native code, CODE-RANGE-BUSY-SAMPLES = TNSR-BUSY-SAMPLES.

For TNS/E systems running TNS/E native code, CODE-RANGE-BUSY-SAMPLES = NATIVE-BUSY-SAMPLES.

Counter type: Sampling.

USERID

Process accessor ID (PAID) of the process.

CREATORID

Creator access ID (CAID). This field identifies the creator of the process and is divided into two subfields: GROUP and USER.

ACCEL-BUSY-SAMPLES

Number of times the sampling operation found the measured code executing accelerated code. This counter applies only to measurements taken on a TNS/R or TNS/E system. ACCEL-BUSY-SAMPLES + TNS-BUSY-SAMPLES = CODE-RANGE-BUSY-SAMPLES.

Counter type: Sampling.

TNS-BUSY-SAMPLES

Number of times the sampling operation found the measured code executing TNS code. This counter applies only to measurements taken on a TNS/R or TNS/E system. ACCEL-BUSY-SAMPLES + TNS-BUSY-SAMPLES = CODE-RANGE-BUSY-SAMPLES.

Counter type: Sampling.

NATIVE-BUSY-SAMPLES

Number of times the sampling operation found the measured code executing native code.

Counter type: Sampling.

ANCESTOR-CPU

Number of the CPU on which the ancestor process resides.

ANCESTOR-PIN

Process identification number of the ancestor process.

ANCESTOR-SYSNAME

Name of the system on which the ancestor process resides.

ANCESTOR-PROCESS-NAME

Name of the ancestor process. The report header displays the ancestor information. If this information cannot be obtained, the string “unknown” appears.

TNSR-BUSY-SAMPLES

Number of times the sampling operation found the measured code executing TNS/R native code. This counter applies only to measurements taken on a TNS/R system. TNSR-BUSY-SAMPLES = CODE-RANGE-BUSY-SAMPLES.

Counter type: Sampling.

DEVICE-NAME

Disk device on which the program file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

OSSPID

On Measure G09 and later PVUs, if the process runs in the OSS environment, this field is the OSS Process ID. For other processes, this field contains zeros.

PROGRAM-FILE-NAME-MID

In Measure G09 and later PVUs, this field has two subfields:

PATHID
: is an internal format representation of an OSS file name for the specified PROGRAM-FILE-NAME. For other files, this field contains zeros.

CRVSN
: is the creation version serial number that identifies a unique instance of an OSS file. For other files, this field contains zeros.

CODE-RANGE-FLAGS

Reserved for future use.

CODE-RANGE-LENGTH

Reserved for future use.

OBJECT-DEVICE-NAME

Reserved for future use.

OBJECT-FILE-NAME-MID

Reserved for future use.

Usage Notes for G-Series PROCESSH Entities

- TNS/R processors include short code sequences called gateways that handle the transition from non-native to native calls. If a sampled process is executing in a gateway, PROCESSH increments the sample count but does not assign the sample to a particular code space. Therefore, the total sample count might be larger than the sum of the various *code-space* counts.
- If the sum of the individual *code-space* sample counts is significantly less than the total samples count for the process, check (with the Native Object File Tool or NOFT utility) to see if any shared run-time libraries (SRLs) should have been added to the PROCESSH configuration. If so, add them in MEASCOM as URLs to the PROCESSH configuration. For an example, see Examples of PROCESSH Measurements.

NOTE: Because sampling does not occur in processor millicode or in gateways, the potential exists for a small number of missing samples.

Usage Notes for H-Series and J-Series PROCESSH Entities

- In H-series and J-series RVUs, the ALLINTR option for PROCESSH sampling is disabled. On systems running H-series and J-series RVUs, all interrupt handlers are individual and separate Interrupt/ Auxiliary Processes and so can be sampled and measured individually. No error is returned if an ALLINTR PROCESSH measurement is configured. But it will return no records. Interrupt/Auxiliary processes can be identified through the value of their priority in their PROCESS records. Only these processes have a PRI value of 255. So a MEASCOM command of LIST PROCESS 0,*, IF PRI = 255 returns a list of all Interrupt/Auxiliary processes that are executing on processor 0, provided that the measurement was done on all processes. The information (program filename, pin, and so on) from these records can be used to configure a PROCESSH measurement on these items if needed.
- In H-series and J-series RVUs, the PROCESSH sampling rate is increased to approximately 500 samples per second. The sampling rate is per IPU for J-series RVUs. This sampling rate is not fixed. If more samples are needed within a shorter duration, the rate can be increased. However, because this might negatively affect the performance of the other processes, use this facility only when necessary. To increase the PROCESSH sampling rate, before issuing the START MEASSUBSYS command, add this DEFINE statement to the TACL session context. The filename specified determines the scaling factor of the sampling rate:

DEFINE Statement	Scaling Factor
ADD DEFINE =_SET_PROCESSH_RATE, file SCALE2X	twice default rate
ADD DEFINE =_SET_PROCESSH_RATE, file SCALE4X	4 times
ADD DEFINE =_SET_PROCESSH_RATE, file SCALE8X	8 times
ADD DEFINE =_SET_PROCESSH_RATE, file SCALE16X	16 time
ADD DEFINE =_SET_PROCESSH_RATE, file SCALE32X	32 times

To determine the approximate sampling rate, issue successive PEEK INT requests in a processor to view the accumulated sample interrupts, or examine the counter PROCESSH-SAMPLES in the ZMSCPU report or through MEASCOM LIST CPU command. For example:

```
TACL 12>  ADD DEFINE  =_SET_PROCESSH_RATE, file SCALE4X
TACL 13>  MEASCOM START MEASSUBSYS
```

- If the *process-spec* is either a *process-name* or a *cpu,pin* combination, a *code-file-spec* needs to be specified to configure a PROCESSH measurement. If an object filename is specified as *process-spec*, the *code-file-spec* is optional. For LIST commands, it is not necessary to specify a *code-file-spec*.

Examples of PROCESSH Measurements

- This example tracks the use of system procedures by all processes in CPU 7. It shows the use of the system code and system library activity for the TNS/R system. This produces sampling reports for all of the system procedures run on a TNS/R system. The use of ALLTIME measures all system code and system library activity for all processes in the CPU.

```
+ADD PROCESSH ALLTIME 7 (SC $SYSTEM.SYSnn.TSC)
+ADD PROCESSH ALLTIME 7 (SL $SYSTEM.SYSnn.TSL)
```

NOTE: A TNS/R system has more than one system code (SC) file.

- This example measures SCR and SLR code spaces.

  To support the TNS/R operating environment, the SYSGEN program creates a new object file (file code 700), named TSYSCLR, which is placed in $SYSTEM.SYSnn subvolume. This object file can be used to sample TNS/R system code (SCR) and the TNS/R system library (SLR). Because TNS/R code spaces contain no code maps, the samples are displayed only for SCR and SLR. The TSYSCLR file is superseded in H-series and J-series RVUs, as described later in this list of examples.

  To sample TNS/R code spaces for SCR and SLR:

```
+ADD PROCESSH proc spec (SCR $SYSTEM.SYSnn.TSYSCLR)
+ADD PROCESSH proc spec (SLR $SYSTEM.SYSnn.TSYSCLR)
```

- This example describes how to check for shared run-time libraries (SRLs) in a program and how to include them in your program's PROCESSH measurement:
  1. Identify all the SRLs associated with a particular program using the Native Object File Tool (NOFT) utility.
  2. Add the SRLs as user library files (URLs) in MEASCOM.

     The accounting and mapping of the PROCESSH total samples count for the program and the sum of the individual code-space sample counts now match as closely as currently possible.

  If a program uses SRLs:

  1. Use the NOFT LISTSRLINFO command to identify the SRLs:

```
noft> FILE $VOL.SUBVOL.FILENAME

              Object File : \node.$system.sysnn.ztcpsrl
              File Format : ELF
                    Scope : (none)
                     Case : Sensitive
noft> LISTSRLINFO

                 SRL Name :      #
                  ZLANCSRL:      0
                  ZLANDSRL:      1
```

2. In MEASCOM, add these SRLs as ULRs to the PROCESSH configuration of the program:

```
2+ ADD PROCESSH $VOL.SUBVOL.FILENAME (UCR FILE-NAME)
3+ ADD PROCESSH $VOL.SUBVOL.FILENAME (ULR ZLANCSRL)
4+ ADD PROCESSH $VOL.SUBVOL.FILENAME (ULR ZLANDSRL)
```

- No TSYSCLR file exists in H-series and J-series RVUs. That file is replaced by the two system DLLs INITDLL and MCPDLL. This example reflects the H-series and J-series library structure. The code-space identifier SL designates nonnative system library code:

```
+ADD PROCESSH $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.TSYSDP2)
+ADD PROCESSH $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.INITDLL)
+ADD PROCESSH $SYSTEM.SYS01.TSYSDP2 (2, *) ($SYSTEM.SYS01.MCPDLL

+ADD PROCESSH ALLTIME 1 ($SYSTEM.SYS01INITDLL)
+ADD PROCESSH ALLTIME 1 ($SYSTEM.SYS01.MCPDLL)
+ADD PROCESSH ALLTIME 1 (SL $SYSTEM.SYS01.TSL)
```

SERVERNET

The SERVERNET entity type measures physical I/O operations involving ServerNet addressable controllers (SACs) on systems running G-series RVUs. The ServerNet entity tracks all interprocessor communication (IPC). The SERVERNET entity is somewhat similar to the CONTROLLER entity, which measured controller operations on systems running D-series RVUs.

Prior to Measure G08, the ServerNet entity measured all intrasystem traffic (for example, all interprocessor traffic within a system or node). With the introduction of the ServerNet Cluster, the SERVERNET entity now measures both intrasystem traffic and intersystem traffic (for example, all interprocessor traffic for remote nodes). Measure refers to the latter as Remote Interprocessor Communication or RIPC.

NOTE: You can measure ServerNet addressable controllers with a D-series RVU measurement application if it specifies all controllers (ADD CONTROLLER *) or only a CPU number (ADD CONTROLLER 2). To measure specific SACs, you must modify the entity identifiers.

Entity Specification Syntax for SERVERNET Entities

To describe a SERVERNET entity:

```
SERVERNET entity-spec [, entity-spec ] ...
```

SERVERNET

collects measurements for physical I/O operations involving SACs.

entity spec

is specified as:

```
{ *                                          }
{ cpu                                        }
{ cpu [ ( node-class [ , group [ , module [ , slot
```

```
[ , subdevice | port [ , fiber ] ] ] ] ] ) ] [ ( type ) ] }
```

*

measures all devices in all CPUs.

cpu

is a CPU number, which must be an integer in the range 0 through 15. The default is all CPUs.

node-class

is the class of SACs to be measured: SCSI, NIOC, ENET, SWAN, IPC , RIPC, COLO, or MONT.

Remote Interprocessor Communication (RIPC) describes communication with a CPU on a remote ServerNet node.

Use an asterisk (*) to indicate all node classes. If *node-class* is not specified, all node classes are measured.

group

is the group number of the SAC to be measured. (The group corresponds to the physical enclosure). Use an asterisk (*) to indicate all groups. The default is all groups.

module

is the module number of the SAC to be measured. For RIPC, the module number is equal to the ServerNet node number. Use an asterisk (*) to indicate all modules. The default is all modules.

slot

is the slot number of the SAC to be measured. Use an asterisk (*) to indicate all slots. The default is all slots.

subdevice

is a specific subdevice identifier in cases where the *group*, *module*, and *slot* identifiers or the configuration name indicate more than one SAC. For Remote Interprocessor Communication (RIPC), subdevice is equal to the CPU number of a processor in a remote node, system, or ServerNet cluster. To indicate all remote processors on the specified remote node, system, or ServerNet cluster, use an asterisk (*). If not specified, the default is all remote processors on the specified remote node, system, or ServerNet cluster.

port

identifies a specific ServerNet connector within the ServerNet PIC (or within the ServerNet Blade Switch on a NonStop BladeSystem) which a CLIM is connected to. On NonStop BladeSystems only, the valid port values are 3 to 8, or "*". On all other systems, valid values for port are 1 to 4, or "*".

fiber

identifies a specific connector within the ServerNet cable the CLIM is connected to. Valid values for fiber are 1 to 4. Fiber is only relevant on a NonStop BladeSystem.

type

is the part number of a class of ServerNet addressable controllers.

NOTE: The *type* specification is not used for CLIM node classes.

DDL Record for SERVERNET Entities (Legacy Style)

This is the Legacy Style SERVERNET DDL record. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

```
RECORD servernet.  FILE is "svnet" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
* entity identification items:
 02 channel                        type binary 16 unsigned.
 02 ctrl                           type binary 16 unsigned.
 02 ctrl-type                      type binary 16 unsigned.
* counter value items:
 02 requests                       type binary 32 unsigned.
 02 total-io-bytes                 type binary 64.
 02 io-qtime                       type binary 64.
 02 io-qlen-max                    type binary 16 unsigned.
* F40 new entity identification items:
 02 adapter-name                   type character 64.
 02 SAC-name                       type character 64.
 02 node-class                     type binary 32 unsigned.
 02 node-class-s                   type character 4
                                   redefines node-class.
 02 GMS.
    03 group                       type binary 32 unsigned.
    03 module                      type binary 32 unsigned.
    03 remote-cluster              type binary 32 unsigned
                                   redefines module.
    03 slot                        type binary 32 unsigned.
 02 subdevice                      type binary 32 unsigned.
 02 remote-CPU                     type binary 32 unsigned
                                   redefines subdevice.
* F40 new counter value items:
 02 io-qbusy-time                  type binary 64.
* G01 counter value items:
 02 read-requests                  type binary 32 unsigned.
 02 read-bytes                     type binary 64.
 02 read-qtime                     type binary 64.
 02 read-qlen-max                  type binary 16 unsigned.
 02 read-qbusy-time                type binary 64.
 02 write-requests                 type binary 32 unsigned.
 02 write-bytes                    type binary 64.
 02 write-qtime                    type binary 64.
 02 write-qlen-max                 type binary 16 unsigned.
 02 write-qbusy-time               type binary 64.

 * G08 new counters:
 02 read-cbytes                    type binary 64.
 02 write-cbytes                   type binary 64.
 02 server-qtime                   type binary 64.
 02 server-qlen-max                type binary 16 unsigned.
 02 retries                        type binary 32.
 02 acks                           type binary 32.
 02 x-defrd-busy-time              type binary 64.
 02 y-defrd-busy-time              type binary 64.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for SERVERNET Entities (ZMS Style)

The ZMS style SERVERNET DDL record is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmssvnet-id.
 02 node-class                  type binary 32 unsigned.
 02 node-class-s                type character 4
                                redefines node-class.
 02 ctrl-type                   type binary 16 unsigned.
 02 reserved-1                  type character 2.
 02 GMS.
    03 group                    type binary 32 unsigned.
    03 module                   type binary 32 unsigned.
    03 remote-cluster           type binary 32 unsigned
                                redefines module.
    03 slot                     type binary 32 unsigned.
 02 subdevice                   type binary 32 unsigned.
 02 remote-cpu                  type binary 32 unsigned
                                redefines subdevice.
 02 PF                          redefines subdevice.
    03 port                     type binary 16 unsigned.
    03 fiber                    type binary 16 unsigned.
 02 adapter-name                type character 64.
 02 SAC-name                    type character 64.
end
```

Counter Fields DDL Definition

```
DEFINITION zmssvnet-ctrs.
 02 requests                    type binary 64.
 02 total-io-bytes              type binary 64.
 02 io-qtime                    type binary 64.
 02 io-qbusy-time               type binary 64.
 02 read-requests               type binary 64.
 02 write-requests              type binary 64.
 02 read-bytes                  type binary 64.
 02 read-cbytes                 type binary 64.
 02 read-qtime                  type binary 64.
 02 read-qbusy-time             type binary 64.
 02 write-bytes                 type binary 64.
 02 write-cbytes                type binary 64.
 02 write-qtime                 type binary 64.
 02 write-qbusy-time            type binary 64.
 02 server-qtime                type binary 64.
 02 retries                     type binary 64.
 02 acks                        type binary 64.
 02 x-defrd-busy-time           type binary 64.
 02 y-defrd-busy-time           type binary 64.
 02 read-crequests              type binary 64.
 02 write-crequests             type binary 64.
 end
```

DDL Record Description Fields

```
RECORD zmssvnet.  FILE is "zmssvnet" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmssvnet-ctrs.
 02 id                          type zmssvnet-id.
end
```

CHANNEL

(Legacy Style only) Not used; returns zero.

CTRL

(Legacy Style only) Not used; returns zero.

CTRL-TYPE

The product number, such as 3128.

For IPC records, CTRL-TYPE is 0 for the linker, 1 for the listener. For more information on IPC records, see Usage Notes for SERVERNET Entities (page 310).

REQUESTS

Total number of requests to the SAC by all IOPs connected to this controller path.

Counter type: Accumulating.

TOTAL-IO-BYTES

Total number of bytes transferred to and from the SAC by all IOPs connected to this controller path.

Counter type: Accumulating.

IO-QTIME

The time the SAC is busy performing I/Os.

With REPORT RATE ON, this is the average number of outstanding I/Os queued to this controller. If the average number of I/Os is less than 1, the number can be interpreted as the controller busy percentage.

Counter type: Queue.

IO-QLEN-MAX

(Legacy Style only) Maximum number of outstanding I/Os on the queue described by the IO-QTIME counter.

Counter type: Max queue.

ADAPTER-NAME

The logical name associated with the adapter in which the SAC resides. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

SAC-NAME

The logical name of the SAC used in this path to the physical disk. Logical names are maintained by the system configuration database. The name is null-filled and null-terminated.

For IPC records, the Measure product generates this identifier instead of a SAC name:

`IPC-CPUxx-TO-CPUyy`

where *xx* is the linker CPU number and *yy* is the listener CPU number.

NODE-CLASS

Identifies the SAC type as defined in the system configuration database.

NODE-CLASS-S

Redefines NODE-CLASS as a four-character alphanumeric field. The possible values (blank filled to four characters) are:

- "CLIM"
- "CLMI"
- "CLMS"

- “CLMO”
- "SCSI"
- "NIOC"
- "ENET"
- "SWAN"
- "IPC "
- "RIPC"
- "COLO"
- "MONT"
- "* "

NOTE: CLIM is a wildcard for all CLIM node classes.

GMS

The physical location address (group, module, or ServerNet node, and slot numbers). The GMS field is divided into these subfields:

GROUP

MODULE or SVNET-NODE-NUMBER **(redefines module)**

SLOT

For IPC records, GROUP, MODULE, and SLOT all have values of 0.

SVNET-NODE-NUMBER is the ServerNet node number of the system with which the entity is communicating.

You can use the keywords GROUP, MODULE, SVNET-NODE-NUMBER, and SLOT in the IF and BY clauses of the LIST and LISTALL commands.

SUBDEVICE

Subdevice identifier for a particular SCSI device on a multifunction I/O board (MFIOB):

Identifier	Device type
0	ServerNet bus interface (SBI)
1	Internal SCSI Controller #1
2	Internal SCSI Controller #2
3	COMM and SP
4	SP to SP communications
5	External SCSI port controller

For IPC records, SUBDEVICE applies to the destination (listener) processor. For more information, see Usage Notes for SERVERNET Entities (page 310).

REMOTE-CPU

The number of the CPU on a remote ServerNet cluster with which the entity is communicating. This redefines `subdevice` in the DDL record.

PF

The structure that holds the port and fiber fields.

PORT

The port identifies a specific ServerNet connector within the ServerNet PIC (or within the ServerNet Blade Switch on a NonStop BladeSystem) which the CLIM is connected to.

FIBER

(NonStop BladeSystems only.) The fiber identifies a specific connector within the ServerNet cable the CLIM is connected to.

IO-QBUSY-TIME

The time spent in a state in which requests of any number or type were queued for this entity.

Counter type: Queue Busy.

READ-REQUESTS

The number of requests that transferred data from this entity into the CPU.

Counter type: Incrementing.

READ-BYTES

The total number of bytes transferred as a result of read requests or pull operations and the bytes received in pre-push operations from the linker CPU.

Counter type: Accumulating.

READ-QTIME

The total time spent by read requests queued for this entity. With REPORT RATE ON, the average number of read requests queued for this entity.

Counter type: Queue.

READ-QLEN-MAX

(Legacy Style only) The maximum number of requests on the read request queue since the counter record was allocated.

Counter type: Max queue.

READ-QBUSY-TIME

The time spent in a state in which requests for data transfer from this entity to memory were queued.

Counter type: Queue Busy.

WRITE-REQUESTS

The number of requests that transferred data from the CPU to this entity.

Counter type: Incrementing.

WRITE-BYTES

The total number of bytes transferred as a result of write requests.

Counter type: Accumulating.

WRITE-QTIME

The total time spent by write requests queued for this entity. With REPORT RATE ON, the average number of write requests queued for this entity. For IPC and RIPC node class entities, this counter has changed (as of G08 Measure) to include the entire time the linker waits for a request to complete.

Counter type: Queue.

WRITE-QLEN-MAX

(Legacy Style only) The maximum number of requests on the write request queue since the counter was allocated.

Counter type: Max queue.

WRITE-QBUSY-TIME

The time spent in a state in which requests for data transfer from memory to this entity were queued.

Counter type: Queue Busy.

READ-CBYTES

For IPC and RIPC node classes, the total number of message system protocol control bytes transferred to this processor as a result of requests. For a CLIM node class, the total number of bytes in the incoming control requests from the CLIM into the CPU. In all cases, these control bytes are not included in the counter READ-BYTES.

Counter type: Accumulating.

WRITE-CBYTES

For IPC and RIPC node classes, the total number of message system protocol control bytes transferred to this processor as a result of requests. For a CLIM node class, the total number of bytes in the outgoing control requests from the CPU into the CLIM. In all cases, these control bytes are not included in the counter WRITE-BYTES.

Counter type: Accumulating.

SERVER-QTIME

The time (in microseconds) between the receipt of messages in this processor and the issuing of replies. This value does not include READ-QTIME for ServerNet read operations to pull the message system data to the CPU. The counter helps to isolate ServerNet latency time from the time required for the receiving process to respond to a request. This applies only to IPC and RIPC records.

Counter type: Accumulating.

SERVER-QLEN-MAX

(Legacy Style only) The maximum number of items on the queue described by the SERVER-QTIME counter. This counter applies only to IPC and RIPC records.

In H-series and J-series RVUs, this counter has a value of 1.

Counter type: Max queue.

RETRIES

The number of times the message system tries to send a message before succeeding. This counter applies only to IPC and RIPC records.

Counter type: Incrementing.

ACKS

The number of explicit acknowledgment messages sent. This counter applies only to IPC and RIPC records.

Counter type: Incrementing.

X-DEFRD-BUSY-TIME

The time (in microseconds) during which the message system was unable to switch from using the Y fabric to using the X fabric. A large value for this counter can be the result of an outstanding message on the Y fabric or a problem with the X fabric. This counter applies only to IPC and RIPC records.

Counter type: Busy.

Y DEFRD-BUSY-TIME

The time (in microseconds) during which the message system was unable to switch from using the X fabric to using the Y fabric. A large value for this counter can be the result of an outstanding message on the X fabric or a problem with the Y fabric. This counter applies only to IPC and RIPC records.

Counter type: Busy.

READ-CREQUESTS

For a CLIM, the number of incoming control requests from this entity into the CPU. This counter appears in the NORMAL but not the BRIEF display. Counter type: Incrementing.

WRITE-CREQUESTS

For a CLIM, the number of outgoing control requests from the CPU into this entity. This counter appears in the NORMAL but not the BRIEF display. Counter type: Incrementing.

Usage Notes for SERVERNET Entities

- The SERVERNET entity type provides a view of all ServerNet operations across a system. When several devices share a SAC, the SERVERNET entity type measures combined activity.
- SUBSYSTEM-VERSION for ZMSSVNET records is provided by the specific subsystem represented by the node^class of the record:
  - For IPC and RIPC, the message system
  - For SCSI and COLO, the SCSI module driver
  - For NIOC, the SLSA module driver
- Some overlap exists between the SERVERNET entity type and the device entity types (DEVICE, DISK, and so on) in cases where a single device is connected to a single controller.
- For storage SACs such as MFIOB devices, the IO-QBUSY-TIME counter is the most accurate indicator of busy time and service time for each request. The READ-QBUSY-TIME and WRITE-QBUSY-TIME counters provide similar data, but these two counters are somewhat less accurate in the ServerNet environment due to request queuing.
- Interprocessor communications (IPC) activity is measured partially in the initiating processor (called the linker) and partially in the receiving or replying processor (called the listener). In each processor, one SERVERNET record tracks the linker activity with another processor, and a second SERVERNET record tracks listener activity. The Measure performance monitor generates this identifier in the SAC Name field for IPC SERVERNET records:

  `IPC-CPUxx-TO-CPUyy`

  where *xx* is the linker CPU number and *yy* is the listener CPU number.
- To support ServerNet cluster connections, the data in the linker and listener records have changed to let the linker record (initiating processor) fully account for message traffic. This support is accomplished by ensuring that both the linker and listener records count the same number of message exchanges and corresponding bytes counts. The result of this change allows for better accounting from the linker record because the listener record of a ServerNet cluster connection will be in another node and not readily available in a Measure data file from only one node.

Usage Notes for ServerNet IPC and RIPC

- Listener READ-BYTES can exceed the corresponding linker record WRITE-BYTES. This occurs in cases of frequent message system MQC allocation failure, which causes a retransfer of the message data.
- For pre-push (non-ServerNet cluster) messages, the count of READ-REQUESTS is not incremented in the listener record. The count of READ-REQUESTS in the listener record reflects the number of message system pull operations performed and is a subset of the WRITE-REQUESTS counter. The comparison of READS and READ-BUSY-QTIME provides an accurate measure of ServerNet latency for pull operations.
- With ServerNet cluster (which does not pre-push messages), when the remote processor or listener is trying to satisfy a WRITE-REQUEST and the messages are large, the number of READ-REQUESTS can potentially be double the number of REQUESTS. For messages larger than readlinkcache size (for example, larger than 2048 bytes), the listener process performs two pulls: one for control and one for data.
- The SERVER-QTIME counter in message system listener records measures the time between the receipt of a message by the listener processor and the subsequent reply. When used in conjunction with the WRITE-QTIME counter in the associated linker processor record, this counter gives a measurement of the ServerNet latency for message system transfers. Average message latency can be calculated by subtracting listener record SERVER-QTIME from linker record WRITE-QTIME, then dividing by the linker record WRITE-REQUESTS.

  Calculating average message latency:

  ```
  WRITE-QTIME(linker) - SERVER-QTIME(listener)/WRITE-REQEUSTS(linker)
  ```
- Measure does not coordinate measurement intervals between systems. To calculate the average message latency between processors in separate ServerNet clusters, calculate the average SERVER-QTIME and WRITE-QTIME values in a similar interval and then subtract the resulting values.
- High values of the RETRIES counter are expected for connections to NonStop S70000 servers (NSR-G processors). Defects in the ServerNet interface (MITE) for these processors are handled through software recovery. These problems are fixed in the NonStop S72000 servers (NSR-T processors), but the S72000 servers might still report high numbers of RETRIES when communicating with NonStop S70000 servers (NSR-G processors).
- In some cases, when the remote processor or listener is trying to satisfy a WRITE-REQUEST and the messages are very large, the number of READ-REQUESTS can potentially be double the number of REQUESTS. For messages larger than readlinkcache size (for example, larger than 2048 bytes), the listener process performs two pulls: one for control and one for data.

Usage Notes for CLIMs

- Starting with the Measure H03 and J01 PVUs, Measure supports the CLIM *node-class* specifications CLMI for IP CLIMs and CLMS for storage CLIMs. For these node classes, the optional fourth specification after *node-class* is *port*. For non-CLIM node classes, it is *subdevice*.

  Starting with the Measure H04 and J02 PVUs, Measure supports the CLIM *node-class* specification CLMO for Telco CLIMs.

  CLIM is a wildcard for all CLIM node classes.

  Except for NonStop BladeSystems, each CLIM is uniquely identified by group-module-slot-port, which identifies the CLIM in terms of the physical location of the P-Switch it is connected to and the port on which it is connected. The group (system enclosure) and module (subset of a group) identifies the P-Switch. The slot (physical, labeled space in a module) identifies a particular ServerNet PIC within that P-Switch and the port. The port

identifies a specific ServerNet connector within the ServerNet PIC the CLIM is connected to. NonStop BladeSystems require the additional fiber number to identify a CLIM.

- The activity for each storage CLIM is reported in two SERVERNET records. One reports activity from the Storage subsystem; the other reports activity from the CIP subsystem.

  The Storage subsystem record will report IO activity (READ-BYTES, WRITE-BYTES, READ-REQUESTS, WRITE-REQUESTS and various Queue counters) for all the devices supported by the CLIM. It will not report any Control IO activity (READ-CBYTES, WRITE-CBYTES, READ-CREQUESTS and WRITE-CREQUESTS counters).

  The CIP subsystem will report Control IO activity performed with the CLIM and will not report any IO activity. Normally, the Control IO activity is just the low frequency pings between the CIP subsystem and the CLIM.

  Here are examples of two report records for a storage CLIM; the first is from the CIP subsystem, the second is from the Storage subsystem.

  This is the example from the CIP subsystem:

```
SAC   C1002531
Adpt  C1002531
CPU   6    Node Class  CLMS   GMS  100,2,5     Port  3   Fiber  1
Format Version:  H04  Data Version:  H04  Subsystem Version:  1
Local System \BLNSK2  From  26 Mar 2009, 17:13:10   For  33.6 Seconds
------------------------------------------------------------------------
Requests                        12 #    Total-IO-Bytes               672 #
IO-Qtime                                IO-Qbusy-Time
--------- Data into this CPU --------  --------- Data out from this CPU ----
Read-Requests                           Write-Requests
Read-Bytes                              Write-Bytes
Read-Cbytes                    336 #    Write-Cbytes                 336 #
Read-Qtime                              Write-Qtime
Read-Qbusy-Time                         Write-Qbusy-Time
Read-Crequests                   6 #    Write-Crequests                6 #
------------------------------------------------------------------------
Server-Qtime
Retries                                 Acks
X-Defrd-Busy-Time                       Y-Defrd-Busy-Time
```

  This is the example from the Storage subsystem:

```
SAC   C1002531
Adpt  C1002531
CPU   6    Node Class  CLMS   GMS  100,2,5     Port  3   Fiber  1
Format Version:  H04  Data Version:  H04  Subsystem Version:  1
Local System \BLNSK2  From  26 Mar 2009, 17:13:10   For  33.6 Seconds
------------------------------------------------------------------------
Requests                     2,520 #    Total-IO-Bytes         82,281,472 #
IO-Qtime                      1.18 sec  IO-Qbusy-Time                1.17 sec
--------- Data into this CPU --------  --------- Data out from this CPU ----
Read-Requests                2,514 #    Write-Requests                 6 #
Read-Bytes              82,231,808 #    Write-Bytes                49,664 #
Read-Cbytes                             Write-Cbytes
Read-Qtime                    1.14 sec  Write-Qtime                  36.71 ms
Read-Qbusy-Time               1.14 sec  Write-Qbusy-Time             36.71 ms
Read-Crequests                          Write-Crequests
------------------------------------------------------------------------
Server-Qtime
Retries                                 Acks
X-Defrd-Busy-Time                       Y-Defrd-Busy-Time
```

Examples of Configuring Measurements for ServerNet Cluster

The ADD, DELETE, LIST, LISTACTIVE and LISTALL, commands can refer to one or more SERVERNET entities. Although this example uses the ADD command, the DELETE, LIST,

LISTACTIVE and LISTALL commands all share the same command syntax for configuring or displaying measurements on remote ServerNet cluster activity.

Command	Measures All Remote IPC Activity Between...
ADD SERVERNET *(RIPC)	All CPUs on the local system and all CPUs on all remote ServerNet clusters
ADD SERVERNET *(RIPC,*,4)	All CPUs on the local system and all CPUs on remote ServerNet node number 4
ADD SERVERNET *(RIPC,*,4,*,5)	All CPUs on the local system and CPU 5 on remote ServerNet node number 4
ADD SERVERNET *(RIPC,*,4)	CPU 1 on the local system and all CPUs on the remote ServerNet node number 4
ADD SERVERNET 1(RIPC,*,4,*,0)	CPU 1 on the local system and CPU 0 on the remote ServerNet node number 4

Examples of Configuring Measurements for CLIMs

The ADD, DELETE, LIST, LISTACTIVE and LISTALL, commands can refer to one or more SERVERNET entities. Although this example uses the ADD command, the DELETE, LIST, LISTACTIVE and LISTALL commands all share the same command syntax for configuring or displaying measurements on remote ServerNet cluster activity.

Command	Measures...
ADD SERVERNET *(CLIM)	All CLIMs on all CPUs
ADD SERVERNET 1 (CLIM)	All CLIMs in CPU 1
ADD SERVERNET * (CLMS,1,2,3)	All ports on all storage CLIMs with GMS equal to 1,2,3 in all CPUs
ADD SERVERNET * (CLMI,1,2,3,1)	Port 1 on all IP CLIMs with GMS equal to 1,2,3 in all CPUs
ADD SERVERNET * (CLMI,1,2,3,8,1)	Port 8, fiber 1 on all IP CLIMs with GMS equal to 1,2,3 in all CPUs
ADD SERVERNET * (CLMO)	All Telco CLIMs on all CPUs

SQLPROC

The SQLPROC entity type provides information about one or more SQL processes. There is one SQLPROC counter record for each SQL process.

In Measure G09 and later PVUs, the SQLPROC entity allows the use of OSS file pathnames.

Entity Specification Syntax for SQLPROC Entities

To describe the SQLPROC entity:

```
SQLPROC entity-spec
```

SQLPROC

collects information about one or more SQL processes on the local system.

entity-spec

is specified as:

```
{ *                          }
{ $process-name [ ( pid ) ] }
{ disk-filename [ ( pid ) ] }
{ "pname" [ ( pid ) ]        }
{ pid                        }
```

*

measures all SQL processes on the local system.

$*process-name*

is the name of the SQL process to be measured.

pid

is a process identifier. The process identifier identifies the owner of the SQL process to be measured. Specify *pid* by using these two variables:

```
{ cpu,pin }
```

cpu

is the number of the CPU on which the SQL process is running .

pin

is the process identification number of the SQL process. To indicate all PINs, use an asterisk (*).

disk-filename

is the name of the object file that contains the SQL/MP code to be measured. This file must be the object file of a running process. Specify the disk-file name as:

```
{ [ \system.]$device.subvol.filename[:CRVSN]     }
{ subvol-name.filename[:CRVSN]                   }
{ filename[:CRVSN]                               }
```

$*device*

is the volume (device) name. To indicate all volumes, use an asterisk (*). The default is the current default volume.

subvol

is the subvolume name. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume.

filename

is the file name. To indicate all files (except temporary files), use an asterisk (*).

:*CRVSN*

in Measure G10 and later, is the timestamp, creation version serial number, or file name extension, necessary to form a unique file name. Use this option to guarantee file name uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

"*pname*"

can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

DDL Record for SQLPROC Entities (Legacy Style)

This is the Legacy Style DDL record for SQLPROC entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

```
RECORD sqlproc. FILE is "sqlproc" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 priority                     type binary 16 unsigned.
 02 userid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
* counter value items:
 02 sql-obj-recompiles           type binary 16 unsigned.
 02 sql-obj-recompile-time       type binary 64.
 02 sql-stmt-recompiles          type binary 16 unsigned.
 02 sql-stmt-recompile-time      type binary 64.
 02 sql-newprocesses             type binary 16 unsigned.
 02 sql-newprocess-time          type binary 64.
 02 opens                        type binary 16 unsigned.
 02 open-time                    type binary 64.
 02 creatorid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
* new entity identification items for D10:
 02 ancestor-cpu                 type binary 16 unsigned.
 02 ancestor-pin                 type binary 16 unsigned.
 02 ancestor-sysname             type character 8.
 02 ancestor-process-name        type character 8.
* SMS changes:
 02 device-name                  type character 8.
* new identifiers for OSS file pathname support:
 02 osspid                       type binary 32 unsigned.
 02 program-file-name-mid.
    03 pathid                    type character 24.
    03 crvsn                     type character  6.
* New identifiers for NetBatch Job Control:
 02 GMOM.
    03 GMOM-node                 type binary 16 unsigned.
    03 GMOM-cpu                  type binary 16 unsigned.
    03 GMOM-pin                  type binary 16 unsigned.
    03 GMOM-jobid                type binary 16 unsigned.
 02 GMOM-full-id                 type binary 64
```

```
                                        redefines GMOM.
 02 GMOM-sysname                 type character 8.
 02 GMOM-process-name            type character 8.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for SQLPROC Entities (ZMS Style)

The ZMS style DDL record for SQLPROC entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmssqlp-id.
 02 pin                          type binary 16 unsigned.
 02 priority                     type binary 16 unsigned.
 02 userid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
 02 creatorid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 osspid                       type binary 32 unsigned.
 02 ancestor-cpu                 type binary 16 unsigned.
 02 ancestor-pin                 type binary 16 unsigned.
 02 ancestor-sysname             type character 8.
 02 ancestor-process-name        type character 8.
 02 device-name                  type character 8.
 02 program-file-name-MID.
    03 PATHID                    type character 24.
    03 CRVSN                     type character 6.
 02 reserved-1                   type character 2.
 02 GMOM.
    03 GMOM-node                 type binary 16 unsigned.
    03 GMOM-cpu                  type binary 16 unsigned.
    03 GMOM-pin                  type binary 16 unsigned.
    03 GMOM-jobid                type binary 16 unsigned.
 02 GMOM-full-id                 type binary 64
                                      redefines GMOM.
 02 GMOM-sysname                 type character 8.
 02 GMOM-process-name            type character 8.
end
```

Counter Fields DDL Definition

```
DEFINITION zmssqlp-ctrs.
 02 sql-obj-recompiles           type binary 64.
 02 sql-obj-recompile-time       type binary 64.
 02 sql-stmt-recompiles          type binary 64.
 02 sql-stmt-recompile-time      type binary 64.
 02 sql-newprocesses             type binary 64.
 02 sql-newprocess-time          type binary 64.
 02 opens                        type binary 64.
 02 open-time                    type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmssqlp. FILE is "zmssqlp" ENTRY-SEQUENCED.
 02 hdr                          type zmsheader.
 02 ctr                          type zmssqlp-ctrs.
 02 id                           type zmssqlp-id.
end
```

PIN

Process identification number of the measured SQL process.

PROCESS-NAME

Name of the SQL process.

PROGRAM-FILE-NAME

Name of the object file the process is executing. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, PROGRAM-FILE-NAME represents the physical file name that was specified when the process was executed. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, PROGRAM-FILE-NAME represents a location-independent logical file name that was used when the process was executed. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

PRIORITY

Creation priority of the process. Changing process priority using the ALTPRI command does not affect this value.

USERID

Process accessor ID (PAID) of the process.

SQL-OBJ-RECOMPILES

The number of objects recompiled for this process. SQL/MX does not support this counter.

Counter type: Incrementing.

SQL-OBJ-RECOMPILE-TIME

The time this process spent recompiling objects. SQL/MX does not support this counter.

Counter type: Elapsed.

SQL-STMT-RECOMPILES

Number of statements recompiled for this process. The SQL-STMT-RECOMPILES counter is incremented when a similarity check fails and automatic recompilation happens. Similarity checks and automatic recompilation are explained in the *SQL/MX Programming Manual for C and COBOL*.

Counter type: Incrementing.

SQL-STMT-RECOMPILE-TIME

The time this process spent recompiling statements.

Counter type: Elapsed.

SQL-NEWPROCESSES

Number of NEWPROCESS calls generated by the SQL executor on behalf of this process.

Counter type: Incrementing.

SQL-NEWPROCESS-TIME

The time this process spent doing SQL-NEWPROCESS.

Counter type: Elapsed.

OPENS

Number of OPEN calls performed by the SQL executor on behalf of this process.

Counter type: Incrementing.

OPEN-TIME

The time this process spent doing OPENs.

Counter type: Elapsed.

CREATORID

Creator access ID (CAID). This field identifies the creator of the process and is divided into two subfields: GROUP and USER.

ANCESTOR-CPU

Number of the CPU on which the ancestor process resides.

ANCESTOR-PIN

Process identification number of the ancestor process.

ANCESTOR-SYSNAME

Name of the system on which the ancestor process resides.

ANCESTOR-PROCESS-NAME

Name of the ancestor process. The report header displays the ancestor information. If this information cannot be obtained, the string “unknown” appears.

DEVICE-NAME

Disk device on which the program file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the PROGRAM-FILE-NAME VOLUME subfield.

OSSPID

If the process runs in the OSS environment, this field contains the OSS process ID. For non-OSS processes, this field contains zeros.

PROGRAM-FILE-NAME-MID

FILE-NAME-MID has two subfields: PATHID and CRVSN. PATHID is an internal format representation of an OSS file pathname. For non-OSS files, this field contains zeros. CRVSN is a creation version serial number that identifies a unique instance of an OSS file. For non-OSS files, this field contains zeros.

GMOM-NODE

The Expand node number of the GMOM process if this process is part of a NETBATCH job and if the GMOM process is remote; otherwise zero.

GMOM-CPU

The processor number of the GMOM process, if this process is part of a NETBATCH job; otherwise zero.

GMOM-PIN

The PIN of the GMOM process if this process is part of a NETBATCH job; otherwise zero.

GMOM-JOBID

The job ID of the NETBATCH job that initiated this process, assigned by the indicated GMOM; otherwise zero.

GMOM-FULL-ID

A 64-bit redefinition of the individual GMOM fields as a single value. Zero indicates the process is not part of a NETBATCH job stream.

GMOM-SYSNAME

The Expand system name of the GMOM node if the GMOM process is remote; otherwise spaces.

GMOM-PROCESS-NAME

The name of the GMOM process that initiated the NETBATCH job that in turn initiated this process; otherwise spaces.

Usage Note for New Format SQLPROC Entities

SUBSYSTEM-VERSION for ZMSSQLP records is provided by the SQL/MP or SQL/MX subsystem.

SQLSTMT

The SQLSTMT entity type provides information about all SQL statements within the SQL process. There is one SQLSTMT counter record for each SQL statement in each procedure or module of a program that contains SQL function.

NOTE: Overhead cost for measuring SQL statements is higher than the cost of measuring other entities. A practical use is to measure SQLSTMT periodically for application-program performance tuning only. Be sure to configure SQLSTMT to measure only the programs to be examined.

In Measure G09 and later PVUs, SQLSTMT entity operation supports SQL/MX and OSS file pathnames. SQL/MX program objects and run units are defined only in the OSS file system space. In the SQL/MP environment, SQL procedures and run units are part of the program object file that executes them. In SQL/MX, however, SQL procedures and modules are compiled and stored separately from the program object file that executes them.

In Measure G11 and later PVUs, the SQLSTMT entity displays ANSI SQL names in SQLSTMT reports. In Measure H01 and later PVUs, MEASCOM LIST commands accept ANSI SQL/MX release 2 module object names (run unit names) in SQLSTMT entity specifications.

SQL/MX module names (previously called run unit names), can be:

Measure PVUs	Length (In Characters)
G10 and earlier	Up to 128
G11 and later	Up to 784

If specific module names are not specified in SQL/MX, a system-generated name based on the Julian timestamp at the time of preprocessing will be assigned. For information regarding the naming and indexing of statements in SQL/MX modules, see the preprocessor output file.

Entity Specification Syntax for SQLSTMT Entities

To describe the SQLSTMT entity:

```
SQLSTMT entity-spec
```

SQLSTMT

collects information about SQL statements within the SQL process.

entity-spec

For the ADD and DELETE commands, specify *entity-spec* as:

```
{ *                                                  }
{ $process-name [ ( pid ) ]                          }
{ disk-filename [ ( pid ) ]                          }
{ pid                                                }
```

For the LIST command, specify *entity-spec* as:

```
{* ["run unit"[,#index]]                                          }
{* ["run unit"[[Index]#index]]                                    }
{ $process-name  [ ( pid ) ] ["run unit"[,#index]]               }
{ $process-name  [ ( pid ) ] ["run unit"[Index]#index]]          }
{ disk-filename [ ( pid ) ] ["run unit"[,#index]]    }
{ disk-filename [ ( pid ) ] ["run unit"[[Index] #index]]}
{ "pname" [ ( pid ) ] ["run unit"[,#index]]                      }
{ "pname" [ ( pid ) ] ["run unit"[[Index] #index]]               }
{ pid ["run unit"[,#index]]}
{ pid ["run unit"[[Index] #index]]                               }
```

For the LISTACTIVE command, specify *entity-spec* as:

```
{ $process-name ( pid ) "run unit", #index                       }
{ $process-name ( pid ) "run unit" [Index] #index                }
{ disk-filename ( pid ) "run unit", #index                       }
{ disk-filename ( pid ) "run unit", [Index] #index               }
{ "pname" (pid) "run unit", #index                                }
{ "pname" (pid) "run unit" [Index] #index                        }
{ pid "run unit", #index                                         }
{ pid "run unit" [Index] #index                                  }
```

*

measures all SQL statements in all processes on the local system.

$*process-name*

is the name of the process that contains the SQL statements to be measured.

pid

is a process identifier. The process identifier identifies the owner of the process to be measured. Specify *pid* by using these two variables:

```
{ cpu,pin }
```

cpu

is the number of the CPU on which the process is running.

pin

is the process identification number of the process. To indicate all PINs, use an asterisk (*).

run-unit

in pre-G09 Measure PVUs, is the name of a procedure to be measured. For example:

```
LIST SQLSTMT * "max32bytelongrununitname"
```

in Measure G09 and later PVUs:

- is the SQL/MP procedure name of up to 32 characters, enclosed in double quotes. For example:

  ```
  LIST SQLSTMT * "max32bytelongrununitname"
  LISTACTIVE SQLSTMT 3,99 "max32bytelongrununitname", #1
  ```

- is the SQL/MX Release 1 module name of up to 128 characters, enclosed in double quotes. For example:

  ```
  LIST SQLSTMT * "max128bytelongrununitname"
  LISTACTIVE SQLSTMT 3,99 "max128bytelongrununitname", #1
  ```

in Measure H01 and later PVUs:

- is the SQL/MX Release 2 module name with only undelimited identifiers of up to 128 bytes, enclosed in double quotes. For example:

  ```
  LIST SQLSTMT * "catA.schemaB.moduleM"
  ```

- is the SQL/MX Release 2 module name with a maximum length as supported by SQL/MX (currently up to 783 characters), enclosed in single quotes. For example:

  ```
  LIST SQLSTMT * 'catA."schema%B".moduleM'
  LIST SQLSTMT * 'moduleM'
  LISTACTIVE SQLSTMT 3,99 'catA."schema%B".moduleM', #1
  ```

Index

is optional. For Measure G09 and later PVUs, if used, *Index* enables use of the run unit and index values (displayed in SQLSTMT reports) for cut-and-paste applications.

index

is the index number of the statement to be measured. If omitted, all current indexed statements are measured.

disk-filename

is the name of an object file that contains SQL code to be measured. This file must be the object file of a running process. Specify the disk-file name as:

```
{ [ \system.]$device.subvol.filename[:CRVSN]}
{ subvol.filename[:CRVSN]                   }
{ filename[:CRVSN]                          }
```

system

is the system name. To indicate all devices, use an asterisk (*). The default is the current default system.

$*device*

is the volume (device) name. To indicate all volumes, use an asterisk (*). The default is the current default volume.

subvol

is the subvolume name. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume.

filename

is the object file name. To indicate all files (except temporary files), use an asterisk (*).

:*CRVSN*

in Measure G10 and later PVUs, is the timestamp, creation version serial number, or file name extension necessary to form a unique file name. Use this option to guarantee file name uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

"*pname*"

in Measure G09 and later PVUs, can be either a fully qualified or partial OSS file pathname. To get a fully qualified and valid OSS file pathname, combine *pname* with the current OSSPATH setting. If the OSS file pathname does not begin with a backslash (/), expand it by putting the current setting for OSSPATH in front of the backslash.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

DDL Record for SQLSTMT Entities (Legacy Style)

This is the Legacy Style DDL record for SQLSTMT entities. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface type**.

```
RECORD sqlstmt. FILE is "sqlstmt" ENTRY-SEQUENCED.
                      .
                      .
                      .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                      .
                      .
                      .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 priority                     type binary 16 unsigned.
 02 userid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
* counter record identifiers and counter values:
 02 run-unit                     type character 32.
 02 statement-index              type binary 16 unsigned.
 02 calls                        type binary 32.
 02 elapsed-busy-time            type binary 64.
 02 records-used                 type binary 32 unsigned.
 02 records-accessed             type binary 32 unsigned.
 02 disc-reads                   type binary 32 unsigned.
 02 messages                     type binary 32 unsigned.
 02 message-bytes                type binary 32 unsigned.
 02 sorts                        type binary 16 unsigned.
 02 elapsed-sort-time            type binary 64.
 02 recompiles                   type binary 16 unsigned.
 02 elapsed-recompile-time       type binary 64.
 02 lock-waits                   type binary 16 unsigned.
 02 timeouts                     type binary 16 unsigned.
 02 escalations                  type binary 16 unsigned.
 02 creatorid.
    03 group                     type binary 8 unsigned.
    03 user                      type binary 8 unsigned.
* new entity identification items for D10:
 02 ancestor-cpu                 type binary 16 unsigned.
 02 ancestor-pin                 type binary 16 unsigned.
```

```
 02 ancestor-sysname           type character 8.
 02 ancestor-process-name      type character 8.

* SMS changes:
 02 device-name                  type character 8.

* F40 new counter value items:
 02 message-bytes-f             type binary 64.
* G02 new counter value items:
 02 calls-f                     type binary 64.
 02 records-used-f              type binary 64.
 02 records-accessed-f          type binary 64.
 02 disc-reads-f                type binary 64.
 02 messages-f                   type binary 64.*

New identifiers for OSS file pathname suport:
 02 osspid                     type binary 32 unsigned.
 02 program-file-name-mid.
    03 pathid                  type character 24.
    03 crvsn                   type character  6.

* New identifiers for SQL/128Support
 02 run-unit-128                type character 128.

* New identifiers for NetBatch Job Control:
 02 GMOM.
    03 GMOM-node                type binary 16 unsigned.
    03 GMOM-cpu                 type binary 16 unsigned.
    03 GMOM-pin                 type binary 16 unsigned.
    03 GMOM-jobid               type binary 16 unsigned.
 02 GMOM-full-id                type binary 64
                                   redefines GMOM.
 02 GMOM-sysname                type character 8.
 02 GMOM-process-name           type character 8.

* New Version Field
 02 VERSION                     type binary 16 unsigned.
 02 CREATOR-TYPE                type binary 16 unsigned.

* New identifiers for full ANSI SQL name
 02 FULL-NAME-OFFSET            type binary 16 unsigned.
 02 FULL-NAME-LEN               type binary 16 unsigned.

* Variable data goes here, pointed to by VARSTRING (offset/length)

* VAR-DATA needs to be defined for the maximum expected variable length data.

 02 FULL-NAME                   type character 784.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for SQLSTMT Entities (ZMS Style)

The ZMS style DDL record for SQLSTMT entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmssqls-id.
 02 pin                          type binary 16 unsigned.
 02 priority                     type binary 16 unsigned.
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 process-name                type character 8.
```

```
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 osspid                       type binary 32 unsigned.
 02 ancestor-cpu                 type binary 16 unsigned.
 02 ancestor-pin                 type binary 16 unsigned.
 02 ancestor-sysname             type character 8.
 02 ancestor-process-name        type character 8.
 02 device-name                  type character 8.
 02 program-file-name-MID.
    03 PATHID                    type character 24.
    03 CRVSN                     type character 6.
 02 statement-index              type binary 16 unsigned.
 02 GMOM.
    03 GMOM-node                 type binary 16 unsigned.
    03 GMOM-cpu                  type binary 16 unsigned.
    03 GMOM-pin                  type binary 16 unsigned.
    03 GMOM-jobid                type binary 16 unsigned.
 02 GMOM-full-id                 type binary 64
                                      redefines GMOM.
 02 GMOM-sysname                 type character 8.
 02 GMOM-process-name            type character 8.
 02 version                      type binary 16 unsigned.
 02 creator-type                 type binary 16 unsigned.
 02 run-unit-len                 type binary 16 unsigned.
 02 reserved-1                   type character 2.
 02 run-unit                     type character 784.
end
```

Counter Fields DDL Definition

```
DEFINITION zmssqls-ctrs.
 02 calls                        type binary 64.
 02 elapsed-busy-time            type binary 64.
 02 records-used                 type binary 64.
 02 records-accessed             type binary 64.
 02 disc-reads                   type binary 64.
 02 messages                     type binary 64.
 02 message-bytes                type binary 64.
 02 sorts                        type binary 64.
 02 elapsed-sort-time            type binary 64.
 02 recompiles                   type binary 64.
 02 elapsed-recompile-time       type binary 64.
 02 lock-waits                   type binary 64.
 02 timeouts                     type binary 64.
 02 escalations                  type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmssqls. FILE is "zmssqls" ENTRY-SEQUENCED.
 02 hdr                          type zmsheader.
 02 ctr                          type zmssqls-ctrs.
 02 id                           type zmssqls-id.
end
```

PIN

Process identification number of the measured process.

PROCESS-NAME

Name of the measured process.

PROGRAM-FILE-NAME

Name of the object file for the measured process. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, PROGRAM-FILE-NAME represents the physical file name that was specified when the process was executed. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, PROGRAM-FILE-NAME represents a location-independent logical file name that was used when the process was executed. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

PRIORITY

Creation priority of the measured process. Changing a process priority using the ALTPRI command does not affect this value.

USERID

Process accessor ID of the measured process.

RUN-UNIT

Name of the procedure that contains the measured SQL statement. For SQL/MX, if the actual name of the procedure exceeds the 32 characters allowed, a field overflow is reported in the ERROR field and only the least-significant 31 bytes (preceded by a tilde (~)) are included. You can obtain the full name from the run-unit-128 field. See DDL Record for SQLSTMT Entities (Legacy Style) (page 322).

STATEMENT-INDEX

Statement index number (0-255) within the specified run unit. These numbers correspond to the section location table (SLT) index numbers on compiler listings. Use the SLT index to correlate the STATEMENT-INDEX number with the corresponding EXEC SQL statement in the source code.

To determine the SLT index number for each SQL statement, use the SQLMAP option of the SQL compiler directive. The SQLMAP option causes the SQL map to appear at the end of the listing. The SQL map is a table that shows the SLT index number for each SQL statement, along with the source file name and the line number.

For more information, see the SQL/MP programming manual for the language you are using.

CALLS

Number of times the measured SQL statement was executed.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the CALLS-F field is a 64-bit version of CALLS.

Counter type: Incrementing.

ELAPSED-BUSY-TIME

The time the process spent executing the measured SQL statement.

If REPORT RATE is on, the output value is the percent of time busy during the interval.

You can calculate the average elapsed time for each call by dividing the ELAPSED-BUSY-TIME counter value by the CALLS counter value.

Counter type: Elapsed.

RECORDS-USED

Number of records inserted, updated, deleted, or read by the SQL executor through this statement.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the RECORDS-USED-F field is a 64-bit version of RECORDS-USED.

Counter type: Accumulating.

RECORDS-ACCESSED

Number of records accessed by the disk process or file system to evaluate this statement.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the RECORDS-ACCESSED-F field is a 64-bit version of RECORDS-ACCESSED.

Counter type: Accumulating.

DISC-READS

Number of physical disk reads performed for this statement. If the REPORT RATE value is on, the rate is disk reads for each second.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the DISC-READS-F field is a 64-bit version of DISC-READS.

Counter type: Incrementing.

MESSAGES

Number of messages sent by the system on behalf of this statement, including messages sent by the file system to the disk process and messages sent by the SQL executor to the SQL compiler for recompiles.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the MESSAGES-F field is a 64-bit version of MESSAGES.

Counter type: Incrementing.

MESSAGE-BYTES

Number of message bytes sent for this statement by the system. This counter accumulates the number of bytes sent for the messages reported by the MESSAGES counter.

For G-series and earlier RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the MESSAGE-BYTES-F field is a 64-bit version of MESSAGE-BYTES.

Counter type: Accumulating.

SORTS

Number of times an external sort process was invoked for this statement.

Counter type: Incrementing.

ELAPSED-SORT-TIME

The time spent by the process on sorts for this statement.

Counter type: Elapsed.

RECOMPILES

Number of times the statement was recompiled.

Counter type: Incrementing.

ELAPSED-RECOMPILE-TIME

The time the process spent recompiling this statement.

Counter type: Elapsed.

LOCK-WAITS

Number of times this statement waited for a lock request.

Counter type: Incrementing.

TIMEOUTS

Number of times a request timed out. A timeout is caused by either a lock-wait or a congested disk process or network.

Counter type: Incrementing.

ESCALATIONS

Number of times a record lock was escalated to a file lock.

Counter type: Incrementing.

CREATORID

Creator access ID (CAID). This field identifies the creator of the process and is divided into two subfields: GROUP and USER.

ANCESTOR-CPU

Number of the CPU on which the ancestor process resides.

ANCESTOR-PIN

Process identification number of the ancestor process.

ANCESTOR-SYSNAME

Name of the system on which the ancestor process resides.

ANCESTOR-PROCESS-NAME

Name of the ancestor process.

DEVICE-NAME

Disk device on which the program file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

MESSAGE-BYTES-F

(G-series, Legacy Style only) Same as MESSAGE-BYTES but accommodates larger values (64 bits rather than 32).

CALLS-F

(G-series, Legacy Style only) Same as CALLS but accommodates larger values (64 bits rather than 32).

RECORDS-USED-F

(G-series, Legacy Style only) Same as RECORDS-USED but accommodates larger values (64 bits rather than 32).

RECORDS-ACCESSED-F

(G-series, Legacy Style only) Same as RECORDS-ACCESSED but accommodates larger values (64 bits rather than 32).

DISC-READS-F

(G-series, Legacy Style only) Same as DISC-READS but accommodates larger values (64 bits rather than 32).

MESSAGES-F

(G-series, Legacy Style only) Same as MESSAGES but accommodates larger values (64 bits rather than 32).

OSSPID

If the process runs in the OSS environment, this field contains the OSS process ID. If OSSPID is a non-OSS process, this field contains zeros.

PROGRAM-FILE-NAME-MID

FILE-NAME-MID has two subfields: PATHID and CRVSN. PATHID is an internal format representation of an OSS file pathname. For non-OSS files, this field contains zeros. CRVSN is a creation version serial number that identifies a unique instance of an OSS file. For non-OSS files, this field contains zeros.

RUN-UNIT-128

(Legacy Style only) Name of the SQL/MP procedure or SQL/MX module that contains the measured SQL statement.

GMOM-NODE

The Expand node number of the GMOM process if this process is part of a NETBATCH job and if the GMOM process is remote; otherwise zero.

GMOM-CPU

The processor number of the GMOM process if this process is part of a NETBATCH job; otherwise zero.

GMOM-PIN

The PIN of the GMOM process if this process is part of a NETBATCH job; otherwise zero.

GMOM-JOBID

The job ID of the NETBATCH job that initiated this process, assigned by the indicated GMOM; otherwise zero.

GMOM-FULL-ID

A 64-bit redefinition of the individual GMOM fields as a single value. Zero indicates the process is not part of a NETBATCH job stream.

GMOM-SYSNAME

The Expand system name of the GMOM node if the GMOM process is remote; otherwise spaces.

GMOM-PROCESS-NAME

The name of the GMOM process that initiated the NETBATCH job that in turn initiated this process; otherwise spaces.

VERSION

The record version: 0, 1, 2 = not used. 3 = G11 PVU.

CREATOR-TYPE

The creator type: 0 = not used. 1 = SQL/MP. 2 = SQL/MX 1.*x*. 3 = SQL/MX 2.*x*.

RUN-UNIT-LEN

The length, in bytes, of the external-format ANSI SQL name stored in the RUN-UNIT field. The maximum value is 783.

RUN-UNIT

The left-justified external-format ANSI SQL name. The field is defined as 784 characters (rounded up for alignment reasons), but the length of the actual name is reported in RUN-UNIT-LEN. Remaining characters are initialized with zeros.

FULL-NAME-OFFSET

The offset, in bytes, from the beginning of the SQLSTMT record of the FULL-NAME field.

FULL-NAME-LEN

The length, in bytes, of the external-format ANSI SQL name stored in the FULL-NAME field. The maximum value is 783.

FULL-NAME

The left-justified external-format ANSI SQL name. This field is defined as 784 characters, but the length of the actual name is reported in FULL-NAME-LEN. Remaining characters are undefined.

Usage Notes for All SQLSTMT Entities

- You do not need to explicitly add the SQLPROC entity for the SQLSTMT entity. When you add the SQLSTMT entity to a measurement, the corresponding SQLPROC entity is automatically added for the measured process. You can use the SQLPROC entity to get more information about the measured SQL process. However, Measure does not automatically add the PROCESS entity when the SQLSTMT entity is added. If PROCESS entity information is also required, you must specifically add the PROCESS entity.
- SUBSYSTEM-VERSION for ZMSSQLS records is provided by the SQL/MP or SQL/MX subsystem.

  In G-series, the 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field MESSAGE-BYTES-F collects the same data as the 32-bit field MESSAGE-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. (A field overflow is indicated by a value of -1 in both the field that has overflowed and in the ERROR field for the measured entity.) However, you should convert your applications to use the 64-bit fields. 32-bit fields might be deactivated in a future RVU.

  In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST DEVICE BY INPUT-BYTES, not LIST DEVICE BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

- In H-series and J-series, all byte-count fields accommodate 64 bits. Fields ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.
- When specifying filenames, you can use an asterisk as a wildcard, provided that the asterisk replaces the entire element (volume, subvolume, or filename):

  * or `$*.*.*` means all files.

  `$vol.*.*` means all files in the specified volume.

  `$vol.subvol.*` means all files in the specified subvolume, in the specified volume.

  `$*.subvol.*` means all files in the specified subvolume, in any volume.

  `$vol.*.file` means the specified file, in any subvolume in the specified volume.

  `$*.*.file` means the specified file in any subvolume in any volume.

Example

The report output for the SQLSTMT entity includes the SQL run unit name in double-quotes or the ANSI SQL module name in single quotes depending on the name type. For example:

```
SQL Statement
Procedure 'CATALOG_12.SCHEMA_34.MODULE_56'   Index #
Process                        Pri           OSSPID:
Program
OSSPath:
Userid             Creatorid           Ancestor
Format Version:  H01  Data Version:  H01  Subsystem Version:  1
Local System
------------------------------------------------------------------------------
Elapsed-Busy-Time            4.17 ms   Calls                          0.17 /s
Elapsed-Sort-Time                      Sorts
Elapsed-Recompile-Time                 Recompiles
Messages                     0.22 /s   Message-Bytes                931.00 /s
Lock-Waits                             Escalations
Timeouts                               Disc-Reads
Records-Accessed             0.17 /s   Records-Used                   0.17 /s
```

SYSTEM

The SYSTEM entity type provides information about network traffic through Expand line handlers. A network link involves at least two processes: the linker that initiated the link and the listener that accepted the link. Because the Measure subsystem can measure only the local system, the measurement data reflects only one side of each link: the linker or the listener, but not both.

Which network links are measured depends on the entity specification in the measurement configuration:

- If you specify all systems, the Measure subsystem counts the local end of all messages passed between the local system and any remote system.
- If you specify a local CPU, the Measure subsystem counts the local end of all messages passed between the local system and any remote system through the line-handler processes (primary or backup) in the specified CPU.
- If you specify a remote system, the Measure subsystem counts the local end of all messages passed between the local system and the remote system.

Entity specification syntax and DDL record Legacy Styles are the same for G-series PVU SYSTEM entities as for D-series PVU SYSTEM entities.

Entity Specification Syntax for SYSTEM Entities

To describe a SYSTEM entity:

```
SYSTEM entity-spec
```

SYSTEM

collects information about network traffic.

entity-spec

is specified as:

```
{ *                    }
{ \system [ ( cpu ) ] }
```

*

measures traffic between the local system and all remote systems.

system

is the system name or system number of a remote system for which to measure network traffic.

cpu

is the number of a CPU on the remote system for which to measure network traffic. Use an asterisk (*) to indicate all CPUs. The default is all CPUs.

DDL Record for All SYSTEM Entities (Legacy Style)

This is the Legacy Style SYSTEM DDL record. This record will not change after the G10 Measure PVU.

```
RECORD system. File is "system" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 remote-system              type binary 16 unsigned.
 02 remote-system-name          type character 8.
* counter value items:
 02 links                      type binary 32 unsigned.
 02 link-time                  type binary 64.
 02 sent                       type binary 32 unsigned.
 02 received                   type binary 32 unsigned.
 02 sent-forward               type binary 32 unsigned.
 02 received-forward           type binary 32 unsigned.

end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for All SYSTEM Entities (ZMS Style)

The ZMS style SYSTEM DDL record is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface**.

ID Fields DDL Definition

```
DEFINITION zmssys-id.
 02 remote-system-name          type character 8.
 02 remote-system               type binary 16 unsigned.
 02 reserved-1                  type character 6.
end
```

Counter Fields DDL Definition

```
DEFINITION zmssys-ctrs.
 02 links                       type binary 64.
 02 link-time                   type binary 64.
 02 sent                        type binary 64.
 02 received                    type binary 64.
 02 sent-forward                type binary 64.
 02 received-forward            type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmssys. FILE is "zmssys" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmssys-ctrs.
 02 id                          type zmssys-id.
end
```

REMOTE-SYSTEM

System number of the remote system for which network traffic was measured.

REMOTE-SYSTEM-NAME

System name of the remote system for which network traffic was measured.

LINKS

Number of messages (each consisting of one or more packets) sent from the local system to the remote system. This counter does not include level 4, file-system interface, acknowledgment messages.

Counter type: Incrementing.

LINK-TIME

The time spent waiting on acknowledgment from the remote system of messages sent to it by the local system.

Counter type: Syslink.

SENT

Number of packets created and sent by the local system to the remote system. The SENT-FORWARD counter counts packets sent by the local system on behalf of another system in the network.

Counter type: Incrementing.

RECEIVED

Number of packets sent from the remote system to the local system with the local system as their final destination. The RECEIVED-FORWARD counter counts packets whose destination is a third system.

Counter type: Incrementing.

SENT-FORWARD

Number of packets sent by the local system to the remote system on behalf of another system in the network.

Counter type: Incrementing.

RECEIVED-FORWARD

Number of packets sent from the remote system to the local system to be forwarded to another system in the network. The RECEIVED counter counts packets whose final destination is the local system.

Counter type: Incrementing.

Usage Notes for All SYSTEM Entities

- The SYSTEM counters are modified by network line-handler processes during network operations between systems linked by Expand line handlers. Such operations are logging on to a remote system or moving a file to a remote system. The protocol determines the Expand line-handler process used.
- SUBSYSTEM-VERSION for ZMSSYS records is provided by Expand.
- These entity types are related:
  - CLUSTER measures FOX traffics.
  - NETLINE measures network communication lines.
- For information on networks and network protocols, see the *Expand Configuration and Management Manual*, the *Expand Network Management and Troubleshooting Manual*, and the *X25AM Configuration and Management Manual*.

TERMINAL

The TERMINAL entity type provides information about terminal I/O.

Entity Specification Syntax for TERMINAL Entities

To describe TERMINAL entities:

```
TERMINAL entity-spec
```

NOTE: You can measure G-series PVU TERMINAL entities with a D-series RVU measurement application if it specifies all TERMINAL entities (ADD TERMINAL *) or only *$device* or *$device (cpu)*, with or without a subdevice name. If the application specifies *controller*, *channel*, or *unit*, you must modify the entity identifiers to measure G-series PVU TERMINAL entities.

TERMINAL

collects information about one or more terminals on the local system.

entity-spec

is specified as:

```
{*                                                          }
{$line[.#subdev ] [ ( cpu [, chan [ , ctrl [ , unit ]]])]}
{ cpu [ ( %Htrackid [, clip [, line ] ] ) ] [ ( type [, subtype ] ) ] }
*
```

measures all lines (terminal connections) in all processors.

$*line*

is the device (line) name of the line to be measured. To indicate all lines, use an asterisk (*).

#subdev

is the subdevice name of the measured line. To indicate all subdevices, use an asterisk (*).

cpu

is the number of the CPU in which the measured line is configured. The default is all CPUs.

chan

(D-series) is the channel number of the measured terminal. The default is all channels.

ctrl

(D-series) is the controller number of the measured terminal. The default is all controllers.

unit

(D-series) is the unit number of the measured terminal. The default is all units.

%Htrackid

(G-series) is the identifier of a specific 3880 SWAN concentrator enclosure of three CLIPs. (Each CLIP controls two lines.) To indicate all TRACKIDs, use an asterisk (*).

The TRACKID is recorded in the SEEPROM of the controller. It is generally printed on an external label on the enclosure as well.

clip

(G-series) is 1, 2, or 3 to indicate a specific CLIP within a SWAN concentrator enclosure or 1, 2, 3, 4, 5, or 6 to indicate a specific CLIP within a SWAN 2 concentrator enclosure. To indicate all CLIPs, use an asterisk (*).

line

(G-series) is 0 or 1 to indicate a specific line managed by a CLIP in a 3880 SWAN concentrator. To indicate all lines, use an asterisk (*).

type

(G-series) is a type number to indicate a particular protocol, such as 61 (X.25). Use SCF LISTDEV to see valid values for line types. To indicate all line types, use an asterisk (*).

subtype

(G-series) is a subtype number, such as 62 (X25AM). To see valid subtype numbers, use SCF LISTDEV. The default is all subtypes that apply to *type*.

DDL Record for TERMINAL Entities (Legacy Style)

This is the Legacy Style DDL record for TERMINAL entities. This record will not change after the G10 Measure PVU.

The Legacy Style DDL record for G-series TERMINAL entities through the G10 Measure PVU is identical to the record for D-series TERMINAL entities except:

- Longer byte-count fields are provided to reduce the possibility of field overflow. In G-series RVUs, each 32-bit byte-count field has a 64-bit counterpart. For information on using the 64-bit fields, see Usage Notes for G-Series TERMINAL Entities (page 337).
- The ERROR field can signal a field overflow in a 32-bit byte-count field.

```
RECORD terminal. FILE is "terminal" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 terminal-name.
    03 line-name                type character 8.
    03 subdev-name              type character 8.
 02 logical-device              type binary 16 unsigned.
* counter value items:
 02 requests                    type binary 32 unsigned.
 02 reads                       type binary 32 unsigned.
 02 writes                      type binary 32 unsigned.
 02 input-bytes                 type binary 32 unsigned.
 02 output-bytes                type binary 32 unsigned.
 02 transactions                type binary 32 unsigned.
 02 response-time               type binary 64.
* F40 new counter value items:
 02 input-bytes-f               type binary 64.
 02 output-bytes-f              type binary 64.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for TERMINAL Entities (ZMS Style)

The ZMS style DDL record for TERMINAL entities is supported in Measure G11 and later PVUs.

The fields included in BRIEF reports are in **boldface type**.

ID Fields DDL Definition

```
DEFINITION zmsterm-id.
 02 terminal-name.
    03 line-name                type character 8.
    03 subdev-name              type character 8.
 02 logical-device              type binary 32 unsigned.
 02 reserved-1                  type character 4.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsterm-ctrs.
 02 requests                    type binary 64.
 02 reads                       type binary 64.
 02 writes                      type binary 64.
 02 input-bytes                 type binary 64.
 02 output-bytes                type binary 64.
 02 transactions                type binary 64.
 02 response-time               type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsterm. FILE is "zmsterm" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmsterm-ctrs.
 02 id                          type zmsterm-id.
end
```

TERMINAL-NAME

Device name of the measured terminal. This field is divided into two subfields: LINE-NAME (a device name) and SUBDEV-NAME (a subdevice name).

LOGICAL-DEVICE

Logical device number of the measured terminal.

REQUESTS

Number of requests received by the terminal process.

To determine how long the requests were queued before being read by the terminal process, measure the process and examine the PROCESS RECV-QTIME counter.

Counter type: Incrementing.

READS

Number of read operations (from the terminal to memory) performed by the terminal process.

Counter type: Incrementing.

WRITES

Number of write operations (from memory to the terminal) performed by the terminal process.

Counter type: Incrementing.

INPUT-BYTES

Number of bytes read from the terminal. The total number of bytes transferred to and from the terminal is this counter plus OUTPUT-BYTES. For SNAX lines, this counter includes the 6-byte transmission header (TH) and the 3-byte request/response (RH) SNA headers.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the INPUT-BYTES-F field is a 64-bit version of INPUT-BYTES.

Counter type: Accumulating.

OUTPUT-BYTES

Number of bytes written to the terminal.

The total number of bytes transferred to and from the terminal is INPUT-BYTES plus this counter.

Because the terminal process modifies this counter before the I/O operation, this counter might not be accurate if the write instruction fails.

For SNAX lines, this counter includes the 6-byte transmission header (TH) and the 3-byte request/response (RH) SNA headers.

For D-series and G-series RVUs, this is a 32-bit counter. For H-series and J-series RVUs, it is a 64-bit counter.

For G-series RVUs, the OUTPUT-BYTES-F field is a 64-bit version of OUTPUT-BYTES.

Counter type: Accumulating.

TRANSACTIONS

Number of terminal transactions (a read followed immediately by a write) performed by the terminal process.

RESPONSE-TIME measures the time spent on the transactions.

For SNAX lines, this counter is always zero.

Counter type: Incrementing.

RESPONSE-TIME

The time the terminal process spent on terminal response.

A transaction is a read from a terminal followed by a write to the terminal. Terminal response is the interval between the end of the read and the beginning of the write.

Block mode operations read a function key and then read data. This counter does not include the time between the two reads.

For SNAX lines, this counter is always zero.

Counter type: Response.

INPUT-BYTES-F

(G-series, Legacy Style only) Same as INPUT-BYTES but accommodates larger values (64 bits rather than 32 bits).

OUTPUT-BYTES-F

(G-series, Legacy Style only) Same as OUTPUT-BYTES but accommodates larger values (64 bits rather than 32 bits).

Usage Notes for All TERMINAL Entities

When you use the TERMINAL entity to measure subdevices on an X25AM communications line, the subdevices can also be measured as FILE entities.

Usage Notes for G-Series TERMINAL Entities

- SUBSYSTEM-VERSION for ZMSTERM records is provided by the specific product subsystem of the line represented by each ZMSLINE record; for example, SNAX/XF, X25AM, or AM3270.
- The 64-bit byte-count fields (fields ending in -F) collect the same data as older 32-bit byte-count fields. For example, the 64-bit field INPUT-BYTES-F collects the same data as the 32-bit field INPUT-BYTES. The 64-bit fields are less subject to overflow caused by high levels of I/O activity.

  The 32-bit fields are currently active and continue to return values. If no field overflow exists, the 32-bit fields and the 64-bit fields return the same value. If a 32-bit field overflows, the corresponding 64-bit field returns the correct value, and the 32-bit field returns a value of -1. The ERROR field for the measured entity also returns -1 to indicate an overflow condition.

Convert your applications to use the 64-bit fields; 32-bit fields might be deactivated in a future RVU.

In MEASCOM commands and in command (OBEY) files, use the names of the 32-bit fields. For example, issue the command LIST TERMINAL BY INPUT-BYTES, not LIST TERMINAL BY INPUT-BYTES-F. MEASCOM uses the names of the 32-bit fields in output displays such as reports and plots.

Measure G09 and later PVUs support up to six CLIPs in a SWAN concentrator.

Usage Notes for H-Series and J-Series TERMINAL Entities

In H-series and J-series RVUs, all byte-count fields accommodate 64 bits. Field names ending in -F are no longer used in ZMS style records but remain available to applications that request data in legacy style.

TMF

The TMF entity type provides information about TMF transactions.

Entity Specification Syntax for TMF Entities

To describe a TMF entity:

```
TMF entity-spec
```

TMF

collects information about TMF transactions in one or more CPUs on the local system.

entity-spec

is specified as:

```
{ *   }
{ cpu } [ , cpu ]...
```

*

measures TMF activity on all CPUs.

cpu

is the number of the CPU on which TMF transactions are measured. Use a comma-separated list to specify multiple CPUs.

DDL Record for TMF Entities (Legacy Style)

This is the Legacy Style TMF DDL record. This record will not change after the G10 Measure PVU.

The fields included in BRIEF reports are in **boldface**.

```
RECORD tmf. FILE is "tmf" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
```

```
                        .
                        .
* entity identification items:
  (none)

* counter value items:
 02 home-trans                  type binary 32 unsigned.
 02 home-trans-qtime            type binary 64.
 02 home-trans-qmax             type binary 16 unsigned.
 02 remote-trans                type binary 32 unsigned.
 02 remote-trans-qtime          type binary 64.
 02 remote-trans-qmax           type binary 16 unsigned.
 02 home-net-trans              type binary 32 unsigned.
 02 home-net-trans-qtime        type binary 64.
 02 home-net-trans-qmax         type binary 16 unsigned.
 02 aborting-trans              type binary 32 unsigned.
 02 trans-backout-qtime         type binary 64.
 02 trans-backout-qmax          type binary 16 unsigned.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for TMF Entities (ZMS Style)

The ZMS style TMF DDL record is supported in Measure G11 and later PVUs. The fields included in BRIEF reports are in **boldface**.

ID Fields DDL Definition

```
DEFINITION zmstmf-id.
 02 reserved-1                  type character 8.
end
```

Counter Fields DDL Definition

```
DEFINITION zmstmf-ctrs.
 02 home-trans                  type binary 64.
 02 home-trans-qtime            type binary 64.
 02 home-net-trans              type binary 64.
 02 home-net-trans-qtime        type binary 64.
 02 remote-trans                type binary 64.
 02 remote-trans-qtime          type binary 64.
 02 aborting-trans              type binary 64.
 02 trans-backout-qtime         type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmstmf. FILE is "zmstmf" ENTRY-SEQUENCED.
 02 hdr                         type zmsheader.
 02 ctr                         type zmstmf-ctrs.
 02 id                          type zmstmf-id.
end
```

HOME-TRANS

Number of transactions that began in the measured CPU.

This counter includes transactions regardless of whether they remain on the local system or are distributed to a remote system. The total number of transactions this system has participated in is the sum of the HOME-TRANS and the REMOTE-TRANS counters for all CPUs in the system.

Counter type: Incrementing.

HOME-TRANS-QTIME

Sum of lifetimes of transactions originated in the measured CPU. This counter includes transactions regardless of whether they remain in the local system or are distributed to a remote system.

Counter type: Queue.

HOME-TRANS-QMAX

- Legacy Style: Maximum number of items on the queue described by the HOME-TRANS-QTIME counter.
- New format: Obsolete.

Counter type: Max queue.

REMOTE-TRANS

Number of transactions that began in a remote system and were distributed to the measured system.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs in response to a PRIMARY command, the REMOTE-TRANS counter in the TMF record of the CPU containing the new primary is advanced. The counter in the TMF record of the CPU containing the old primary does not change.

The total number of transactions this system has participated in is the sum of the HOME-TRANS and REMOTE-TRANS counters for all CPUs in the system.

Counter type: Incrementing.

REMOTE-TRANS-QTIME

The time that transactions that began in a remote system and were distributed to the local system spent on the local system.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs in response to a PRIMARY command, the new $TMP increments the REMOTE-TRANS-QTIME counter in the TMF record of its CPU until it accurately reflects the number of remote network transactions on the system. The REMOTE-TRANS-QTIME counter in the TMF record of the CPU that contained the old primary becomes zero.

Counter type: Queue.

REMOTE-TRANS-QMAX

- Legacy Style: Maximum number of items on the queue described by the REMOTE-TRANS-QTIME counter.
- New format: Obsolete.

Counter type: Max queue.

HOME-NET-TRANS

Number of transactions that began in this system and were subsequently distributed to another system.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs due to a PRIMARY command, the HOME-NET-TRANS counter in the TMF record of the CPU containing the new primary is advanced. The counter in the TMF record of the CPU containing the old primary remains intact.

The total number of network transactions the local system has participated in is the sum of the HOME-NET-TRANS and REMOTE-NET-TRANS counters in all of the CPUs. Number of

transactions that began on the local system and were never distributed to another system is HOME-TRANS minus HOME-NET-TRANS.

Counter type: Incrementing.

HOME-NET-TRANS-QTIME

The time that transactions that began on the local system and were subsequently distributed to a remote system spent on the remote system.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs in response to a PRIMARY command, the new $TMP increments the HOME-NET-TRANS-QTIME counter in the TMF record of its CPU until it accurately reflects the number of HOME NETWORK transactions on the system. The HOME-NET-TRANS-QTIME counter in the TMF record of the CPU that contained the old primary becomes zero.

Counter type: Queue.

HOME-NET-TRANS-QMAX

- Legacy Style: Maximum number of items on the queue described by the HOME-NET-TRANS-QTIME counter.
- New format: Obsolete.

Counter type: Max queue.

ABORTING-TRANS

Number of aborted transactions. An aborted transaction is counted regardless of whether it began in the local system and remained there, began in the local system and was distributed to a remote system, or began in a remote system and was distributed to the local system.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs in response to a PRIMARY command, the ABORTING-TRANS counter in the TMF record of the CPU containing the new primary is advanced. The counter in the TMF record of the CPU containing the old primary remains intact.

A PRIMARY switch of $TMP causes a small number of aborting transactions to be counted twice, once in the old primary and once in the new primary. Multiple PRIMARYs can cause an aborting transaction to be counted more than twice.

Counter type: Incrementing.

TRANS-BACKOUT-QTIME

The time that transactions spent waiting for and being serviced by BACKOUT. After a transaction is designated as aborting and its audit is on disk, that transaction is scheduled for backout services, and this counter is advanced. When backout completes undoing the transaction, this counter is decremented.

This counter is advanced only in the TMF record of the CPU containing the primary $TMP process. If the primary $TMP switches CPUs in response to a PRIMARY command, the new $TMP increments the TRANS-BACKOUT-QTIME counter in the TMF record of its CPU until it accurately reflects the number of transactions in the local system waiting for BACKOUT. The TRANS-BACKOUT-QTIME counter in the TMF record of the CPU that contained the old primary becomes zero.

Counter type: Queue.

TRANS-BACKOUT-QMAX

- Legacy Style: Maximum number of items on the queue described by the TRANS-BACKOUT-QTIME counter.
- New format: Obsolete.

Counter type: Max queue.

Usage Note for All TMF Entities

SUBSYSTEM-VERSION for ZMSTMF records is provided by the TMF subsystem.

For information on TMF, see the TMF set of manuals.

USERDEF

The USERDEF entity type provides information about user-defined counters.

In Measure G09 and later PVUs, the USERDEF entity type handles OSS file pathnames.

In the ZMS style interface (Measure G11 or later), the record template, report, and structured data file name for USERDEF records is ZMSUDEF.

Entity Specification Syntax for USERDEF Entities

To describe a USERDEF entity:

```
USERDEF entity-spec
```

USERDEF

collects information about one or more user-defined counters modified by the USERDEF process.

entity-spec

is specified as:

```
{ *                                                  }
{ cpu,pin                                            }
{ $process-name [ (pid) ]                            }
{ [[$device.]subvolume.]filename[:CRVSN][ (pid) ]    }
{ "pname" [ (pid) ] }
```

*

measures all user-defined processes on the local system.

cpu

is the number of the CPU on which the process to be measured is running. To indicate all CPUs, use an asterisk (*). The default is all CPUs.

pin

is the process identification number of the process to be measured. To indicate all processes, use an asterisk (*). The default is all processes.

$*process-name*

is the name of the process to be measured.

pid

is the process identifier of the process to be measured. The process identifier identifies the owner of the process. Specify *pid* using these two variables:

```
{ cpu,pin }
```

cpu
is the CPU on which the process is running.

pin
is the process identification number of the process.

$*device*
is the name of the volume (device) that contains the process to be measured. To indicate all devices, use an asterisk (*). The default is the current default device.

subvolume
is the name of the subvolume that contains the process to be measured. To indicate all subvolumes, use an asterisk (*). The default is the current default subvolume.

filename
is the name of the object file for the process. *filename* must be the object file of an executing process. To indicate all files (except temporary files), use an asterisk (*).

:*CRVSN*
in Measure G10 and later PVUs, is the timestamp, creation version serial number, or file-name extension necessary to form a unique filename. Use this option to guarantee file-name uniqueness. The CRVSN is available from the Measure report and the LISTGNAME command.

"*pname*"
can be either a fully qualified or partial OSS file pathname. An OSS file pathname that does not begin with a slash (/) is considered to be a partial pathname and is expanded by prefacing it with the current setting for OSSPATH.

NOTE: OSS file pathnames are case-sensitive and must be specified within double quotation marks (" "). Valid OSS file pathnames can refer to specific files or to a set of files within a specific directory. If a directory is specified, only files in that directory are included. Files in directories subordinate to the specified directory are not included.

DDL Record for USERDEF Entities (Legacy Style)

This is the Legacy Style DDL record for USERDEF entities. This record will not change after the G10 Measure PVU.

```
RECORD userdef. FILE is "userdef" ENTRY-SEQUENCED.
                        .
                        .
                        .
(error, time items, and measurement identification items;
see Common Entity Header Fields (page 141))
                        .
                        .
                        .
* entity identification items:
 02 pin                          type binary 16 unsigned.
 02 process-name                 type character 8.
 02 program-file-name.
    03 volume                    type character 8.
    03 subvol                    type character 8.
    03 filename                  type character 8.
 02 priority                     type binary 16 unsigned.
* counter identifiers and value:
 02 name                         type character 16.
 02 type                         type binary 16 unsigned.
 02 index                        type binary 16 unsigned.
 02 max-value                    type binary 16 unsigned.
 02 time-value                   type binary 64.
```

```
 02 counts                      redefines time-value.
   03 FILLER                    type binary 32 unsigned.
   03 count-value               type binary 32 unsigned.
* field for entity identification:
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
* entity identification items for D10:
 02 ancestor-cpu                type binary 16 unsigned.
 02 ancestor-pin                type binary 16 unsigned.
 02 ancestor-sysname            type character 8.
 02 ancestor-process-name       type character 8.
* SMS changes:
 02 device-name                 type character 8.
* Identifiers for OSS file pathname support:
 02 osspid                      type binary 32 unsigned.
 02 program-file-name-mid.
    03 pathid                   type binary 64.
    03 crvsn                    type binary 64.
* New identifiers for NetBatch Job Control:
 02 GMOM.
    03 GMOM-node                type binary 16 unsigned.
    03 GMOM-cpu                 type binary 16 unsigned.
    03 GMOM-pin                 type binary 16 unsigned.
    03 GMOM-jobid               type binary 16 unsigned.
 02 GMOM-full-id                type binary 64
                                    redefines GMOM.
 02 GMOM-sysname                type character 8.
 02 GMOM-process-name           type character 8.
end
```

For descriptions of the header fields used by all entities, see Common Entity Header Fields (page 141).

DDL Record for USERDEF Entities (ZMS Style)

The ZMS style DDL record for USERDEF entities is supported in Measure G11 and later PVUs.

ID Fields DDL Definition

```
DEFINITION zmsudef-id.
 02 pin                          type binary 16 unsigned.
 02 priority                     type binary 16 unsigned.
 02 userid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 creatorid.
    03 group                    type binary 8 unsigned.
    03 user                     type binary 8 unsigned.
 02 process-name                type character 8.
 02 program-file-name.
    03 volume                   type character 8.
    03 subvol                   type character 8.
    03 filename                 type character 8.
 02 osspid                      type binary 32 unsigned.
 02 ancestor-cpu                type binary 16 unsigned.
 02 ancestor-pin                type binary 16 unsigned.
 02 ancestor-sysname            type character 8.
 02 ancestor-process-name       type character 8.
 02 device-name                 type character 8.
 02 program-file-name-MID.
    03 PATHID                   type character 24.
```

```
    03 CRVSN                      type character 6.
 02 reserved-1                    type character 2.
 02 GMOM.
    03 GMOM-node                  type binary 16 unsigned.
    03 GMOM-cpu                   type binary 16 unsigned.
    03 GMOM-pin                   type binary 16 unsigned.
    03 GMOM-jobid                 type binary 16 unsigned.
 02 GMOM-full-id                  type binary 64
                                       redefines GMOM.
 02 GMOM-sysname                  type character 8.
 02 GMOM-process-name             type character 8.
 02 type                          type binary 16 unsigned.
 02 index                         type binary 16 unsigned.
 02 reserved-2                    type character 4.
 02 name                          type character 16.
end
```

Counter Fields DDL Definition

```
DEFINITION zmsudef-ctrs.
 02 value                         type binary 64.
end
```

DDL Record Description Fields

```
RECORD zmsudef. FILE is "zmsudef" ENTRY-SEQUENCED.
 02 hdr                           type zmsheader.
 02 id                            type zmsudef-id.
 02 ctr                           type zmsudef-ctrs.
end
```

PIN

Process identification number of the measured process.

PROCESS-NAME

Name of the measured process.

PROGRAM-FILE-NAME

Name of the object file the process is executing. This field is divided into three subfields: VOLUME, SUBVOL, and FILENAME. The name can apply to either a NonStop file or an SMF file.

For NonStop files, FILE-NAME represents the physical file name that was specified when the process was executed. The VOLUME subfield gives the device name of the physical volume on which the disk file is located.

For SMF files, FILE-NAME represents a location-independent logical file name that was used when the process was executed. The device location of the physical file that corresponds to the logical file name is stored in the DEVICE-NAME field.

PRIORITY

Priority of the measured process.

NAME

Name of the user-defined counter.

TYPE

Type of the user-defined counter:

1	Accumulating counter, 32 bits (ACCUM)
2	Busy counter (BUSY)
3	Queue counter (QUEUE)
4	Accumulating counter, 64 bits (FACCUM)
5	Queue busy counter (QBUSY)
6	Busy counter, implemented with timer cells (TCELLBUSY)
7	Queue counter, implemented with timer cells (TCELLQUEUE)
8	Queue busy counter, implemented with timer cells (TCELLQBUSY)

INDEX

Index value of the counter. A single counter value has an index value of 0. Index values for an array can range from 0 through 127.

MAX-VALUE

- Legacy Style: Maximum number of items on the queue described by the TIME-VALUE counter.
- New format: Obsolete.

TIME-VALUE

- Legacy Style: Redefined. See COUNTS.
- New format: Obsolete; use VALUE.

COUNTS

- Legacy Style: Value of a user-defined accumulating counter.
- New format: Obsolete; use VALUE.

USERID

Process accessor ID (PAID). This field helps to identify the owner of the process.

CREATORID

Creator access ID (CAID). This field identifies the creator of the process and is divided into two subfields: GROUP and USER.

ANCESTOR-CPU

Number of the CPU on which the ancestor process resides.

ANCESTOR-PIN

Process identification number of the ancestor process.

ANCESTOR-SYSNAME

Name of the system on which the ancestor process resides.

ANCESTOR-PROCESS-NAME

Name of the ancestor process.

DEVICE-NAME

Disk device on which the program file is located. For SMF files, this field provides the physical location that corresponds to the logical file name. For NonStop files, this field is the same as the FILE-NAME VOLUME subfield.

OSSPID

If the process runs in the OSS environment, this field contains the OSS Process ID. If it is another process, this field contains zeros.

PROGRAM-FILE-NAME-MID

FILE-NAME-MID has two subfields: PATHID and CRVSN. PATHID is an internal format representation of an OSS file pathname. For other files, this field contains zeros. CRVSN is a creation version serial number that identifies a unique instance of an OSS file. For other files, this field contains zeros.

GMOM-NODE

(ZMS style only) The expand node number of the GMOM process if this process is part of a NETBATCH job and if the GMOM process is remote, otherwise zero.

GMOM-CPU

(ZMS style only) The processor number of the GMOM process if this process is part of a NETBATCH job; otherwise zero.

GMOM-PIN

(ZMS style only) The PIN of the GMOM process if this process is part of a NETBATCH job; otherwise zero.

GMOM-JOBID

(ZMS style only) The jobid of the NETBATCH job that initiated this process, assigned by the indicated GMOM; otherwise zero.

GMOM-FULL-ID

(ZMS style only) A 64-bit redefinition of the individual GMOM fields as a single value. Zero indicates the process is not part of a NETBATCH job stream.

GMOM-SYSNAME

(ZMS style only) The Expand system name of the GMOM node if the GMOM process is remote; otherwise spaces.

GMOM-PROCESS-NAME

(ZMS style only) The name of the GMOM process that initiated the NETBATCH job that in turn initiated this process; otherwise spaces.

Usage Notes for All USERDEF Entities

- To define a counter in an application, modify the source code to call the MEASCOUNTERBUMPINIT and MEASCOUNTERBUMP procedures at appropriate times. For descriptions, see MEASCOUNTERBUMP (page 396) and MEASCOUNTERBUMPINIT (page 397). For full information on instrumenting user-defined applications, see the *Measure User's Guide*.
- To collect information from a user-defined counter, use the ADD USERDEF command to specify the process that contains the user-defined counter and the ADD COUNTER command

to specify the name of the counter. For descriptions, see ADD USERDEF and ADD COUNTER under ADD *entity-type* (page 43).

Usage Notes for G-Series USERDEF Entities

- For G07 and later Measure PVUs, the USERDEF entity has a 64-bit accumulator counter type. To add this new counter type, use the MEASCOM ADD COUNTER command with the keyword FACCUM.
- For G11 and later Measure PVUs:
  - Customer applications can provide SUBSYSTEM-VERSION for ZMSUSER records if the applications use the SUBSYSTEM-VERSION extension to the MEASCOUNTERBUMPINIT() procedure.
  - The USERDEF report distinguishes between Accum counters (32-Bit) and Faccum counters (64-bit). In the ZMSUSER report, all counters are formatted from 64-bit fields and reported as type Accum.

Usage Notes for H-Series and J-Series USERDEF Entities

- Several new counter types are defined for H-series and J-series RVUs.
  - Queue Busy. This counter type measures the amount of time during which the queue for a resource is busy: that is the amount of time during which items were waiting on the queue.
  - Busy, implemented with timer cells. This counter has the same semantics as a conventional busy counter but uses the timer-cell mechanism to provide finer granularity timers.
  - Queue, implemented with timer cells. This counter has the same semantics as a conventional queue counter but uses the timer-cell mechanism to provide finer granularity timers.
  - Queue busy, implemented with timer cells. This counter has the same semantics as a conventional queue busy counter but uses the timer-cell mechanism to provide finer granularity timers.
- If you define counters of the new types, Measure implicitly uses timer cells to maintain the counters. For more information about using the timer cell mechanism, see Chapter 4: Measure Callable Procedures (page 349).

4 Measure Callable Procedures

This chapter describes the Measure callable procedures, which let an application process control the Measure performance monitor and access measurement data.

For general information on Measure procedures, see Measure Procedures Overview (page 351).

Summary of Measure Procedures

Table 4-1 lists the Measure procedures by function. For examples of programs that use the Measure procedures, see the *Measure User's Guide*.

Table 4-1 Measure Callable Procedures

Function	Procedure	Description	Page
Collecting data	MEASOPEN	Accesses a measurement data file	421
	MEASCONFIGURE	Defines a measurement configuration	358
	MEASCONTROL	Starts and stops a measurement	395
	MEASCLOSE	Deletes access to a measurement data file	358
Examining data	MEASGETVERSION	Returns either the Measure product version or the external entity record lengths for a data file or both	398
	MEASINFO	Returns configuration information from measurement data file that is not open	403
	MEASSTATUS	Returns information about an active measurement	446
	MEASREAD	Reads data from a measurement data file; returns result to caller	424
	MEASREAD_DIFF_	Reads data from a measurement data file; returns result to caller	426
	MEASWRITE_DIFF_	Reads data from a measurement data file; puts result in a file	448
	MEASREADACTIVE	Reads data from active counter records	430
	MEAS_READACTIVE_	Reads data from active counter records	433
	MEAS_READACTIVE_MANY_	Reads data from multiple active counter records in the same call	434
	MEASREADCONF	Reads configuration information from a measurement data file	437

Table 4-1 Measure Callable Procedures *(continued)*

Function	Procedure	Description	Page
Bumping user-defined counters	MEASCOUNTERBUMPINIT	Returns the offset of a user-defined counter	397
	MEASCOUNTERBUMP	Bumps a user-defined counter	396
Controlling the Measure subsystem	MEASMONCONTROL	Starts and stops the subsystem	418
	MEASMONSTATUS	Returns the name and number of currently active measurement	419
Getting system configuration information	MEASLISTCONFIG	Returns system configuration information supplied by the MEASCTL process	407
Translating file names	MEASLISTENAME	Translates Guardian file names or MIDs to corresponding external format ANSI SQL names or OSS pathnames	408
	MEASLISTEXTNAMES	Writes Measure list structured OSS and ANSI SQL name information to the EXTNAMES file	411
	MEASLISTGNAME	Translates OSS file pathnames to Guardian file names, returns the MID (PATHID and CRVSN) content for entity descriptor construction, and returns the contents of an OSS directory	412
	MEAS_GETDESCINFO_	Translates an ANSI SQL name to its corresponding entity descriptor components	400
	MEASLISTOSSNAMES	Returns structured OSS file pathname translation information to the file OSSNAMES	416
	MEAS_CODERANGENAME_DEMANGLE_	Translates the internal (mangled) representation of a procedure name back into the original name specified by the programmer.	358
	MEASLISTPNAME	Translates Guardian file name or OSS pathid to its OSS file pathname equivalent	417
Using ANSI SQL names	MEAS_SQL_MAP_INIT_	Initiates the SQL/MX mapping session.	441
	MEAS_SQL_MAP_STOP_	Stops the SQL/MX mapping session.	442
	MEAS_SQLNAME_COMPARE_	Compares two fully qualified ANSI SQL names in external format.	442

Table 4-1 Measure Callable Procedures *(continued)*

Function	Procedure	Description	Page
	MEAS_SQLNAME_RESOLVE_	Combines ANSI SQL name parts to create a fully qualified name in normalized external format.	443
	MEAS_SQLNAME_SCAN_	Parses a fully qualified, possibly wildcarded, ANSI SQL name in external format.	445
Adjusting record formats	MEAS_ADJUSTZMSRECORD_	Adjusts ZMS style structure records to the MEASDDLS format with which an application was compiled	356
Maintaining fine granularity timers	MEAS_ALLOCATE_TIMERCELLS_	Allocates timer cells for maintaining TCELLBUSY, TCELLQUEUE, and TCELLQBUSY timers	356
	MEAS_DEALLOCATE_TIMERCELLS_	Deallocates timer cells	398
	MEAS_BUMP_TIMERCELL_	Sets, resets, increments, or decrements a TCELLBUSY, TCELLQUEUE, or TCELLQBUSY timer	357
	MEAS_RETRIEVE_TIMERCELLS_	Retrieves timer values for one or more TCELLBUSY, TCELLQUEUE, and TCELLQBUSY timers	440

Measure Procedures Overview

Reading in Measure Records (DDL)

Measure DDL records change from PVU to PVU. New counters are always added to the end of the existing Measure DDL records. This ensures that applications using existing counters are not affected by the addition of new counters.

Some simple coding practices can help you avoid compatibility problems with future PVUs. User applications that request records through the programmatic interface should use a buffer large enough to accommodate the new longer records (even if the new fields are of no interest). Because Measure does not return partial records, applications written to use buffers of several thousand bytes avoid compatibility problems in reading the DDL records.

Legacy and ZMS Style Records

In H01 and later PVUs, the internal representation of Measure data is in ZMS style records, and any further enhancements will occur only in ZMS style. Yet the default style of records returned by Measure APIs is still legacy.

To take advantage of future enhancements and improve application performance, it is advisable to port existing applications to use ZMS style. To achieve this effect, use the `templateversion` parameter (present in various procedure calls).

If your application continues to request legacy style records, Measure converts the data from its internal representation to legacy style. This conversion affects legacy counter fields in several ways:

- If the legacy record has two versions of the same counter--one 32-bit version and one 64-bit version (typically ending in -F)--the 64-bit field will invariably contain a value. The 32-bit field will contain a value only if the value in the internal ZMS style record fits into 32 bits.

If the internal value does not fit into 32 bits, the 32-bit counter returned to the application has a value of -1.

- MAX-QLEN counters are no longer maintained. Those fields are returned in legacy style records with a value of 1.
- Fields that have been added to ZMS style records since the Measure G12 PVU are not available to applications requesting legacy style records.

Reading in the Declaration Files

Use the TAL compiler ?SOURCE command to read these declaration files into the source code global declarations:

- $SYSTEM.SYSTEM.EXTDECS0 contains the external declarations for the Measure procedures. Each Measure procedure used in your program must be specified in the ?SOURCE command.
- $SYSTEM.SYS*nn*.MEASDECS contains the template structures for the Measure control block, configuration table, and entity descriptors. It also declares literal identifiers for the error message codes returned by the procedures and for constants used in the configuration table and entity descriptors.

Allocating Space for the Measure Control Block

Unless you use only the MEASCLOSE, MEASOPEN, MEASREAD, and MEASREADCONF procedures in a program, you must allocate space in the program's global data area for the Measure control block. You need only one Measure control block per process, regardless of the number of measurements the process makes.

MEASCB^DEF in the MEASDECS file is the template structure for the control block. This referral structure allocates space for the control block:

```
STRUCT .MEASCB(MEASCB^DEF)
```

The Measure subsystem uses the control block to store data for procedure calls. Do not modify the contents of the block after you pass it to a procedure.

Before you call the first procedure that uses the control block, you must initialize each element in the block to -1:

```
$FILL16(MEASCB, $LEN(MEASCB)/2, -1);
```

Specifying Entity Descriptors

Like the entity specifications used in the command interface, entity descriptors describe an entity or set of entities. Use entity descriptors to define the entities to measure and to specify the counter records to maintain for each entity. Entity descriptors are part of the configuration table that you pass to the MEASCONFIGURE procedure before starting a measurement.

The MEASDECS file declares template structures for the entity descriptors. For example, the CPU^DESC template contains the fields you use to define a CPU entity. Each entity descriptor contains:

- *Type*. A numeric identifier or a literal that identifies the type of entity being defined. The MEASDECS file declares the literal for each entity type. See Table 4-2: Entity Descriptors and Type Values (page 360)
- *Len*. The length, in bytes, of the descriptor.
- *Cpu^number*. The number of the CPU where the entity resides. (A few descriptors use a different name for this field.)

The remaining fields in a descriptor identify the entity or entities to be measured.

For detailed descriptions of entity descriptors, see MEASCONFIGURE (page 358).

Measuring a Set of Entities

In many descriptor fields, you can use wild-card values to indicate all entities of a particular type. For example:

- In a CPU entity descriptor, the literal -1 in the *cpu^number* field specifies all CPUs on the system.
- In a field representing a PIN, channel, ctl, unit, or system number, the literal -1 specifies all.
- In name fields, an asterisk (*) specifies all. Such fields must be left justified and blank filled. Include, as the left-most character, any appropriate leading character, such as a backslash (\) for a node, a dollar sign ($) for a volume, or a pound sign (#) for a subdevice.
- To designate all system processes, set the PIN to -2.

Wildcards are not always valid. For example, when Measure reads from active counters, the entity descriptor you pass to MEASREADACTIVE must specify a single entity.

Excluding Entities from a Set

After specifying a set of entities to measure, you can exclude individual entities from the measurement. To exclude an entity, add a descriptor for that entity to the configuration table, and set the *type* field to a negative value. You can specify a negative value for the *type* string literal or the numeric identifier. For example, to measure all CPUs except CPU 6:

1. Add a CPU descriptor in which *type* is set to CPU^T and *cpu^number* is set to ALL.
2. Add a second CPU descriptor in which *type* is set to -CPU^T (the negative literal that corresponds to CPU^T) and *cpu^number* is set to 6.

To use numeric type identifiers instead of string literals:

1. Add a descriptor in which *type* is set to 1 (the numeric identifier for CPU) and *cpu^number* is ALL (or the equivalent numeric literal, -1).
2. Add a second descriptor in which bit 0 of the *type* field is set to 1. This sets the exclude flag and is equivalent to setting a negative value.
3. Set bits 1 through 15 of the *type* field to 1, the numeric type identifier for CPU.
4. Set the *cpu^number* field to 6.

Specifying File and Device Names

All disk file, system, device, and process names must be in local internal name format. For descriptions of internal file name formats, see the *Guardian Procedure Calls Reference Manual*. If you use an asterisk in a name field or subfield, you must include the appropriate leading character (\, $, or #) and pad the field with blanks.

Specifying ANSI SQL Names

With Measure G11 and H01 PVU support for ANSI SQL names, additional fields have been added to some of the entity descriptors.

When an ANSI SQL name is passed as an argument to a procedure, it should not be enclosed in single quotes.

For more information, see Handling of ANSI SQL Names (page 146).

Creating the Configuration Table

The configuration table (*contab*) defines which entities are in a measurement. You pass the configuration table to MEASCONFIGURE before you start a measurement. Once a measurement

is configured, you can read its configuration table by calling MEASREADCONF or MEASINFO. The configuration table consists of:

- A header record
- The entity descriptor sections
- A trailer record

Both the header and trailer records have fixed lengths. The entity descriptor sections can vary in length, so the table as a whole is variable length.

Header Record

Use the template structure CONTAB^HDR to define the header record, which contains:

Variable	Description
Type	The numeric value 50 or the literal CONTAB^T (declared in the MEASDECS file). This field identifies this record as the header record.
Len	The length of the entire configuration table, in bytes.
Sections	An array of offsets in bytes that point to each entity type's descriptor section within the table. The first word of the array is always zero. The following offsets are ordered by entity type number and indexed to the beginning of the configuration table. (For a description of entity type numbers, see MEASCONFIGURE (page 358).) If an entity type is not being measured, its offset is zero.
Maxents	The header format. *Maxents* is the first element in the sections array (that is, SECTIONS[0]).

NOTE: If you have older custom programs that do not use this header format, recompile those programs.

Entity Descriptor Sections

The body of the configuration table is an array of entity descriptors. The entity descriptors are grouped into sections according to entity type, with the sections ordered by their numeric identifiers. All CPU descriptors (type number 1) appear first, followed by all PROCESS descriptors (type number 2), then all PROCESSH descriptors (type number 3), and so on.

If an entity type is not being measured, do not include a section for that type.

DELETE descriptors take precedence over ADD descriptors. If a match is found in the *contab* on an ADD descriptor, the return value is tentatively set to true. However, if a match is then found on a DELETE descriptor, a return value of false is immediately returned.

Trailer Record

Use the template structure CONTAB^TRAILER to define the trailer record. It contains:

Variable	Description
Type	The numeric value 51 or the string literal CONTAB^TRAILER^T (declared in the MEASDECS file). This field identifies this record as the trailer record.
Len	The length of the trailer record, in bytes.

Use the template structure CONTAB^TRAILER to define the trailer record and to assign the literal CONTAB^TRAILER^T to the *type* field.

Interpreting Error Codes in Measure Procedures

The Measure procedures are integer function procedures that return an error word. If no error occurs, the procedures return 0 (zero) in the error word. If an error occurs, the procedures return an error code.

Errors unique to the Measure procedures have error codes between 3200 and 3499 and are described in Appendix B: Error Codes (page 471). The MEASDECS file declares literal identifiers for the error codes.

The Measure procedures use space on your program data stack to execute. (The maximum space used by a Measure procedure is approximately three pages.) If you receive an invalid address reference trap (trap 0) or a stack overflow trap (trap 3) while a Measure procedure is executing, try allocating more data pages with the TAL compiler ?DATAPAGES command or the RUN command MEM parameter.

Maintaining Compatibility With New Structures in MEASDDLS and MEASCHMA

In some instances, a recompile is required for C and C++ applications. For example, G08 Measure redefines the MODULE and SUBDEVICE fields of the SERVERNET entity descriptor as the ServerNet Cluster NODE number and CPU number of the remote system's processor. For C or C++ applications that use the Measure callable procedures to configure ServerNet measurements, to use the new structures provided by MEASDDLS and MEASCHMA, a recompile is required. These redefined fields must be accessed as members of a union.

Using Timer Cells

H-series Measure introduces a timer-cell facility, which improves the accuracy of timer calculations for events that complete within the context of a dispatch or a simple message exchange between two processes in the same processor. This mechanism lets you keep fine-granularity timers without incurring the cost of loose processor synchronization on all timer updates. Timer cell values are synchronized only when copied at intervals or reported to the requesting application.

Counter types associated with the timer cell mechanism are:

- TCELLBUSY. This counter type is equivalent to a conventional BUSY counter but is implemented with timer cells.
- TCELLQUEUE. This counter type is equivalent to a conventional QUEUE counter but is implemented with timer cells.
- TCELLQBUSY. This counter type is equivalent to a conventional QBUSY counter but is implemented with timer cells.

For USERDEF counters, you can take advantage of the timer-cell mechanism by defining counters of the new types. For example, if an application ran on an G-series RVU and you now plan to run it on an H-series and J-series RVU, consider redefining Busy, Queue, and Queue Busy timers if the events they measure occur at fine intervals. Events that involve interprocessor communication or I/O need not be redefined, because in such cases, loose processor synchronization has little or no impact.

Redefining a USERDEF counter to use timer cells does not necessitate application changes. Although new callable procedures support the timer-cell mechanism, Measure calls those procedures implicitly to maintain timer-cell counters that you define for USERDEF entities. The only change required is in the configuration of the USERDEF entity, where you specify that a counter is of one of the new types.

Use the timer-cell calls explicitly only if your application maintains counters outside of Measure. For more information, refer to the descriptions of MEAS_ALLOCATE_TIMERCELLS_, MEAS_DEALLOCATE_TIMERCELLS_ , MEAS_BUMP_TIMERCELL_, and MEAS_RETRIEVE_TIMERCELLS_.

MEAS_ADJUSTZMSRECORD_

Takes ZMS style structure records in any published format and adjusts them to the MEASDDLS format with which an application was compiled.

```
error := MEAS_ADJUSTZMSRECORD_ (  in^record           ! i
                                , out^record          ! o
                                , templateversion );  ! i
```

in^record

INT:EXT:ref:*n*

is a buffer containing a ZMS style external record to be padded or truncated to a new desired format. The length of *in^record* is derived from the information encoded in the template version of the header record pointed to by *in^record*.

out^record

INT:EXT:ref:*n*

is a buffer in which to return a ZMS style external record padded or truncated to a format specified by the *templateversion* parameter. The length of *out^record* is derived from the information encoded in the *templateversion* parameter.

templateversion

FIXED:value

is the record template version for the format of record being requested. It is obtained from the MEASDDLS with which the requesting application was compiled.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

Usage Notes

If the adjustment is to a smaller or earlier record format, the fields added or increased between the two record versions are stripped or truncated.

If the adjustment is to a larger or newer record format, the location of the new or larger fields in the more recent format are padded.

All padding is with the value zero, except for long variable-length identifiers at the end of the identifiers section. If these increase in size, they are padded with spaces.

To have an application read a structured data file (file code 170) containing records of a template version other than that with which the application was compiled:

1. Open the file structured data file.
2. Use FILE_GETINFO_ to obtain the record length of the data in the file.
3. Read each record of the file to a buffer long enough to hold the complete record.
4. Declare a buffer area for a reformatted record based on the MEASDDLS template.
5. Invoke MEAS_ADJUSTZMSRECORD_ passing the record from the file, the declared buffer area for the reformatted record, and the correct zms *entity*-template-version literal from MEASDDLS.

MEAS_ALLOCATE_TIMERCELLS_

Allocates the specified number of timer cells and returns an array of indexes to those timers. Use this procedure if your application requires fine granularity timers to maintain counters outside of Measure. Measure calls this procedure implicitly when you define a timer-cell counter for the USERDEF entity.

```
error := MEAS_ALLOCATE_TIMERCELLS_ ( count       ! i
                                   , types       ! i
                                   , indexes);   ! i,o
```

count

INT:VALUE

is the number of timer cells to allocate. Allocate one timer cell for each counter to be maintained. The maximum number of timer cells per process is 10,000.

types

INT:EXT:REF:*count*

is an array of counter types. The array size is given by *count*. The permissible counter types are TCELLBUSY (6), TCELLQUEUE (7), and TCELLQBUSY (8). This array maps, item by item, to the array of indexes.

indexes

INT:EXT:REF:*count*

is an array of indexes to the timer cells. The array size is given by *count*. On input, the array is empty. On output, it contains the indexes to use in subsequent calls. This array maps, item by item, to the array of types.

MEAS_BUMP_TIMERCELL_

Modifies the value of a counter by increasing, decreasing, setting, or resetting the timer cell value. Use this procedure if your application requires fine granularity timers to maintain counters outside of Measure. Measure calls this procedure implicitly when you define a timer-cell counter for the USERDEF entity.

```
error := MEAS_BUMP_TIMERCELL_ ( index       ! i
                              , op          ! i
                              ,[ count]);   ! i
```

index

INT:VALUE

is the index of the timer cell.

op

INT:VALUE

indicates the operation to be performed on the timer cell. The possible values are:

- SETBUSY (3) to set a TCELLBUSY counter. Use this value when the resource measured by the counter begins to be used.
- RESETBUSY (4) to reset a TCELLBUSY counter. Use this value when the resource measured by the counter stops being used.
- INCQUEUE (5) to increase a TCELLQUEUE counter. Use this value when an item is placed in the queue.
- DECQUEUE (6) to decrease a TCELLQUEUE counter. Use this value when an item is removed from the queue.
- INCQBUSY (7) to increase a TCELLQBUSY counter. Use this value when the first item is placed in a queue after the queue has been empty.
- DECQBUSY (8) to decrease a TCELLQBUSY counter. Use this value when the last item is removed from the queue, rendering the queue empty.

count

INT:VALUE

is an optional parameter, used to increase or decrease the same counter multiple times in the same call. For SETBUSY and RESETBUSY, the only valid value is 1.

MEASCLOSE

Closes the measurement data file and stops the file-handling process (MEASFH) that the Measure subsystem created to access the data file. To open a measurement data file, use the MEASOPEN procedure.

```
error := MEASCLOSE ( dfnum ); ! i
```

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

MEAS_CODERANGENAME_DEMANGLE_

Returns the original procedure name that was provided by the programmer.

```
error := MEAS_CODERANGENAME_DEMANGLE_ ( InputName, ! i
                          ,InputNameLen,                  ! i
                          ,OutputName                     ! o
                          ,OutputNameMaxLen               ! i
                          ,OutputNameLen                  ! o
                          ,NameType );                    ! o
```

InputName

STRING.EXT:ref:*

is a buffer of up to 1024 bytes containing a name string to be converted. No terminating null is expected.

InputNameLen

INT:value

is the count of bytes in the input name string.

OutputName

STRING.EXT:ref:*

is a buffer of up to 1024 bytes containing the converted name. If the input name is not a mangled C++ name, the input string is copied to the output buffer without change.

OutputNameMaxLen

INT:value

is the size of the input buffer.

OutputNameLen

INT.EXT:ref:*

is the count of bytes in the output buffer.

NameType

INT.EXT:ref:*

is 1 for a non-C++ name; 2 for a C++ name.

MEASCONFIGURE

Configures a measurement.

Your measurement application must call the MEASOPEN procedure (to initialize a Measure data file) before calling MEASCONFIGURE.

In Measure H01 and later PVUs, MEASCONFIGURE accepts a *contab* which contains DISCOPEN, DISKFILE, FILE, or SQLSTMT entity descriptors for ANSI SQL objects or partitions.

```
error := MEASCONFIGURE ( meascb          ! i,o
                         ,dfnum          ! i
```

```
                        ,measnum        ! o
                        ,contab         ! i
                        ,[errDetail]) ; ! o
```

meascb

input, output

INT:ref:$LEN (MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. Once you pass *meascb* to a Measure procedure, do not modify its contents.

The MEASDECS file contains the structure definition for the control block descriptor (MEASCB^DEF).

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

measnum

output

INT:ref:1

is the measurement number. Use this number in subsequent procedure calls to identify the measurement.

contab

input

INT.EXT:ref:*

is an array that defines the measurement configuration. The *contab* array contains a header section, a trailer section, and a section for each entity type.

Structure definitions for use in the *contab* array are defined in the MEASDECS file. MEASDECS declares both a literal and a numeric identifier for each entity type and for the header and trailer records.

For ANSI SQL objects or partitions, the entity descriptors can be DISCOPEN, DISKFILE, FILE, or SQLSTMT entity descriptors.

Table 4-2 lists each section of the configuration table, the name of the associated structure in the MEASDECS file, the literal, and the numeric identifier. The sections must appear in the order shown.

errDetail

output

FIXED.EXT:ref:1

The *errDetail* parameter may only be used with Measure H03-AFV and above, and J01-AFW and above. This parameter is only valid if the error ERR^BADDESC is returned. It contains information about the first encountered error in the *contab* buffer.

errDetail.32:47	> 0 =	The ordinal number of the bad descriptor within an entity type, starting with 1.
errDetail.48:63	-1 =	This argument is not supported by the used version of MEASFH
	0 =	There is something wrong with the contab, but not necessarily with any of the descriptors.
	> 0 =	The entity type.

Table 4-2 Entity Descriptors and Type Values

Section	Descriptor Template Structure (Systems Running D-Series RVUs)	Descriptor Template Structure (Systems Running G-Series and later RVUs)	Type Literal	Numeric Identifier
CONTAB header	CONTAB^HDR	CONTAB^HDR	CONTAB^T	50
CPU	CPU^DESC	CPU^DESC	CPU^T	1
PROCESS	PROCESS^DESC	PROCESS^DESC PROCESS^OSS^DESC	PROCESS^T	2
PROCESSH	PROCESSH^DESC	PROCESSH^DESC CODE^SPACE^DESC PROCESSH^OSS^DESC CODE^SPACE^OSS^DESC	PROCESSH^T	3
USERDEF	USERDEF^DESC COUNTER^DESC	USERDEF^DESC COUNTER^DESC USERDEF^OSS^DESC	USERDEF^T	4
FILE	FILE^OPEN^DESC	FILE^OPEN^DESC FILE^OPEN^DESC^D10 FILE^OPEN^OSS^DESC FILE^OPEN^OSS^DESC^D10 FILE^OPEN^ANSI^DESC	FILOP^T	5
DISCOPEN	FILE^OPEN^DESC	FILE^OPEN^DESC FILE^OPEN^OSS^DESC FILE^OPEN^ANSI^DESC	DFILOP^T	6
DISC	DEVICE^DESC	DEVICE^CLIM^DESC DEVICE^SVNET^DESC DEVICE^SVNET^DESC^G05	DISC^T	7
DEVICE	DEVICE^DESC	DEVICE^CLIM^DESC DEVICE^SVNET^DESC DEVICE^SVNET^DESC^G05	IODEV^T	8
LINE	DEVICE^DESC	WAN^DESC	LINE^T	9
NETLINE	DEVICE^DESC	WAN^DESC	NETLINE^T	10
SYSTEM	SYSTEM^DESC	SYSTEM^DESC	REMSYS^T	11
CLUSTER	SYSTEM^DESC	Not applicable	CLUSTER^T	12
TERMINAL	DEVICE^DESC	WAN^DESC	TERM^T	13
TMF	CPU^DESC	CPU^DESC	TMF^T	14
SQLPROC	SQLPROC^DESC	SQLPROC^DESC SQLPROC^OSS^DESC	SQLPROC^T	15
SQLSTMT	SQLSTMT^DESC	SQLSTMT^DESC SQLSTMT^OSS^DESC SQLSTMT^ANSI^DESC	SQLSTMT^T	16
OPDISK	OPDISK^DESC	Not applicable	OPDISK^T	17
CONTROLLER	CTRL^DESC	Not applicable	CTRL^T	18
SERVERNET	Not applicable	SVNET^DESC	SVNET^T	18
DISKFILE	DISKFILE^DESC	DISKFILE^DESC DISKFILE^OSS^DESC DISKFILE^ANSI^DESC	DISKFILE^T	19

Table 4-2 Entity Descriptors and Type Values *(continued)*

Section	Descriptor Template Structure (Systems Running D-Series RVUs)	Descriptor Template Structure (Systems Running G-Series and later RVUs)	Type Literal	Numeric Identifier
OSSCPU	Not applicable	CPU^DESC	OSSCPU^T	20
OSSNS	Not applicable	OSSNS^DESC	OSSNS^T	21
Maximum value			MAX^T	24
CONTAB trailer	CONTAB^TRAILER	CONTAB^TRAILER	CONTAB^TRAILER^T	51

NOTE: You can use measurement applications containing D-series descriptors to measure G-series entities if the descriptors specify all entities. For example, if an application uses the DEVICE^DESC descriptor and contains -1 or an asterisk in all identifier fields, you can use that application to measure G-series DEVICE entities. You need not modify the application to use the DEVICE^SVNET^DESC or DEVICE^SVNET^DESC^G05 descriptors.

CONTAB^HDR

describes the contents of the *contab* array. The CONTAB^HDR descriptor consists of these fields, in order:

type

INT:value:1

identifies the header section of the *contab* array. Specify *type* as the literal CONTAB^T or the numeric identifier 50.

len

INT:value:1

is the length in bytes of the entire *contab* array. The maximum length is 32,000 bytes, including the header and the trailer.

sections

INT:value:MAX^T+1

is an array of byte offsets to each section of the *contab* array. The array elements are in entity-type order, as shown in Table 4-2: Entity Descriptors and Type Values (page 360).

Specify the first byte offset (array element 0) as 0. If you omit a section of the *contab* array, specify the byte offset for that section as 0.

maxents

INT:value:1

identifies the header format. This field is equated to SECTIONS[0], so it does not add to the length of CONTAB^HDR.

CTRL^DESC

is the descriptor for CONTROLLER entities on systems running D-series RVUs.

The CTRL^DESC descriptor has the format:

type

INT:value:1

is the literal value CTRL^T or the numeric identifier 18.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor.

cpu^number

INT:value:1

is the number of the CPU on which the controller is configured. To indicate all CPUs, use the literal -1.

channel

INT:value:1

is the channel number of the controller to be measured.

ctrl

INT:value:1

is the controller number (0 through 31). To indicate all controllers, use the literal -1.

ctrl-type

INT:value:1

is the controller type (HP product number, such as 3129).

CPU^DESC

is the descriptor for CPU or TMF entities on systems running D-series or G-series RVUs. The CPU^DESC descriptor consists of these fields, in order:

type

INT:value:1

Entity Type	Literal	Numeric Identifier
CPU	CPU^T	1
TMF	TMF^T	14
OSSCPU	OSSCPU^T	20

is one of:

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor.

cpu^number

INT:value:1

is the number of the CPU to be measured. Use the literal -1 to indicate all CPUs.

DEVICE^DESC

is the descriptor for DEVICE, DISC, LINE, NETLINE, or TERMINAL entities on systems running D-series RVUs.

The DEVICE^DESC descriptor consists of these fields, in order:

type

INT:value:1

is one of:

Entity Type	Literal	Numeric Identifier
DISC	DISC^T	7
DEVICE	IODEV^T	8
LINE	LINE^T	9
NETLINE	NETLINE^T	10
TERMINAL	TERM^T	13

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is 36 (the length in bytes of this descriptor).

cpu^number

INT:value:1

is the number of the CPU on which the device to be measured is running. Use the literal -1 to indicate all CPUs.

ctl

INT:value:1

is the controller number of the device to be measured (0 through 31). Use the literal -1 to indicate all controllers.

unit

INT:value:1

is the unit number of the device to be measured. Use the literal -1 to indicate all units.

device^name

INT:value:12

is the name of the device to be measured. The name must be left-justified and blank-filled and must start with a dollar sign ($). Use a dollar sign and an asterisk ($*) to indicate all devices.

channel-num

INT:value:1

is the channel number of the device to be measured.

DEVICE^CLIM^DESC

is the descriptor for DEVICE or DISC entities connected to a system running a H- or J-series RVU by a CLIM. DEVICE^CLIM^DESC includes a PLPT field, which contains the *lun*, *path* and *target-id* fields that uniquely identify devices connected by a CLIM. You can use DEVICE^SVNET^DESC or DEVICE^SVNET^DESC^G05 to describe other NonStop S-series devices.

type

INT:value:1

is one of:

Entity Type	Literal	Numeric Identifier
DISC	DISC^T	7
DEVICE	IODEV^T	8

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. *len* is 52 if the descriptor does not include the configuration name or 116 if the descriptor includes the configuration name.

cpu^number

INT:value:1

is the number of the CPU on which the device is configured. Use the literal -1 to indicate all CPUs.

servernet

INT:value:1

is the ServerNet fabric. This field identifies a specific path or set of paths to a supported physical device. For G-series and later RVUs, use 0 to indicate the X fabric, 1 to indicate the Y fabric, or the literal -1 to indicate both fabrics.

For devices connected by a ServerNet/DA, use the literal -1.

For devices connected by an FCSA or CLIM, use 0 to indicate the X fabric. By convention, dual fabric controllers are specified by X.

device^name

INT:value:12

is the name of the device to be measured. The name must be left-justified and blank-filled and must start with a dollar sign ($). Use a dollar sign and an asterisk ($*) to indicate all devices.

plpt

starting with the Measure H03 and J01 PVUs, this structure contains *lun*, *path* and *target-id* information about the device to be measured.

plpt^flags

UNSIGNED(8):value:1

is an array of bit flags that describe how this structure should be interpreted.

Bit 0 (%H80) MEAS_CLIM_REL: This is always 1 in Measure H03 and later data files.

Bit 1 (%H40) MEAS_PATH_SEL: This is only used in the entity descriptor when selecting a device for measurement, or in a MEASREADACTIVE, MEAS_READACTIVE_ or MEAS_READACTIVE_MANY_ procedure call. When it is 1, it specifies device selection by *device^name* and *path* only, and not by *GMS*, *lun*, *target^id* or *config^name*.

Bit 2 (%H20) MEAS_CLIM_DEVICE: Indicates that this descriptor is for a CLIM device.

path

UNSIGNED(8):value:1

is the path of the device to be measured. Use the literal -1 to indicate all paths. The following are literals that can be used here: MEAS_PATH_PRIMARY, MEAS_PATH_BACKUP, MEAS_PATH_MIRROR, and MEAS_PATH_MIRROR_BACKUP.

lun

INT:value:1

is the logical unit number of the CLIM device to be measured. Not used for non-CLIM devices. Use the literal -1 to indicate all *lun*s.

reserved

INT:value:1

reserved for future use.

target^id

INT:value:1

is the SCSI port identifier of the non-CLIM device to be measured. Not used by CLIM devices. Use the literal -1 to indicate all SCSI ports.

config^name

INT:value:32

is the configuration name of the device, as defined by the system configuration database.

config^name can also be the configuration name of either a single ServerNet addressable controller (SAC) or an adapter that contains several ServerNet addressable controllers. In this case, the specification includes all devices of the requested type that are supported by that SAC or adapter.

config^name must be null-terminated and null-filled. An asterisk in the first byte indicates all configuration names (this can be set with `config^name^s := "*"`).

To save space in the configuration table, omit CONFIG^NAME from the DEVICE^CLIM^DESC structure. You must set the descriptor length (LEN) to reflect whether you are using CONFIG^NAME because Measure uses the descriptor length to determine the start of the next descriptor in the configuration table. To not use CONFIG^NAME, set LEN to 52 and move only the first 52 bytes of the descriptor to the configuration table. To include CONFIG^NAME, set LEN to 116 and move the entire descriptor to the configuration table.

DEVICE^SVNET^DESC

is the descriptor for DEVICE or DISC entities on systems running G-series RVUs.

To measure devices connected by a ServerNet/DA or FCSA, use the DEVICE^SVNET^DESC^G05 descriptor. This descriptor includes a field to identify the SCSI port of the device being measured. You can also use the DEVICE^SVNET^DESC descriptor to measure such devices if the descriptor specifies all devices or specifies only a device name and CPU number.

NOTE: LINE, NETLINE, and TERMINAL entities, which use the DEVICE^DESC descriptor in D-series RVUs, use the WAN^DESC descriptor in G-series RVUs.

type

INT:value:1

is one of:

Entity Type	Literal	Numeric Identifier
DISC	DISC^T	7
DEVICE	IODEV^T	8

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. *len* is 44 if the descriptor does not include the configuration name or 108 if the descriptor includes the configuration name.

cpu^number

INT:value:1

is the number of the CPU on which the device is configured. Use the literal -1 to indicate all CPUs.

servernet

INT:value:1

is the ServerNet fabric. This field identifies a specific path or set of paths to a supported physical device. For G-series and later RVUs, use 0 to indicate the X fabric, 1 to indicate the Y fabric, or the literal -1 to indicate both fabrics.

GMS

is the physical location address (group, module, slot). The GMS structure consists of these fields:

group

INT(32):value:1

is the group number of the device to be measured. (The group corresponds to the physical enclosure.) Use the literal -1 to indicate all groups.

module

INT(32):value:1

is the module number of the device to be measured. Use the literal -1 to indicate all modules.

slot

INT(32):value:1

is the slot number of the device to be measured. Use the literal -1 to indicate all slots.

device^name

INT:value:12

is the name of the device to be measured. The name must be left-justified and blank-filled and must start with a dollar sign ($). Use a dollar sign and an asterisk ($*) to indicate all devices.

config^name

INT:value:32

is the configuration name of the disk or device, as defined by the system configuration database.

config^name can also be the configuration name of either a single ServerNet addressable controller (SAC) or an adapter that contains several ServerNet addressable controllers. In this case, the specification includes all devices of the requested type supported by that SAC or adapter.

config^name must be null-terminated and null-filled. An asterisk in the first byte indicates all configuration names.

To save space in the configuration table, you can omit CONFIG^NAME from the DEVICE^SVNET^DESC structure. You must set the descriptor length (LEN) to reflect whether you are using CONFIG^NAME because Measure uses the descriptor length to determine the start of the next descriptor in the configuration table. To not use CONFIG^NAME, set LEN to 44 and move only the first 44 bytes of the descriptor to the configuration table. To include CONFIG^NAME, set LEN to 108 and move the entire descriptor to the configuration table.

DEVICE^SVNET^DESC^G05

is the descriptor for DEVICE or DISC entities connected to a system running a G-series RVU by a Servernet/DA or FCSA . DEVICE^SVNET^DESC^G05 includes a field for a SCSI port identifier, which is necessary to uniquely identify devices connected by ServerNet/DA. You can use DEVICE^SVNET^DESC or DEVICE^SVNET^DESC^G05 to describe other NonStop S-series devices.

type

INT:*value*:1

is one of:

Entity Type	Literal	Numeric Identifier
DISC	DISC^T	7
DEVICE	IODEV^T	8

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. *len* is 52 if the descriptor does not include the configuration name or 116 if the descriptor includes the configuration name.

cpu^number

INT:value:1

is the number of the CPU on which the device is configured. Use the literal -1 to indicate all CPUs.

servernet

INT:value:1

is the ServerNet fabric. This field identifies a specific path or set of paths to a supported physical device. For G-series and later RVUs, use 0 to indicate the X fabric, 1 to indicate the Y fabric, or the literal -1 to indicate both fabrics.

For devices connected by a ServerNet/DA, use the literal -1.

GMS

is the physical location address (group, module, slot). The GMS structure consists of these fields:

group

INT(32):value:1

is the group number of the device to be measured. (The group corresponds to the physical enclosure.) Use the literal -1 to indicate all groups.

module

INT(32):value:1

is the module number of the device to be measured. Use the literal -1 to indicate all modules.

slot

INT(32):value:1

is the slot number of the device to be measured. Use the literal -1 to indicate all slots.

device^name

INT:value:12

is the name of the device to be measured. The name must be left-justified and blank-filled and must start with a dollar sign ($). Use a dollar sign and an asterisk ($*) to indicate all devices.

scsi^id

INT(64):value:1

is the SCSI port identifier of the device to be measured. Use the literal -1 to indicate all SCSI ports.

config^name

INT:value:32

is the configuration name of the device, as defined by the system configuration database.

config^name can also be the configuration name of either a single ServerNet addressable controller (SAC) or an adapter that contains several ServerNet addressable controllers. In this case, the specification includes all devices of the requested type that are supported by that SAC or adapter.

config^name must be null-terminated and null-filled. An asterisk in the first byte indicates all configuration names.

To save space in the configuration table, omit CONFIG^NAME from the DEVICE^SVNET^DESC^G05 structure. You must set the descriptor length (LEN) to reflect whether you are using CONFIG^NAME because Measure uses the descriptor length to determine the start of the next descriptor in the configuration table. To not use CONFIG^NAME, set LEN to 52 and move only the first 52 bytes of the descriptor to the configuration table. To include CONFIG^NAME, set LEN to 116 and move the entire descriptor to the configuration table.

DISKFILE^DESC

is the descriptor for DISKFILE entities. For OSS file pathnames, use the template DISKFILE^OSS^DESC.

This is the DISKFILE^DESC structure and a listing of the associated fields:

```
Struct  Diskfile^desc (*);  -- Diskfile descriptor declaration
 Begin
   Int  Type;               -- Diskfile entity type
   Int  Len;                -- byte length of the descriptor
   Int  Filecode;           -- file code
   Int  Filetype;           -- file type
```

```
   Int  File^name[0:11];    -- as it says
 End;
```

type

INT:value:1

is the literal value DISKFILE^T or the numeric identifier 19.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 32.

file^name

INT:value:12

is the name of the file to be measured. The name must be left-justified, blank-filled, and in local internal format. You can use asterisks (*) for the volume, subvolume, and file-name fields within *file^name*.

filetype

INT:value:1

is the type of the file to be measured. The four file types are:

File Type	Value
Unstructured	0
Relative	1
Entry-sequenced	2
Key-sequenced	3

filecode

INT:value:1

is the three-digit file code (for example, 100 for object files and 101 for edit files). For a complete list of file codes, see the *File Utility Program (FUP) Reference Manual*.

DISKFILE^OSS^DESC

is the descriptor for DISKFILE entities and supports the use of OSS file pathnames.

This is the DISKFILE^OSS^DESC structure and a listing of the associated fields that differ from DISKFILE^DESC :

```
Struct  Diskfile^OSS^desc (*);  -- Diskfile descriptor declaration
 Begin
 Diskfile^desc^fields;
 Int  Type;               -- Diskfile entity type
 Struct File^name^MID;    -- Internal identifier for OSS pathname
   Begin
   Int    Pathid[0:11];   -- Internal format OSS file pathname
   Int    Crvsn[0:2];     -- Creation Volume Sequence Number
   End;
 End;
```

file^name^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname for the specified disk file that belongs to an OSS file set. Initialize PATHID[0] to -1 to indicate all PATHIDs.

file^name^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

DISKFILE^ANSI^DESC

is the descriptor for DISKFILE entities and supports the use of ANSI SQL objects.

This is the DISKFILE^ANSI^DESC structure and a listing of the associated fields that differ from DISKFILE^DESC :

```
Struct  Diskfile^ANSI^desc (*)  -- Diskfile descriptor declaration
?IF PTAL
FIELDALIGN (SHARED2)
?ENDIF PTAL
;

Begin
   Int  Type;                          -- Diskfile entity type
   Int  Len;                           -- byte length of the descriptor
   Int  Filecode;                      -- file code
   Int  Filetype;                      -- file type
   Int  File^name[0:11];               -- as it says
   Struct File^name^MID( MID^def );    -- Internal identifier
                                       -- for OSS pathname
   Struct SQLMX^Obj^Desc (ANSI^SQL^object^def); -- ANSI SQL name
                                                -- selection info
End;
```

SQLMX^Obj^Desc

INT.value:12

is the internal format representation of the ANSI SQL object definition. This field is filled in using the MEAS_GETDESC_INFO_ procedure, the input for which is the actual ANSI SQL Name, while the output is the contents of this structure.

FILE^OPEN^DESC

The DISCOPEN entity uses the FILE^OPEN^DESC template for entity descriptors. For OSS file pathnames in the DISCOPEN entity, use the template FILE^OPEN^OSS^DESC.

The FILE entity uses entity descriptors:

- The FILE^OPEN^DESC template describes EDIT files, object files, and temporary files.
- For OSS file pathnames in the FILE entity, use the template FILE^OPEN^OSS^DESC.
- For ANSI SQL names in the FILE entity, use the template FILE^OPEN^ANSI^DESC.

Another entity descriptor, FILE^OPEN^DESC^D10 is used for FILE entity records and provides fields for describing the process owning the open and the opener process (its process name and program file).

Although you can specify FILE^NAME^MID without including OPENER^PROGRAM^FNAME^MID, you cannot use OPENER^PROGRAM^FNAME^MID by itself.

The FILE entity has separate entity descriptor formats, which are distinguished by their length.

Entity Descriptor	Number of Bytes
FILE^OPEN^DESC	46
FILE^OPEN^OSS^DESC	76
FILE^OPEN^DESC^D10	80

Entity Descriptor	Number of Bytes
FILE^OPEN^OSS^DESC^D10	110 or 140
FILE^OPEN^ANSI^DESC	164

This is the FILE^OPEN^DESC structure and a listing of the associated fields:

```
Struct File^open^desc (*);
   Begin
   Int    Type;               !Entity type
   Int    Len;                !Byte length of this record
   Int    Opener^CPU;         !Opener's CPU
   Int    Opener^PIN;         !Opener's PIN
   Int    File^number;        !File ACB number
   Int    OCB^number = File^number; ! Discopen OCB
   Int    File^name[0:11];    !File name being opened
   Int    DP^CPU;       ! Disc process' cpu number (primary or backup)
   Int    File^open^type = DP^CPU; !Type of file open
   Int    System^number;      !Of remote file or remote opener.
   Int    System^name[0:3]    !Of remote file or remote opener; Set
                              !to all blanks for local system files
                              !or openers
   End;
```

type

INT:value:1

Entity Type	Literal	Numeric Identifier
FILE	FILEOP^T	5
DISCOPEN	DFILEOP^T	6

is one of:

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 46.

opener^cpu

INT:value:1

is the number of the CPU on which the opener process is running. Use the literal -1 to indicate all CPUs.

opener^pin

INT:value:1

is the process identification number of the opener process. To indicate all PINs, use the literal -1.

Use *opener^cpu* and *opener^pin* to identify the process that is accessing the file to be measured.

file^number

INT:value:1

is the file identification number of the file to be measured. To indicate all files, use the literal -1.

ocb^number

INT:value:1

is no longer used.

file^name

INT:value:12

is the name of the file to be measured. The file name must be left-justified, blank-filled, and in local internal format. You can use asterisks (*) for the volume, subvolume, and file-name fields within *file^name*.

dp^cpu | File^open^type

INT:value:1

is the number of the CPU that contains the disk process (primary or backup) for a DISCOPEN specification. Starting with the G10 Measure PVU, *dp^cpu* is obsolete, replaced by *file^open^type*.

For a DISCOPEN entity, *file^open^type* should contain either a CPU number or the value -1 to mean all CPUs.

For a FILE entity, *file^open^type* can have any of the values:

```
ALLFILES      = %B0000000111111111,
GUARDIAN      = %B1111111111111111,
ENSCRIBE      = %B1111111011111111,
UNSTRUCT      = %B1111111011100001,
RELFILE       = %B1111111011100010,
ENTRYFILE     = %B1111111011100100,
KEYFILE       = %B1111111011101000,
SQLFILE       = %B1111111111100000,
PROCFILE      = %B1111111011110000,
OSS           = %B0000000011100000,
OSSDISK       = %B0111111011100000,
OSSFIFO       = %B1101111011100000,
OSSPIPE       = %B1011111011100000,
OSSSOCKET     = %B1110000011100000,
OSSUNIXSOCKET = %B1110011011100000,
OSSUNIXDGRAM  = %B1110111011100000,
OSSUNIXSTREAM = %B1111011011100000,
OSSINETSOCKET = %B1111100011100000,
OSSINETDGRAM  = %B1111101011100000,
OSSINETSTREAM = %B1111110011100000;
```

system^number

INT:value:1

is the number of the system on which the opener process is running. Use the literal -1 to indicate all systems.

system^name

INT:value:4

is the name of the system on which the opener process is running. The system name must be left-justified and blank-filled. It must start with a backslash (\) followed by a system name in internal format. Use a backslash and an asterisk (*) to indicate all systems.

FILE^OPEN^DESC^D10

is the descriptor for the FILE entity and provides fields for describing the process that owns the open and the opener process. It provides the process name and program name file. For OSS file pathnames, use the template FILE^OPEN^OSS^DESC^D10.

This is the FILE^OPEN^DESC^D10 structure and a listing of the associated fields:

```
Struct File^open^desc^d10 (*);
   Begin
```

```
File^open^desc^fields;
Int    Measflags;          !to be used later, initialize to 0 for
                           !now
Int    Opener^pname[0:3];  !opener's process name
Int    Opener^program^fname[0:11]; ! opener's program file name
End;
```

measflags

INT:value:1

is a field for future use. Initialize it to 0.

opener^pname

INT:value:4

is the name of the opener process. The process name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a process name in internal format. Use a dollar sign followed by an asterisk ($*) to indicate all processes.

opener^program^fname

INT:value:12

is the object file that the opener process is executing. The process name must be left-justified, blank-filled, and in local internal file name format. You can use asterisks (*) for the volume, subvolume, and file-name fields.

FILE^OPEN^OSS^DESC

is the descriptor for FILE or DISCOPEN entities on systems running G06.12 and later RVUs. You can use the FILE^OPEN^OSS^DESC descriptor if OSS file pathnames are required in entity specification.

This is the FILE^OPEN^OSS^DESC structure and a listing of the associated fields that differ from FILE^OPEN^DESC:

```
Struct File^open^oss^desc (*);
   Begin
   File^open^desc^fields;
   Struct File^name^MID;      !Internal identifier for OSS pathname
     Begin
     Int    Pathid[0:11];     !Internal format OSS file pathname
     Int    Crvsn[0:2];       !Creation Volume Sequence Number
     End;
   End;
```

file^name^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname. If the internal format representation was used to access the disk file, initialize Pathid[0] to -1 to indicate all PATHID values.

file^name^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

FILE^OPEN^OSS^DESC^D10

is the descriptor for the FILE entity and supports OSS file pathnames. The FILE^OPEN^OSS^DESC^D10 descriptor provides fields for describing the process that owns the open and the opener process. Including the OPENER^PROGRAM^FNAME^MID descriptor is optional.

This is the FILE^OPEN^OSS^DESC^D10 structure and a listing of the associated fields that differ from FILE^OPEN^DESC^D10:

```
Struct File^open^oss^desc^d10 (*);
   Begin
   File^open^desc^fields;
   Struct File^name^MID;      !Internal identifier for OSS pathname
     Begin
     Int    Pathid[0:11];
     Int    Crvsn[0:2];
     End;
   Int    Measflags;          !to be used later, initialize to 0 for
                              !now
   Int    Opener^pname[0:3];  !opener's process name
   Int    Opener^program^fname[0:11]; ! opener's program file name
   Struct Opener^program^fname^MID;    !Internal identifier for OSS
                                       !pathname
     Begin
     Int    Pathid[0:11];     !Internal format OSS file pathname
     Int    Crvsn[0:2];       !Creation Volume Sequence Number
     End;
   End;
```

file^name^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname. If the internal format representation was used to access the disk file, initialize Pathid[0] to -1 to indicate all PATHID values.

file^name^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

measflags

INT:value:1

is a field for future use; initialize to 0.

opener^pname

INT:value:4

is the name of the opener process. The process name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a process name in internal format. Use a dollar sign followed by an asterisk ($*) to indicate all processes.

opener^program^fname

INT:value:12

is the object file that the opener process is executing. The process name must be left-justified, blank-filled, and in local internal file-name format. You can use asterisks (*) for the volume, subvolume, and file-name fields.

opener^program^fname^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname for the OPENER^PROGRAM^FILENAME if the file belongs to an OSS file set. Initialize Pathid[0] to -1 to indicate all PATHID values.

opener^program^fname^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

FILE^OPEN^ANSI^DESC

is the descriptor for the FILE and DISCOPEN entities and specifies an ANSI SQL object. The FILE^OPEN^ANSI^DESC descriptor field is filled in using the MEAS_GETDESC_INFO_ procedure, the input for which is the actual ANSI SQL Name, while the output is the contents of this structure.

This is the FILE^OPEN^ANSI^DESC structure and a listing of the associated fields that differ from FILE^OPEN^DESC^D10:

```
Struct  File^open^ANSI^desc (*)  -- Diskfile descriptor declaration
?IF PTAL
FIELDALIGN (SHARED2)
?ENDIF PTAL
;

Begin
  Int    Type;               -- Entity type
  Int    Len;                -- Byte length of this record
  Int    Opener^CPU;         -- Opener's CPU
  Int    Opener^PIN;         -- Opener's PIN
  Int    File^number;        -- File ACB number
  Int    OCB^number = File^number; -- Discopen OCB
  Int    File^name[0:11];    -- File name being opened
  Int    DP^CPU;             -- Disc process' cpu number
  Int    File^open^type = DP^CPU;  -- Type of file open
  Int    System^number;      -- Of remote file or remote opener.
  Int    System^name[0:3]    -- Of remote file or remote opener;Set
                             -- to all blanks for local system files
                             -- or openers
  Struct File^name^MID( MID^def );            -- Internal id for OSS
                                              -- pathname
  Int    Measflags;                           -- To be used later,
                                              -- initialize to 0 for
                                              -- now
  Int    Opener^pname[0:3];                   -- opener's process name
  Int    Opener^program^fname[0:11];          -- opener's program file
                                              -- name
  Struct Opener^program^fname^MID( MID^def );  -- Internal id for OSS
                                               -- pathname
  Struct SQLMX^Obj^Desc (ANSI^SQL^object^def); -- ANSI SQL Name
                                               -- selection info

End;
```

SQLMX^Obj^Desc

INT.value:12

is the internal format representation of the ANSI SQL object definition. This field is filled in using the MEAS_GETDESC_INFO_ procedure, the input for which is the actual ANSI SQL Name, while the output is the contents of this structure.

OPDISK^DESC

is the descriptor for OPDISK entities on systems running D-series RVUs. The OPDISK entity type is currently not used on systems running G-series RVUs.

type

INT:value:1

is the literal OPDISK^T or the numeric identifier 17.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor.

cpu^number

INT:value:1

is the number of the CPU on which the optical disk is configured. Use the literal -1 to indicate all CPUs.

ctl

INT:value:1

is the controller number of the optical disk to be measured (0 through 31). Use the literal -1 to indicate all controllers.

unit

INT:value:1

is the unit number of the optical disk to be measured. Use the literal -1 to indicate all units.

device^name

INT:value:12

is the device name of the optical disk to be measured. The name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a device name in internal format. Use a dollar sign followed by an asterisk ($*) to indicate all devices.

volume^name

INT:value:12

is the name of the optical disk volume to be measured.

channel-num

INT:value:1

is the channel number of the optical disk to be measured.

OSSNS^DESC

is the descriptor for the OSSNS entity on systems running G-series RVUs. For OSS file pathnames, use the template PROCESS^OSS^DESC.

This is the OSSNS^DESC structure and a listing of the associated fields:

```
Struct OSSNS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record. In case of
                ! the name server it includes
                ! length of all the code-space
                ! descriptors and for userdef
                ! it includes length of all the counter
                ! descriptors.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11];
   End;
```

type

INT:value:1

is the literal OSSNS^T or the numeric identifier 21.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 40.

cpu^number

INT:value:1

is the number of the CPU on which the name server is running. Use the literal -1 to indicate all CPUs.

pin

INT:value:1

is the process identification number of the name server. You can use these literal values:

- -1 to indicate all PINs
- -2 to indicate all system processes
- SYSPROCS to indicate all processes installed by SYSGEN

process^name

INT:value:4

is the OSS name server name. The name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a process name in internal format. *process^name* can be a system process name. Use a dollar sign followed by an asterisk ($*) to indicate all name servers.

program^file^name

INT:value:12

is the object file that the OSS name server is executing. The name must be left-justified and blank-filled and must be in local internal file name format. You can use asterisks (*) in the volume, subvolume, and file-name fields within *file^name*. However, the volume name must begin with a dollar sign ($). For example, to specify all program file names, use the value "$* * *".

PROCESS^DESC

is the descriptor for PROCESS entity on systems running D-series, G-series, or later RVUs. For OSS file pathnames, use the template PROCESS^OSS^DESC.

This is the PROCESS^DESC structure and a listing of the associated fields:

```
Struct Process^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record. In case of
                ! process it includes
                ! length of all the code-space
                ! descriptors and for userdef
                ! it includes length of all the counter
                ! descriptors.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11]
   End;
```

type

INT:value:1

is the literal PROCESS^T or the numeric identifier 2.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 40.

cpu^number

INT:value:1

is the number of the CPU on which the process is running. Use the literal -1 to indicate all CPUs.

pin

INT:value:1

is the process identification number of the process. You can use these literal values:

- -1 to indicate all PINs
- -2 to indicate all system processes
- SYSPROCS to indicate all processes installed by SYSGEN

process^name

INT:value:4

is the process name. The name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a process name in internal format. *process^name* can be a system process name. Use a dollar sign followed by an asterisk ($*) to indicate all processes.

program^file^name

INT:value:12

is the object file that the process is executing. The name must be left-justified and blank-filled and must be in local internal file-name format. You can use asterisks (*) in the volume, subvolume, and file-name fields within *file^name*. However, the volume name must begin with a dollar sign ($). For example, to specify all program file names, use the value "$* * *".

PROCESS^OSS^DESC

is the descriptor for PROCESS entity and supports OSS file pathnames.

This is the PROCESS^OSS^DESC structure and a listing of the associated fields that differ from PROCESS^DESC:

```
Struct Process^OSS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record. In case of
                ! process it includes
                ! length of all the code-space descriptors
                ! and for userdef it includes length of
                !all the counter descriptors.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11];
   Struct Program^file^name^MID; !Internal identifier for
                                 !OSS pathname
     Begin
     Int    Pathid[0:11]; !Internal format OSS file
                          !pathname
     Int    Crvsn[0:2];   !Creation Volume Sequence Number
     End;
   End;
```

type

INT:value:1

is the literal PROCESS^T or the numeric identifier 2.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 70.

cpu^number

INT:value:1

is the number of the CPU on which the process is running. Use the literal -1 to indicate all CPUs.

pin

INT:value:1

is the process identification number of the process. You can use these literal values:

- -1 to indicate all PINs
- -2 to indicate all system processes
- SYSPROCS to indicate all processes installed by SYSGEN

process^name

INT:value:4

is the process name. The name must be left-justified and blank-filled. It must start with a dollar sign ($) followed by a process name in internal format. *process^name* can be a system process name. Use a dollar sign followed by an asterisk ($*) to indicate all processes.

program^file^name

INT:value:12

is the object file that the process is executing. The name must be left-justified and blank-filled and must be in local internal file-name format. You can use asterisks (*) in the volume, subvolume, and file-name fields within *file^name*. However, the volume name must begin with a dollar sign ($). For example, to specify all program file names, use the value "$* * *".

program^file^name^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname for the program file. If the file belongs to the OSS file system, initialize Pathid[0] to -1 to indicate all PATHID values.

program^file^name^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

PROCESSH^DESC

is the descriptor for PROCESSH entities on systems running D-series or G-series RVUs. PROCESSH entity descriptors are variable in length and can contain CODE^SPACE^DESC templates following the initial PROCESSH^DESC structure. For OSS file pathnames, use the PROCESSH^OSS^DESC and CODE^SPACE^OSS^DESC templates.

This is the PROCESSH^DESC structure and a listing of the associated fields:

```
Struct Processh^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record. In case of processh it includes
                ! length of all the code-space descriptors and for userdef
                ! it includes length of all the counter descriptors.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11]
   Fixed  Conftime;    !Set to 0F, for use by MEASFH only
   Int    Code^spaces; !Number of code spaces to be measured
   End; ! A subsequent code^space^desc is not needed for an Entitydesc of
        ! a previously installed entity.  But for Contab use, as many
        ! code^space^desc as desired must imediately follow this struct.
```

type

INT:value:1

is the literal value PROCESSH^T or the numeric identifier 3.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor and any CODE^SPACE^DESC descriptors associated with it. LEN is 50 if the descriptor does not contain any CODE^SPACE^DESC extensions.

cpu^number

INT:value:1

is the number of the CPU on which the process to be measured is running. Use the literal -1 to indicate all CPUs.

pin

INT:value:1

is the process identification number of the process to be measured. You can use these literal values:

- -1 to indicate all processes
- SYSPROCS to indicate all processes installed by SYSGEN
- ALLTIMES to measure execution time for all processes
- ALLINTR to measure only interrupt time for all processes

If you specify *pin* as the literal value ALLTIMES or ALLINTR, you must use *code-space* to specify a code space of SC.0, SL.*n*, SCR, or SLR.

process^name

INT:value:4

is the name of the process to be measured. The name must be left-justified and blank-filled. It must consist of a dollar sign ($) followed by a process name in internal format. *process^name* can be a system process name. Use the literal -1 to indicate all process names.

program^file^name

INT:value:12

is the name of the object file that the process to be measured is executing. The name must be left-justified, blank-filled, and in local internal file-name format. You can use asterisks (*) in the volume, subvolume, and file-name fields within *file^name*. However, the

volume name must begin with a dollar sign ($). For example, to specify all program file names, use the value "$* * *".

conftime

FIXED:value:1

Specify *conftime* as 0F. For use by the Measure subsystem only.

code^spaces

INT:value:1

is the number of CODE^SPACE^DESC descriptors that follow, describing the code spaces to be measured.

NOTE: When passing a PROCESSH^DESC descriptor to the MEASREAD procedure, do not include the CODE^SPACE^DESC descriptors.

CODE^SPACE^DESC

is the descriptor for a code space specification.

This is the CODE^SPACE^DESC structure and a listing of the associated fields:

```
Struct Code^space^desc (*);
   Begin
   Int      Code^space;    !<0>=0 user,        =1 system
                           !<1>=0 non lib,     =1 lib
                           !<2>=0 non native,  =1 native
                           !<7>=0 no OSS,      =1 OSS MID included.
                           !<8:15> space number
   Int      Code^ranges^file^name[0:11];
   String   Code^ranges^file^name^s = Code^ranges^file^name;
   End;
```

code^space

INT:value:1

is the code space number, formatted as:

.0:2 identifies the space. On H-series and J-series RVUs, this value is significant only for nonnative code.

0 = user code — TNS or accelerated (UC)
1 = user code — TNS/R native (UCR)
2 = user library — TNS or accelerated (UL)
3 = user library — TNS/R native (ULR)
4 = system code — TNS or accelerated (SC)
5 = system code — TNS/R native (SCR)
6 = system library — TNS or accelerated (SL)
7 = system library — TNS/R native (SLR)
.3:7 is reserved for future use.
.7 indicates code space template is CODE^SPACE^OSS^DESC format.
.8:15 is relevant only to TNS and accelerated code spaces, and indicates the space number:

UC	0 through 15
UL	0 through 15
SC	0
SL	0 through 31

code^ranges^file^name

INT:value:12

is the name of the file containing the code to be measured. *code^ranges^file^name* must be a TNS object file, EDIT file, or Executable and Linking Format (ELF) file. The name must be left-justified, blank-filled, and in local internal format.

If *code^ranges^file^name* is an object file, MEASFH examines the procedure names and addresses and adds each procedure to the configuration. If *code^ranges^file^name* is an EDIT file, you must specify code ranges in ascending address order:

code-range-name code-range-address

code-range-name

is the procedure name of the code range to be measured. Leading dollar signs ($) are allowed for TNS/R and TNS/E code ranges.

code-range-address

is the address of the code range to be measured.

For TNS EDIT files, *code-range-address* is an address in octal or an offset from a preceding octal address.

For TNS/R and TNS/E EDIT files, *code-range-address* is a virtual address in hexadecimal or an offset from a preceding hexadecimal address.

To indicate an offset from the previous *code-range-address*, precede the octal or hexadecimal address with a plus sign (+). The last *code-range-address* specified without a plus sign is added to the offset to yield an effective address.

PROCESSH^OSS^DESC

is the descriptor for PROCESSH entities in G06.12 and later RVUs and supports OSS file pathnames. PROCESSH entity descriptors are variable in length and can contain some number of CODE^SPACE^DESC or CODE^SPACE^OSS^DESC templates following the initial PROCESSH^OSS^DESC descriptor. These templates can be used in any order. A bit in the CODE^SPACE field (bit 7) indicates the format of the descriptor and the offset to the next descriptor.

NOTE: The initializing of PROCESSH^DESC.CONFTIME to 0F on input determines whether PROCESSH^DESC or PROCESSH^OSS^DESC is used.

This is the PROCESSH^OSS^DESC structure and a listing of the associated fields that differ from PROCESSH^DESC:

```
Struct Processh^OSS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record. In case of processh it
                ! includes length of all the code-space descriptors
                ! and for userdef it includes length of all the
                ! counter descriptors.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11]
   Struct Program^file^name^MID; !Internal identifier for OSS pathname
     Begin
     Int    Pathid[0:11]; !Internal format OSS file pathname
     Int    Crvsn[0:2];   !Creation Volume Sequence Number
     End;
   Fixed  Conftime;    !Set to 0F, for use by MEASFH only
   Int    Code^spaces; !Number of code spaces to be measured
   End; ! A subsequent code^space^desc is not needed for an Entitydesc
```

```
        ! of a previously installed entity.  But for Contab use, as
        ! many code^space^desc as desired must imediately follow this
        ! struct.
```

len

INT:value:1

is the length in bytes of this descriptor and any CODE^SPACE^OSS^DESC descriptors associated with it. LEN is 80 if the PROGRAM^FILE^NAME^MID structure is included but does not include the CODE^SPACE^DESC or CODE^SPACE^OSS^DESC extensions.

program^file^name^mid.pathid

INT.value:12

is the internal format representation of the OSS file pathname for a program file that belongs to the OSS file system. Initialize Pathid[0] to -1 to indicate all PATHIDs.

program^file^name^mid.crvsn

INT.value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. Initialize CRVSN[0] to -1 to indicate all CRVSN values.

conftime

FIXED:value:1

Specify *conftime* as 0F. For use by the Measure subsystem only.

code^spaces

INT:value:1

is the number of CODE^SPACE^OSS^DESC or CODE^SPACE^DESC descriptors that follow, describing the code spaces to be measured.

NOTE: When passing a PROCESSH^DESC descriptor to the MEASREAD procedure, do not include the CODE^SPACE^DESC or CODE^SPACE^OSS^DESC descriptors.

CODE^SPACE^OSS^DESC

is the descriptor for code space specification and supports OSS file pathnames.

This is the CODE^SPACE^OSS^DESC structure and a listing of the associated fields that differ from CODE^SPACE^DESC:

```
Struct Code^space^oss^desc (*);
                          !To be used in place of code^space^desc
                          !structure when OSS file pathname has
                          ! been used in the code^range^file^name
                          ! specification.
  Begin
  Int      Code^space;    !<0>=0 user,       =1 system
                          !<1>=0 non lib,    =1 lib
                          !<2>=0 non native, =1 native
                          !<7>=0 no OSS,     =1 OSS MID included.
                          !<8:15> space number
  Int      Code^ranges^file^name[0:11];
  String   Code^ranges^file^name^s = Code^ranges^file^name;
  Struct   Code^ranges^file^name^MID; !Internal identifier for OSS
                                      !pathname
    Begin
    Int    Pathid[0:11]; !Internal format OSS file pathname
    Int    Crvsn[0:2];   !Creation Volume Sequence Number
    End;
  End;
```

code^space

INT:value:1

is the code space number, formatted as:

.0:2 identifies the space. On H-series and J-series RVUs, this value is significant only for nonnative code.

0 = user code — TNS or accelerated (UC)
1 = user code — TNS/R native (UCR)
2 = user library — TNS or accelerated (UL)
3 = user library — TNS/R native (ULR)
4 = system code — TNS or accelerated (SC)
5 = system code — TNS/R native (SCR)
6 = system library — TNS or accelerated (SL)
7 = system library — TNS/R native (SLR)
.3:7 is reserved for future use.
.7 indicates code space template is CODE^SPACE^OSS^DESC format.
.8:15 is relevant only to TNS and accelerated code spaces, and indicates the space number:

UC	0 through 15
UL	0 through 15
SC	0
SL	0 through 31

code^ranges^file^name^mid.pathid

INT:value:12

is the internal representation of the OSS file pathname for code file. To indicate all PATHIDs, initialize PATHID[0] to -1.

code^ranges^file^name^mid.crvsn

INT:value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. To indicate all CRVSN values, initialize CRVSN[0] to -1.

SQLPROC^DESC

is the descriptor for the SQLPROC entity. For OSS file pathnames, use the template SQLPROC^OSS^DESC.

This is the SQLPROC^DESC structure and a listing of the associated fields:

```
Struct Sqlproc^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11]

   End;
```

type

INT:value:1

is the literal SQLPROC^T or the numeric identifier 15.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 40.

cpu^number

INT:value:1

is the number of the CPU on which the process to be measured is running. To indicate all CPUs, use the literal -1.

pin

INT:value:1

is the process identification number of the process to be measured. To indicate all PINs, use the literal -1.

process^name

INT:value:4

is the name of the process to be measured. The name must be left-justified and blank-filled. It must consist of a dollar sign ($) followed by a process name in internal format. To indicate all processes, use a dollar sign followed by an asterisk ($*).

program^file^name

INT:value:12

is the name of the object file that the process to be measured is executing. The name must be left-justified, blank-filled, and in local internal format. You can use asterisks (*) to specify the volume, subvolume, and file-name fields within *program^file^name*.

SQLPROC^OSS^DESC

is the descriptor for the SQLPROC entity and supports OSS file pathnames.

This is the SQLPROC^OSS^DESC structure and a listing of the associated fields that differ from SQLPROC^DESC:

```
Struct Sqlproc^OSS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11];
   Struct Program^file^name^MID; !Internal identifier for OSS pathname
     Begin
     Int    Pathid[0:11]; !Internal format OSS file pathname
     Int    Crvsn[0:2];   !Creation Volume Sequence Number

      End;
   End;
```

len

INT:value:1

is the length in bytes of this descriptor. LEN is 40 if the descriptor does not contain the PROGRAM^FILENAME^MID structure. If the descriptor does contain the PROGRAM^FILENAME^MID structure, LEN is 70.

program^file^name^mid.pathid

INT:value:12

is the internal format representation of an OSS file pathname for the specified program file name if the file is part of an OSS file set. To indicate all PATHIDs, initialize PATHID[0] to -1.

program^file^name^mid.crvsn

INT:value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. To indicate all CRVSN values, initialize CRVSN[0] to -1.

SQLSTMT^DESC

is the descriptor for the SQLSTMT entity. For OSS file pathnames, use SQLSTMT^OSS^DESC. For ANSI SQL file names, use SQLSTMT^ANSI^DESC.

NOTE: When the measurement is being configured, only the process part of the descriptor is used; when doing a the listing, all the fields can be used.

This is the SQLSTMT^DESC structure and a listing of the associated fields:

```
Struct Sqlstmt^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11]
   Int    Run^unit[0:15]; !procedure or run-unit name
   Int    Stmt^index;     !statement index number within the run-unit
   String Run^Unit^s = Run^unit;
   End;
```

type

INT:value:1

is the literal SQLSTMT^T or the numeric identifier 16.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. LEN is 74.

cpu^number

INT:value:1

is the number of the CPU on which the process to be measured is running.

pin

INT:value:1

is the process identification number of the process to be measured. You can use these literal values:

- -1 to indicate all processes
- SYSPROCS to indicate all processes installed by SYSGEN

process^name

INT:value:4

is the name of the process to be measured. The name must be left-justified and blank-filled. It must consist of a dollar sign ($) followed by a process name in internal format. To indicate all processes, use a dollar sign and an asterisk ($*).

program^file^name

INT:value:12

is the name of the object file that the process to be measured is executing. The name must be left-justified, blank-filled, and in local internal format. You can use asterisks (*) in the volume, subvolume, and file-name fields within *file^name*. However, the volume name must begin with a dollar sign ($). For example, to specify all program file names, use the value "$* * *".

run-unit

INT:value:32

is the name of the procedure that contains the SQL statement to measure. To include all run-units, the name of the procedure should be blank-filled. Do not use the asterisk (*) wild-card character.

Not used for ANSI SQL.

statement^index

INT:value:1

is the statement index number within the specified run unit (procedure).

Not used for ANSI SQL.

SQLSTMT^OSS^DESC

is the descriptor for the SQLSTMT entity and includes support for OSS file pathnames, SQL/MP procedures, or SQL/MX modules.

NOTE: When the measurement is being configured, only the process part of the descriptor is used; when doing a the listing, all the fields can be used.

This is the SQLSTMT^OSS^DESC structure and a listing of the associated fields that differ from SQLSTMT^DESC:

```
Struct Sqlstmt^OSS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11];
   Struct Program^file^name^MID; !Internal identifier for OSS pathname
      Begin
      Int    Pathid[0:11]; !Internal format OSS file pathname
      Int    Crvsn[0:2];   !Creation Volume Sequence Number
     End;
   Int    Run^unit[0:63]; !procedure or run-unit name
   Int    Stmt^index;     !statement index number within the run-unit
   String Run^Unit^s = Run^unit;
 End;
```

program^file^name^mid.pathid

INT:value:12

is the internal format representation of an OSS file pathname for the specified program file name. To indicate all PATHIDs, initialize PATHID[0] to -1.

program^file^name^mid.crvsn

INT:value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. To indicate all CRVSN values, initialize CRVSN[0] to -1.

SQLSTMT^ANSI^DESC

is the descriptor for the SQLSTMT entity and includes support for ANSI SQL file names.

NOTE: When an ANSI SQL name is copied into the descriptor's *run^unit* field, it should not be enclosed in single quotes.

NOTE: When the measurement is being configured, only the process part of the descriptor is used; when doing a the listing, all the fields can be used.

This is the SQLSTMT^ANSI^DESC structure and a listing of the associated fields:

```
Struct Sqlstmt^ANSI^desc (*)
?IF PTAL
FIELDALIGN (SHARED2)
?ENDIF PTAL
;
  Begin                  ! used for SQL/MX ANSI SQL name run-unit
                         !  comparisons only.
  process^oss^desc^fields;

  Int    Version;        ! Version of Sqlstmt desc; added in G11
                         ! Use the literal sqlstmt^ANSI^desc^version
  Int    Stmt^index;     ! statement index number within the run-unit
  Int    Run^unit^name^len;
  Int    Minimum^ANSI^Run^unit[0:63];      ! desc is at least this long...
  Int    Run^unit = Minimum^ANSI^Run^unit; ! up to
                                           ! max^wlen^ANSI^RUN^UNIT^NAME
  String Run^Unit^s = Run^unit;
  End;
```

version

INT:value:1

is the sqlstmt^ANSI^desc^version to indicate this type of descriptor.

Stmt^index

INT:value:1

is the statement index number within the ANSI SQL module.

Run^unit^name^len

INT:value:1

is the actual length of the ANSI SQL module name in the *run^unit* field.

NOTE: The ANSI SQL module name can be both shorter than and longer than 128 bytes. If it is longer than 128 bytes, you must make sure there is memory allocated for the "extra" space.

Minimum^ANSI^Run^unit[0:(*min^wlen^ANSI^SQLNAME* - 1)];

INT:value:64

where *min^wlen^ANSI^SQLNAME* is 64 words, which is the minimum length of this field.

Run^unit = Minimum^ANSI^Run^unit

is the word representation of the run unit name.

Run^Unit^s = Run^unit

is the string representation of the run unit name.

NOTE: The run unit name can be any valid ANSI SQL module name, without enclosing single quotes.

SVNET^DESC

is the descriptor for the SERVERNET entity on systems running G-series RVUs.

The SVNET^DESC descriptor consists of these fields, in order:

type

INT:value:1

is the literal value SVNET^T or the numeric identifier 18.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor. *len* is 28 for an SVNET^DESC descriptor that does not include the configuration name or 92 for an SVNET^DESC descriptor that includes the configuration name.

cpu^number

INT:value:1

is the number of the CPU on which the SAC to be measured is configured. To indicate all CPUs, use the literal -1.

ctrl^type

INT:value:1

is the HP product number for this SAC type. Use the literal -1 to indicate all SAC types. For information about the product number, see the documentation specific to each controller.

GMS

is the physical location address (group, module, slot) of the SAC to be measured. The GMS structure consists of these fields:

group

INT(32):value:1

is the group number. (The group corresponds to the physical enclosure.) To indicate all groups, use the literal -1.

SvNet^node^number

INT(32):value:1 redefines group

is the ServerNet node number of the system with which the entity is communicating. To indicate all ServerNet cluster systems, use the literal -1.

module

INT(32):value:1

is the module number. To indicate all modules, use the literal -1.

slot

INT(32):value:1

is the slot number. To indicate all slots, use the literal -1.

subdevice

INT(32):value:1

is the subdevice number of the SAC to be measured. Use the literal -1 to indicate all SACs. For information about subdevice numbers, see the documentation specific to each SAC.

remote^CPU

INT(32):value:1 redefines subdevice

is the number of the CPU on a remote ServerNet cluster with which the entity is communicating. To indicate all remote CPUs, use the literal -1.

PF

is the port-fiber structure for CLIM. The PF structure consists of these fields:

port

INT:value:1

is the port number of the CLIM to be measured. To indicate all CLIMs, use the literal -1.

fiber

INT:value:1

is the fiber number of the of the CLIM to be measured. To indicate all CLIMs, use the literal -1.

node^class

INT(32):value:1

is the general class of SAC to be measured. The node-class name must be enclosed in quotation marks and blank-filled to four bytes:

```
node^class^clim = "CLIM",
node^class^clim^ip = "CLMI",
node^class^clim^storage = "CLMS",
node^class^clim^open = "CLMO",
node^class^scsi = "SCSI",
node^class^nioc = "NIOC",
node^class^lan = "ENET",
node^class^wan = "SWAN",
node^class^ipc = "IPC",
node^class^ripc = "RIPC",
node^class^colo = "COLO",
node^class^mont = "MONT",
node^class^all = "* ";
```

config^name

INT:value:32

is the configuration name of the SAC to be measured, as defined by the system configuration database.

config^name can also be the configuration name of an adapter that contains several ServerNet addressable controllers. In this case, the specification includes all devices of the requested type that are supported by that adapter.

config^name must be null-terminated and null-filled. An asterisk in the first byte indicates all configuration names.

To save space in the configuration table, you can omit CONFIG^NAME from the SVNET^DESC structure. You must set the descriptor length (LEN) to reflect whether or not you are using CONFIG^NAME because Measure uses the descriptor length to determine the start of the next descriptor in the configuration table. To not use CONFIG^NAME, set LEN to 28 and move only the first 28 bytes of the descriptor to the configuration table. To include CONFIG^NAME, set LEN to 92 and move the entire descriptor to the configuration table.

SYSTEM^DESC

is the descriptor for the SYSTEM or CLUSTER entities on systems running D-series or G-series RVUs.

The SYSTEM^DESC descriptor consists of these fields, in order:

type

INT:value:1

Entity Type	Literal	Numeric Identifier
SYSTEM	REMSYS^T	11
CLUSTER	CLUSTER^T	12

is one of:

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor.

lh^cpu

INT:value:1

is the number of the CPU that contains the line handler (primary or backup). To indicate all CPUs, use the literal -1.

system^number

INT:value:1

is the system number of the remote system from which network traffic is to be measured. To indicate all systems, use the literal -1.

system^name

INT:value:4

is the system name of the remote system from which network traffic is to be measured. The name must be left-justified and blank-filled. It must consist of a backslash (\) followed by a system name in internal format. To indicate all systems, use a backslash followed by an asterisk (*).

USERDEF^DESC

is the descriptor for the USERDEF entity on both D-series and G-series RVUs. The USERDEF entity also uses the descriptor COUNTER^DESC. For OSS file pathnames, use the USERDEF^OSS^DESC template.

This is the USERDEF^DESC structure and a listing of the associated fields:

```
Struct Userdef^desc (*);
  Begin
  Int    Type; !Entity type
  Int    Len;  ! Byte length of record.
  Int    CPU^number;
  Int    PIN;
```

```
  Int    Process^name[0:3];
  Int    Program^file^name[0:11];
  Int    Numcounters;  !Number of counters to be installed
  End;  ! A subsequent counter^desc is not needed for an Entitydesc of
        ! a previously installed entity.  But for Contab use, as many
        ! Counter^desc as desired must imediately follow this struct.
```

type

INT:value:1

is the literal USERDEF^T or the numeric identifier 4.

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor and any COUNTER^DESC descriptors associated with it. See COUNTER^DESC.

cpu^number

INT:value:1

is the number of the CPU on which the process to be measured is running.

pin

INT:value:1

is the process identification number of the process to be measured. To indicate all PINs, use the literal -1.

process^name

INT:value:4

is the name of the process to measure. The name must be left-justified and blank-filled. It must start with a dollar sign ($) and be in internal format. To indicate all processes, use a dollar sign followed by an asterisk ($*).

program^file^name

INT:value:12

is the name of the object file that the process to be measured is executing. The name must be left-justified, blank-filled, and in local internal format. To specify the volume, subvolume, and file-name fields within *program^file^name*, use asterisks (*).

numcounters

INT:value:1

is the number of COUNTER^DESC descriptors that follow. Each COUNTER^DESC descriptor describes a user-defined counter to measure.

NOTE: When passing a USERDEF^DESC descriptor to the MEASREAD or MEASREADACTIVE procedure, do not include the COUNTER^DESC descriptors.

COUNTER^DESC

describes a user-defined counter specification. OSS file pathnames use the COUNTER^DESC template.

This is the COUNTER^DESC structure and a listing of the associated fields:

```
Struct Counter^desc (*);
                          ! Numcounters of these structs go here
   Begin                  ! when used as a Contab descriptor
   Int    Name[0:7];  !User assigned counter name
   Int    Type;       !1=accum, 2=busy, 3=queue
```

```
   Int    Index;      !0:index counters will be allocated
   End;
```

name

INT:value:8

is a left-justified, blank-filled counter name.

type

INT:value:1

is one of these literal values:

ACCUM	Accumulating counter, 32 bits
FACCUM	Accumulating counter, 64 bits
BUSY	Busy counter
QUEUE	Queue counter
QBUSY	Queue busy counter
TCELLBUSY	Busy counter, maintained with timer cells
TCELLQUEUE	Queue counter, maintained with timer cells
TCELLQBUSY	Queue busy counter, maintained with timer cells

index

INT:value:1

is the highest index value needed to address the array of counters associated with this counter name. If the counter is a single value, set *index* to 0.

USERDEF^OSS^DESC

is the descriptor for the USERDEF entity in G06.12 and later RVUs. Both the USERDEF^OSS^DESC and the COUNTER^DESC templates are used for the USERDEF entity.

This is the USERDEF^OSS^DESC structure and a listing of the associated fields that differ from USERDEF^DESC:

```
Struct Userdef^OSS^desc (*);
   Begin
   Int    Type; !Entity type
   Int    Len;  ! Byte length of record.
   Int    CPU^number;
   Int    PIN;
   Int    Process^name[0:3];
   Int    Program^file^name[0:11];
   Struct Program^file^name^MID; !Internal identifier for OSS pathname
     Begin
     Int    Pathid[0:11]; !Internal format OSS file pathname
     Int    Crvsn[0:2];   !Creation Volume Sequence Number

      End;
   Int    Numcounters;  !Number of counters to be installed
   End; ! A subsequent counter^desc is not needed for an Entitydesc of
        ! a previously installed entity.  But for Contab use, as many
        ! Counter^desc as desired must imediately follow this struct.
```

program^file^name^mid.pathid

INT:value:12

is the internal format representation of an OSS file pathname for the specified program file name. To indicate all PATHID values, initialize PATHID[0] to -1.

program^file^name^mid.crvsn

INT:value:3

is the creation version serial number that identifies a unique instance of an OSS disk file. To indicate all CRVSN values, initialize CRVSN[0] to -1.

WAN^DESC

is the descriptor for LINE, NETLINE, or TERMINAL entities on systems running G-series RVUs.

The WAN^DESC descriptor consists of these fields, in order:

type

INT:value:1

is one of:

Entity Type	Literal	Numeric Identifier
LINE	LINE^T	9
NETLINE	NETLINE^T	10
TERMINAL	TERM^T	13

A positive *type* value includes the specified entities in the measurement configuration. A negative *type* value excludes the specified entities from measurement. For an example, see Specifying Entity Descriptors (page 352).

len

INT:value:1

is the length in bytes of this descriptor.

cpu^number

INT:value:1

is the number of the CPU on which the measured line is configured. To indicate all CPUs, use the literal -1.

trackid

INT:value:3

is the 6-byte ASCII identifier reported for the 3880 controller by the SCF subsystem (and usually printed on an external label as well). To indicate all TRACKIDs, use an asterisk (*) in the first byte and blank-fill the rest of the field.

clip

INT:value:1

is the number of the communications line interface processor (CLIP) being measured within the SWAN controller (1 through 3 for SWAN I, 1 through 6 for SWAN II). To indicate all CLIPs, use the literal -1.

line

INT:value:1

is the specific line controlled by a CLIP within a SWAN controller (0 or 1). To indicate both lines, use the literal -1.

device^name

INT:value:12

is the device name of the line being measured. The name must be left-justified and blank-filled. It must start with a dollar sign ($) and be in internal format. To indicate all devices, use a dollar sign followed by an asterisk ($*).

device^type

INT:value:1

is the device type of a specific subsystem, such as 61 (X.25), as reported in the SCF LISTDEV listing. To indicate all devices, use the literal -1.

device^subtype

INT:value:1

is the device subtype of a specific subsystem, such as 62 (X25AM), as reported in the SCF LISTDEV listing. To indicate all devices, use -1.

CONTAB^TRAILER

marks the end of the *contab* array. The CONTAB^TRAILER consists of these fields, in order:

type

INT:ref:1

identifies the trailer section of the *contab* array. Specify *type* as the literal CONTAB^TRAILER^T or the numeric identifier 51.

len

INT:value:1

is the length in bytes of the CONTAB^TRAILER descriptor (typically four).

Related Procedures

To start a measurement, use the MEASCONTROL procedure.

MEASCONTROL

Starts and stops a measurement, including a measurement interval as an option.

Before calling MEASCONTROL, your program must call MEASCONFIGURE and configure the measurement.

```
error := MEASCONTROL ( meascb             ! i,o
                     ,measnum             ! i
                     ,[ starttime ]       ! i
                     ,[ stoptime  ]       ! i
                     ,[ interval ] );     ! i
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. After you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

measnum

input

INT:value

is the measurement number. Use the *measnum* value returned by the MEASCONFIGURE procedure.

starttime

input

FIXED:value

is the start time of the measurement. If you omit it or specify it as -1, the measurement is configured but not started. If you specify *starttime* and *stoptime*, *starttime* must be less than *stoptime*. Specify *starttime* as a Julian date based on local civil time (LCT) in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP. See the *Guardian Procedure Calls Reference Manual*.

stoptime

input

FIXED:value

is the stop time of the measurement. If you omit it or specify it as -1, the measurement runs until a stop time is supplied by a subsequent call. If you specify *starttime* and *stoptime*, *starttime* must be less than *stoptime*. Specify *stoptime* as a Julian date based on LCT in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP. See the *Guardian Procedure Calls Reference Manual*.

interval

input

FIXED:value

is the collection interval. Counter values are written to the data file at the specified time intervals, when a transient entity starts or when a transient entity stops. If you omit *interval*, counter values are written to the data file twice: once at the beginning of the measurement and once at the end. Specify *interval* in microseconds. You cannot change the collection interval after starting the measurement.

MEASCOUNTERBUMP

Modifies the counter identified by *offset* provided that:

- The counter is part of a currently active measurement. If the counter is not currently being measured, MEASCOUNTERBUMP returns ERR^UDCNOTPRESENT.
- The counter type is compatible with *bumptype*. You specify the counter type when you add the counter to the measurement configuration.

For more information about user-defined counters, see MEASCOUNTERBUMPINIT (page 397).

```
error := MEASCOUNTERBUMP ( offset           ! i
                         ,bumptype          ! i
                         ,[ addvalue ]      ! i
                         ,[ index ]         ! i
                         ,[ doubleadd ] )   ! i
                         ,[ quadadd ] );    ! i
```

offset

input

INT:value

is the offset of the counter value within the internal counter record. Use the *offset* value returned by the MEASCOUNTERBUMPINIT procedure.

bumptype

input

INT:value

is a literal value, which indicates the counter bump action. *bumptype* must be compatible with counter type.

Value	Action	Counter type
INC	Increment counter	Accumulating
ADD	Add *addvalue* to counter	Accumulating
SETBUSY	Set counter to busy	Busy or TCELLBUSY
RESETBUSY	Set counter to not busy	Busy or TCELLBUSY
INCQUEUE	Increase (queue) counter	Queue or TCELLQUEUE
DECQUEUE	Decrease (queue) counter	Queue or TCELLQUEUE
INCQBUSY	Increase (queue busy) counter	Queue busy (QBUSY or TCELLQBUSY)
DECQBUSY	Decrease (queue busy) counter	Queue busy (QBUSY or TCELLQBUSY)

addvalue

input

INT:value

is the value (positive or negative) to add to the counter. Applicable only if you specified *bumptype* as ADD and the counter type is ACCUM or FACCUM. It cannot be specified if the *doubleadd* parameter is also specified.

index

input

INT:value

is an index into an array of counters. The maximum index value is 127. By default, a counter name is associated with a single counter value (index value 0). You can also associate a counter name with an array of counter values.

doubleadd

input

INT(32):value

is the value (positive or negative) to add to the counter. Applicable only if you specified *bumptype* as ADD and the counter type as FACCUM. *Doubleadd* cannot be specified if *addvalue* is also specified.

quadadd

input

FIXED:value

is the value (positive or negative) to add to a 64-bit counter. Applicable only if you specified *bumptype* as ADD and the counter type as FACCUM. *Doubleadd* cannot be specified if *addvalue* is also specified.

MEASCOUNTERBUMPINIT

Determines whether the user-defined counter, *name*, is part of a currently active measurement. If it is, MEASCOUNTERBUMPINIT returns the location of the counter value to be bumped in *offset*. If it is not, MEASCOUNTERBUMPINIT returns -1 in *offset* and ERR^UDCNOTPRESENT in *error*.

To define a counter in an application, modify the source code to call the MEASCOUNTERBUMPINIT and MEASCOUNTERBUMP (bumps the counter) procedures at appropriate times. To collect information from a user-defined counter:

1. Use the MEASCOM command ADD USERDEF or the MEASCONFIGURE procedure to add the process that modifies the counter to the configuration.
2. Use the MEASCOM command ADD COUNTER or the MEASCONFIGURE procedure to add the counter to the configuration.

```
error := MEASCOUNTERBUMPINIT ( name            ! i
                             ,offset );        ! o
                             ,[ version ] );   ! I
```

name

input

INT:ref:8

is an array that contains a user-defined counter name. The name must be 1 through 16 alphanumeric characters, hyphens, or underscores, the first of which must be a letter. Counter names are not case-sensitive. They are displayed in uppercase characters.

offset

output

INT:ref:1

is the offset of the counter value within the internal counter record. If the specified counter is not part of an active measurement, MEASCOUNTERBUMPINIT returns *offset* as -1.

version

input

INT:value

(Measure G11 and later) is an incremental version to be reported as SUBSYSTEM-VERSION in the Measure data. HP recommends that version start as 1 and increase any time the USERDEF instrumentation logic in the application is altered. If not specified, SUBSYSTEM-VERSION is reported as 0.

MEAS_DEALLOCATE_TIMERCELLS_

Deallocates timer cells previously allocated by the application. Use this procedure if your application requires fine granularity timers to maintain counters outside of Measure. Measure calls this procedure implicitly for timer-cell counters defined for the USERDEF entity. Measure also calls this procedure implicitly if your program ends without invoking the procedure.

```
error := MEAS_DEALLOCATE_TIMERCELLS_ ( count       ! i
                                     , indexes);   ! i,o
```

count

INT:VALUE

is the number of timer cells to deallocate.

indexes

INT:EXT:REF:*count*

is an array of indexes for timer cells to be deallocated. On output, the array contains the value -1 if the cell was deallocated.

MEASGETVERSION

Returns the Measure version of the specified data file. If the call provides a buffer, the procedure also returns an array of external entity record lengths for the version, indexed by entity type. You can use MEASGETVERSION for two purposes:

- You can use the Measure version of the data file to decide which MEASFH version to use.
- You can use the record length returned to decide the address of the records returned by other procedures (such as MEASREAD).

```
error := MEASGETVERSION ( { dfile }              ! i
                        ,{ dfnum }               ! i
                        ,version                 ! o
                        ,[ buffer ]              ! o
                        ,[ buflen ]              ! i
                        ,[ sysname ]             ! o
                        ,[ release ]             ! o
                        ,[ processor^type ]      ! o
                        ,[ sysidbuf ]            ! o
                        ,[ sysidbuflen ]         ! i
                        ,[ sysidlen ] );         ! o
```

NOTE: Support for *sysidbuf*, *sysidbuflen*, and *sysidlen* begins with Measure product release H05/J03.

dfile

input

INT.EXT:ref:12

is an array containing the data file name. You must specify either *dfile* or *dfnum*. If you do not, error 3201 (ERR^MISSINGPARAM) is returned. Optionally, you can specify both.

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure. You must specify either *dfile* or *dfnum*. If you do not, error 3201 (ERR^MISSINGPARAM) is returned. Optionally, you can specify both.

version

output

INT:ref:1

is the Measure product version when the data file was created. The value returned consists of two parts. Bits 0:7 contain an alphabetic character (for example, D), and bits 8:15 contain a numeric value (for example, 30). Together they express the product version (for example, D30).

buffer

output

INT.EXT:ref:*

is an array of 16-bit integers to be accessed using the entity type index. *buffer* is typically declared as INT.BUFFER[0:MAX^T]. Each array element indicates the legacy style external record length for that entity type. If you pass the *buffer* parameter, you must specify *buflen*, or error 3201 (ERR^MISSINGPARAM) is returned.

buflen

input

INT:value

is an integer indicating the legacy style byte length of the *buffer* array. This value should typically be equal to (MAX^T+1) ¥ 2.

sysname

output

INT.EXT:ref:4

is an 8-byte string that returns the name of the system on which the data file was created.

release

output

INT.EXT:ref:2

is a two-word array containing the Measure version identification:

```
RELEASE[0] = F40
RELEASE[1] = 1
```

processor^type

output

INT.EXT:ref:1

is an integer indicating the CPU type of the system on which Measure is running:

1	TNS II
2	TXP
3	VLX
4	CLX (except CLX 2000)
5	CYCLONE
6	NSR-L
7	NSR-N, NSR-P, and NSR-K
8	NSR-W
9	NSR-G, NSR-T, NSR-V
10	NSE-*n*, where *n* represents a list of Itanium processors (refer to the *HP NonStop System Glossary* for a list of these processors)

Parameters are assumed to be in the calling-process data stack.

sysidbuf

output

STRING.EXT:ref:*

is the buffer where the system serial number is returned.

sysidbuflen

input

INT:value

is an integer indicating the byte length of the *sysidbuf* array.

sysidlen

output

INT:ref:1

is an integer indicating the byte length of the system serial number returned in the *sysidbuf* array.

MEAS_GETDESCINFO_

In Measure H01 and later PVUs, MEAS_GETDESCINFO_ translates a fully qualified, possibly wildcarded, ANSI SQL name to its corresponding entity descriptor components.

MEAS_GETDESCINFO_ provides the MID (PATHID and CRVSN) and UID content for entity descriptor construction.

```
error := MEAS_GETDESCINFO_ ( [dfnum]            ! i
                           , [name]             ! i
```

```
, [nam_len]          ! i
,  pathid            ! o
,  crvsn             ! o
,  uid               ! o
, [index]            ! i,o
, [connectionInfo] );! i
```

error

INT:ref:11

is the error code indicating the outcome of the operation. Possible error codes include:

Error Code	Description
0	Successful completion.
err^badformatsqlname	The specified ANSI SQL name contained syntax errors or was not fully qualified.
err^unknownsqlname	The specified ANSI SQL name could not be translated.
err^sql^api^internal	The specified ANSI SQL name could not be parsed.
err^missing^sqljournal	A SQL/MX journal segment was required and not available.
err^sqlmx^map^process	Either *connectionInfo* was not passed, or *connectionInfo* had incorrect contents, probably because it had not been filled in by a call to MEAS_SQL_MAP_INIT.

dfnum

input

INT:value

is the data file access number, or -1, which is also the default value. To access the journal segment of an open measurement data file containing a journal segment, use the *dfnum* value returned by the MEASOPEN procedure. If -1 is specified, or if no journal segment is available for the specified data file but the file is from the current system, resolution is from the SQL/MX subsystem. If the descriptor is used for an active measurement, -1 should be passed.

name

input

STRING:EXT:ref:*

is a buffer containing the possibly wildcarded, fully qualified ANSI SQL object name or ANSI SQL partition name in external format to be translated. If no name is passed, the output fields will be filled with wildcard indicators.

name_len

input

INT:value

is the size, in bytes, of the input name. Must be present if name is input.

pathid

output

INT:EXT:ref:12

is the *pathid* component of the MID, if a partition name was passed. Otherwise, a wildcard *pathid* value is returned. This corresponds to the File^name^MID.Pathid field in an entity descriptor.

crvsn

output

INT:EXT:ref:3

is the CRVSN component of the MID, if a partition name was passed. Otherwise, a wildcard CRVSN value is returned. This corresponds to the File^name^MID.Crvsn field in an entity descriptor.

uid

output

INT:EXT:ref:12

are the UID components (catalog, schema, object) of the object name. Otherwise, wildcard values are returned for a partition name. This corresponds to the SQLMX^Obj^Desc field in an entity descriptor.

index

input, output

FIXED:EXT:ref:1

is a context value for iterative calls. On the initial call of MEAS_GETDESCINFO_, *index* must be set to -1f. When no more instances of the object can be found, *index* is returned as -1f. If not passed, -1F is assumed.

connectionInfo

input

INT(32) EXT:ref:25

allocated by caller to be at least 100 bytes, this field is filled in by a call to MEAS_SQL_MAP_INIT_. This is only needed when the SQL journal is not being used.

Usage Notes

- When used in conjunction with the SQL journal, each ANSI SQL name can correspond to several instances of the same object.
- If this API is called with a *name_len* of 0, the *pathid*, *crvsn* and *uid* fields are filled with wildcard values.

Examples

This a simplified example in pTAL pseudo code of how to use MEAS_SQLNAME_RESOLVE_ and MEAS_GETDESCINFO_:

```
STRING .EXT defcat[0:MAXCATLEN] :=["cat_12"];
STRING .EXT inbuf[0:MAXANSINAME] :=["table sch.t"];
STRING .EXT namebuf[0:MAXANSINAME];
INT         namelen;
INT         inlen;
FIXED       index;
INT    .EXT dfile^desc(diskfile^ANSI^desc);

! resolve defaults and syntax check
! calculate inlen and catlen
if (error := MEAS_SQLNAME_RESOLVE_ (namebuf,
                                    MAXANSINAME + 1,
                                    namelen,
                                    inbuf,
                                    inlen,
                                    defcat,
                                    catlen ))
  ! handle error

! we have a syntactically correct fully qualified name
! initialize non SQL fields of the descriptor
index := -1f;
do
begin
```

```
  if (error := MEAS_GETDESCINFO_ (dfnum,
                                   namebuf,
                                   namelen,
                                   dfile^desc.File^name^MID.pathid,
                                   dfile^desc.File^name^MID.crvsn,
                                   dfile^desc.SQLMX^Obj^Desc,
                                   index )) then
    ! handle error
  else
    ! use the descriptor, e.g. call measread_diff_
  end;
end
until (index < 0f);
```

MEASINFO

Returns the measurement configuration from a data file.

Unlike the MEASREADCONF procedure, this procedure does not require that the calling process has already opened the file (using MEASOPEN). Because MEASINFO uses less disk space than MEASREADCONF, HP recommends it for tasks that do not require retrieval of actual data records. Use it to help you decide whether a data file contains the needed information before creating MEASFH.

In Measure G09 and later PVUs, MEASINFO has a parameter for retrieving settings that reports the configuration of the journal segment functions.

```
error := MEASINFO ( dfile           ! i
                  ,[ contab ]      ! o
                  ,[ bufsize ]     ! i
                  ,[ bytesret ]    ! o
                  ,[ starttime ]   ! o
                  ,[ stoptime ]    ! o
                  ,[ interval ]    ! o
                  ,[ entities ]    ! o
                  ,[ ctrspace ]    ! o
                  ,[ buflen   ]    ! i
                  ,[ settings] ); ! o
```

dfile

input

INT.EXT:ref:12

is an array containing the data file name in internal (12-word) format.

contab

output

INT.EXT:ref:*

is a buffer that holds the measurement configuration table. For the format of the *contab* array, see the *contab* parameter of MEASCONFIGURE (page 358).

bufsize

input

INT:value

is the size in bytes of the *contab*. buffer. *bufsize* cannot exceed 32,000 bytes, or error 3203 (ERR^BADPARAMS) is returned.

bytesret

output

INT:ref:1

is the byte size of the *contab* array (and the number of bytes returned to the destination buffer, *contab*.) If *bytesret* is larger than *bufsize*, error 3204 (ERR^BUFTOOSMALL) is returned.

starttime

output

FIXED:ref:1

is the start time of the measurement.

stoptime

output

FIXED:ref:1

is the stop time of the measurement. A value of -1 indicates that no stop time was specified.

interval

output

FIXED:ref:1

is the collection interval of the measurement. A value of -1 indicates that no collection interval was specified.

entities

output

INT(32) .EXT:ref:MAX^T+1

is an array that contains the maximum number of entities of each entity type under concurrent measurement. The array elements are in entity-type order, as this table shows. 0 indicates the entity type is not included in the configuration. MEASSTATUS, not MEASINFO, returns the *entities* and *ctrspace* arrays for active measurements. Declare this array with a zero base. For example:

```
INT(32).ENTITIES[0:MAX^T];
or
INT(32).EXT ENTITIES[0:MAX^T];
```

Entity Type	Literal Value	Numeric Identifier
CPU	CPU^T	1
PROCESS	PROCESS^T	2
PROCESSH	PROCESSH^T	3
USERDEF	USERDEF^T	4
FILE	FILOP^T	5
DISCOPEN	DFILOP^T	6
DISC	DISC^T	7
DEVICE	IODEV^T	8
LINE	LINE^T	9
NETLINE	NETLINE^T	10
SYSTEM	REMSYS^T	11
CLUSTER	CLUSTER^T	12
TERMINAL	TERM^T	13
TMF	TMF^T	14
SQLPROC	SQLPROC^T	15

Entity Type	Literal Value	Numeric Identifier
SQLSTMT	SQLSTMT^T	16
OPDISK	OPDISK^T	17
CONTROLLER	CTRL^T	18
SERVERNET	SVNET^T	18
DISKFILE	DISKFILE^T	19
OSSCPU	OSSCPU^T	20
OSSNS	OSSNS^T	21
(Max value)	MAX^T	24

ctrspace

output

INT(32) .EXT:ref:MAX^T+1

is an array that contains the maximum counter space in words used by each entity type. The array elements are in order by numeric identifier. A value of 0 indicates the entity type was not included in the configuration. MEASINFO does not return the *entities* and *ctrspace* arrays for currently active measurements; MEASSTATUS does. Declare this array with a zero base. For example:

```
INT(32).CTRSPACE[0:MAX^T];
or
INT(32).EXT CTRSPACE[0:MAX^T];
```

buflen

input

INT:value

is the length in bytes of *entities*, *ctrspace*, or both. *buflen* should normally be 4 ¥ (MAX^T+1) to specify an *entities* and/or *ctrspace* array of [0:MAX^T] dimension. If specified as a smaller amount, it limits (truncates) the amount of information returned by *entities*, *ctrspace*, or both arrays.

settings

output

INT:ref:1

is an array of flags that identifies configuration attributes of the measurement represented by the data file.

settings.0:8	Reserved for future use.
settings.9	1 = Counter data records are suppressed in this file.
settings.10	Reserved for future use.
settings.11	1 = SQL/MX journal segment is under construction for this file.
settings.12	1 = SQL/MX journal segment is present in this file.
settings.13	Reserved for future use.
settings.14	1 = OSS journal segment is under construction for this file.
settings.15	1 = OSS journal segment is present in this file.

If the SQL journal file is under construction, bits 12 and 13 are both reported as ON.

Usage Notes

- MEASINFO calls ALLOCATESEGMENT to obtain a private extended data segment (segment ID 10) for read traversal of the data file. The extended segment size is set to accommodate two 30 KB (30,000-byte) read buffers and a 32 KB (32,000-byte) temporary buffer for reconstructing records that span the read buffers.

  MEASINFO opens the data file (unstructured, read only, nowait), issues a SETMODE 141 for long transfers, and performs (30KB) READX I/O through the data file to find the information requested.

- The length parameters *bufsize* and *buflen* must be greater than zero bytes and less than 32,001 bytes, or error 3203 (ERR^BADPARAMS) is returned.
- MEASINFO parameters are optional. However:

  dfile

  can be in the process stack segment or in an extended data segment. If omitted, error 3201 (ERR^MISSINGPARAM) is returned.

  contab, bufsize, bytesret

  are a group (that is, one cannot be passed without the other two, or error 3201 (ERR^MISSINGPARAM) is returned). *contab* can be in the process stack segment or in an extended data segment. *bufsize* and *bytesret* must be in the process stack segment.

  starttime, stoptime, interval

  if passed, must be in the process stack segment.

  entities, ctrspace

  can be in the process stack segment or in an extended data segment. If either or both of these parameters are passed, the parameter *buflen* must also be passed, or error 3201 (ERR^MISSINGPARAM) is returned.

  buflen

  must be in the process stack segment. If this parameter is passed, *entities* or *ctrspace* must also be passed, or error 3201 (ERR^MISSINGPARAM) is returned.

  If this parameter is used to limit the array size, the zero base of the array must be taken into account. For example, a *buflen* of 4 (the zero base) does not allow for the return of either *entities* or *ctrspace* information. A *buflen* of 8 allows for the return of *entities* and *ctrspace* information for the CPU entity only. A *buflen* of 12 allows for the return of CPU and process information, and so on.

- If any parameters are declared in an extended data segment, they must be in the same segment, and the segment must be in use prior to calling MEASINFO.
- Parameters typed as extended references (such as *dfile* and *contab*) are expected to be either in the process stack segment or in the same extended data segment. If any passed parameters are part of an extended data segment, they must be in the same segment, and the segment must be in use prior to calling MEASINFO.
- pTAL automatically handles segment management if you declare extended arrays (that is, for *dfile, contab, entities,* and *ctrspace*). However, if you are allocating and managing multiple extended data segments, you must ensure that the correct extended data segment is in use prior to a call to MEASINFO. (Also, any returns by MEASINFO can occur to only one extended data segment.)
- If an OSS journal file is under construction, both bits 14 and 15 of the settings field are reported.

MEASLISTCONFIG

Gets system configuration information from the MEASCTL process in a specified CPU. Each MEASLISTCONFIG call gets configuration information for one type of device associated with one CPU.

```
error := MEASLISTCONFIG ( cpunum          !i
                        ,entity           !i
                        ,buf              !o
                        ,bufsize          !i
                        ,bytesret         !o
                        ,firstcall);      !i,o
```

cpunum

input

INT:value

is the number of a CPU from which you want configuration information.

entity

input

INT:value:1

is the entity type for which you want configuration information. *entity* must be one of:

DISK^T	IODEV^T	LINE^T	NETLINE^T	SERVERNET^T

buf

output

INT.EXT:ref:*

is the buffer in which entity descriptors are returned. MEASLISTCONFIG returns as many descriptors as it can fit in the buffer. Use *bufsize* to set the buffer size.

bufsize

input

INT:value

is the buffer size. *bufsize* should be large enough for the buffer to hold at least one descriptor of the type requested.

If the buffer cannot hold at least one descriptor, error 3204 (ERR^BUFTOOSMALL) is returned.

bytesret

output

INT.EXT:ref:1

is the number of bytes returned in the buffer.

If ERR^BUFTOOSMALL has been returned, *bytesret* returns the minimum buffer length needed for the specified descriptor type.

firstcall

input, output

FIXED.EXT:ref:1

is a context value. The first time you pass *entity* to MEASLISTCONFIG, specify *firstcall* as 0. MEASLISTCONFIG modifies and returns *firstcall*. A *firstcall* value other than -1 indicates that more descriptors are available. Call MEASLISTCONFIG again with the same *entity* value and the returned *firstcall* value.

When no more descriptors are available, MEASLISTCONFIG sets *firstcall* to -1 and returns error 3022 (WARN^NO^MORE^DATA). To retrieve information for a different entity, reinitialize *firstcall* to 0 and specify a new *entity* value.

Usage Note

If MEASLISTCONFIG is called at the same time a ServerNet device is dynamically added or deleted (e.g., with an SCF ADD or SCF DELETE command), the returned configuration data could contain duplicate entries or have missing entries.

MEASLISTENAME

Translates a Guardian file name, a MID or an ANS UID to its corresponding external format ANSI SQL name or OSS pathname.

The specific ANSI SQL name or OSS pathname must be valid (translatable) at the time of the call.

If the CRVSN is specified, only a translation that matches the specified CRVSN returns a name.

```
error := MEASLISTENAME  ( dfnum                ! i
                        , fname-pathid-uid     ! i
                        ,[ sysname ]           ! i
                        ,[ crvsn   ]           ! i
                        ,[ pathid  ]           ! o
                        ,[ extname ]           ! o
                        ,[ extname_max ]       ! i
                        ,[ extname_len ]       ! o
                        ,[ index ]             ! i,o
                        ,[ extname_type ]      ! o
                        ,[ context_crvsn ]     ! o
                        ,[ extname_format ]    ! i
                        ,[ input_type ]        ! i
                        ,[ connectionInfo ]    ! i
```

error

INT

is the error code indicating the outcome of the operation. Possible error codes include:

Error Code	Description
0	Successful completion.
err^unknownsqlname	The specified ANSI SQL name could not be translated.
err^crvsnnotspecified	A CRVSN was required but not specified.
err^missing^sqljournal	An SQL journal segment was required but not available.
err^sqlmx^map^process	Either connectionInfo was not passed, or connectionInfo had incorrect contents, probably because it had not been filled in by a call to MEAS_SQL_MAP_INIT.

dfnum

input

INT:value

is the data file access number, or -1. To access the journal segment of an open Measure data file that contains a journal segment, use the *dfnum* value returned by the MEASOPEN procedure. If the data file access number is omitted, -1 is specified. If no journal segment is available for the specified data file and the file is from the current system, *dfnum* is retrieved from the SQL/MX or OSS subsystem.

fname_pathid_uid

input

INT:EXT:ref:12

is an array of 12 words that can contain the local internal format Guardian name of the SQL/MX partition or OSS file, or is an internal format *pathid* of the SQL/MX partition or OSS file, or is an ANS UID as it can be found in the SQLMX^Obj^Desc structure of Diskfile^ANSI^desc and file^open^ANSI^desc. For the last case, the *input_type* must be set to MEAS_ANSUID, as shown under *input_type*, below.

sysname

input

INT:EXT:ref:4

is an array of four words that can contain the Expand system name of the file specified in *filename*, if the file is remote.

crvsn

input

INT:EXT:ref:3

is an array of three words that can contain a CRVSN value for qualifying the specified *pathid* or *filename*.

pathid

output

INT:EXT:ref:12

is the internal format *pathid* associated with the file.

extname

output

STRING.EXT:ref:*

is a buffer to which the external name is to be returned, null-terminated.

extname_max

input

INT:value

is the maximum length, in bytes, that the *extname* buffer can accommodate. *extname_max* is required when *extname* is specified.

extname_len

output

INT:ref:1

is the length, in bytes, of the name returned in *extname*, including the null terminator. *extname_len* is required when *extname* is specified.

index

input, output

FIXED:ref:1

is a context value for iterative calls. On the initial call of MEASLISTENAME, index must be set to -1f. If the translation resolves to a single name, index is returned as -1f.

If the translation resolves to multiple names, index contains an internal value that must be passed to subsequent MEASLISTENAME calls. If index is returned as -1f, no more data is available. To explicitly stop the iteration of the MEASLISTENAME call sequence before encountering a -1f termination value, make a final call using a negated index (index := -index).

extname_type

output

INT:ref:1

indicates whether the name returned in *extname* is to be treated as an OSS pathname or as an ANSI SQL name:

1	OSS pathname
2	ANSI SQL name

context_crvsn

output

INT:ref:1

is an array of three words containing the CRVSN value of the instance of the Guardian file name corresponding to the external name returned in the *extname* buffer. This value is used to distinguish between multiple instances of the same Guardian file name during the measurement period.

extname_format

input

INT:ref:1

specifies the format of the external name returned in *extname*:

For OSS pathnames:	
0	OSS pathname format
1	OSS pathname format

For ANSI SQL names:	
0	SQL/MX partition name format
1	ANSI SQL object name format

input_type

input

INT:ref:1

specifies the type of the data in the *fname_pathid_uid* field. *input_type* is a required parameter if the data is an ANS UID, else it is optional.

For ANSI SQL names:	
0	File name data type
1	Path ID data type
2	ANS UID data type

connectionInfo

input

EXT:ref:25

is allocated by the caller to be at least 100 bytes. *connectionInfo* must have been filled in by a call to MEAS_SQL_MAP_INIT_. *connectionInfo* is only needed when the SQL journal is not being used.

Usage Notes

- If a *pathid* is passed in *fname_pathid_uid*, *crvsn* must also be provided.
- If an ANS UID is passed in *fname_pathid_uid*, the *input_type* parameter must be specified to be MEAS_ANSUID.
- If you want *extname* and *extname_len* as output, specify *extname_max*.
- If the caller of MEASLISTENAME does not include the *extname_format* parameter, zero is assumed for the parameter.
- Never change a returned index to another positive value. It can result in a never ending loop of calls to MEASLISTENAME.

Example

This a simplified example in pTAL pseudo code of how to call MEASLISTENAME with an ANS UID:

```
! get the ANSI SQL name for a specified UID
uid := desc.AnsiUIDs;

if ( error := measlistename( dfnum,
                             uid,
                             !sysname!,
                             !crvsn!,
                             !pathid!,
                             ansi_name,
                             ansi_name_max,
                             ansi_name_len,
                             !index!,
                             !extname_type!,
                             !context_crvsn!,
                             !extname_format!,
                             MEAS_ANSUID )) then
   return error
else
   ! use the ansi_name
```

MEASLISTEXTNAMES

Makes the Measure subsystem list structured OSS and ANSI SQL name information to the EXTNAMES file in the specified subvolume.

```
error := MEASLISTEXTNAMES  ( dfnum              ! i
                           , volume^subvol      ! i
```

error

INT

is the error code indicating the outcome of the operation. Possible error codes include:

Error Code	Description
0	Successful completion.
3236 err^missingossjournal	An OSS journal segment was required but not available.

dfnum

input

INT:value

is the data file access number returned by the MEASOPEN procedure.

volume^subvol

input

INT:EXT:ref:8

is the destination volume and subvolume for the EXTNAMES file.

For a local system, the format of *volume^subvol* (when redefined as *string*[0:15]) is:

volume^subvol[0]	'$'
volume^subvol[1:7]	volume name (padded with spaces)
volume^subvol[8:15]	subvolume name (padded with spaces)

For a remote system, the format is:

volume^subvol[0]	'\'
volume^subvol[1]	system number
volume^subvol[2:7]	volume name (padded with spaces)
volume^subvol[8:15]	subvolume name (padded with spaces)

Usage Notes

- Before you issue this call, you must have a measurement data file containing the SQL journal segment open with the SQL journal segment attached, and/or an OSS journal segment open with the OSS journal segment attached. If the measurement data file is for an active measurement, or if MEASFH is still constructing the SQL or OSS journal segment, err^buildingsqljournal or err^buildingossjournal is returned from MEASOPEN(). You must use the MEASOPEN() option parameter to open and "attach" a journal segment. For more information, refer to the MEASOPEN (page 421) Usage Notes.
- If the EXTNAMES file already exists in the specified *volume^subvol*, new entries are written to the existing file, and all instances of error 10 (duplicate record) are ignored. A single EXTNAMES file can contain information for many measurements from many systems.

MEASLISTGNAME

Translates an OSS file pathname or ANSI SQL object name to its Guardian file-name equivalent. MEASLISTGNAME provides the MID (PATHID and CRVSN) content for entity descriptor construction. The specified OSS file pathname or ANSI SQL name must be valid at the time of the call. To get OSS directory entries or partitions of an ANSI SQL object, call MEASLISTGNAME iteratively.

If the specified OSS file pathname refers to an OSS directory, the file name is returned blank (space-filled), but the PATHID and CRVSN are valid and properly identify the directory in internal format for use in MID structures.

```
error := MEASLISTGNAME  ( dfnum                  ! i
                        ,name                    ! i
                        ,name_len                ! i
                        ,[ filename ]            ! o
                        ,[ sysname ]             ! o
                        ,[ pathid  ]             ! o
                        ,[ crvsn   ]             ! o
                        ,[ extname]              ! o
                        ,[ extname_max]          ! i
                        ,[ extname_len]          ! o
                        ,[ index   ]             ! i,o
```

```
                              ,[ name_type ]          ! i
                              ,[ connectionInfo ] );! i
```

error

INT

is the error code indicating the outcome of the operation. Possible error codes include:

Error Code	Description
0	Successful completion.
3233 err^invalidosspath	The specified pathname could not be resolved.
3236 err^missingossjournal	An OSS journal segment was required but not available.
3239 err^badformatsqlname	The specified ANSI SQL name contained syntax errors or was not fully qualified
3238 err^unkownsqlname	The specified ANSI SQL name could not be translated.
3241 err^missing^sqljournal	A SQL/MX journal segment was required but not available
3295 err^sql^api^internal	The ANSI SQL name specified could not be parsed.
3296 err^sqlmx^map^process	Either *connectionInfo* was not passed, or *connectionInfo* had incorrect contents, probably because it had not been filled in by a call to MEAS_SQL_MAP_INIT.

dfnum

input

INT:value

is the data file access number, or -1. To access the journal segment of an open Measure data file that contains a journal segment, use the *dfnum* value returned by the MEASOPEN procedure. If the data file access number is omitted, -1 is specified. If no journal segment is available for the specified data file and the file is from the current system, *dfnum* is retrieved from the SQL/MX or OSS subsystem.

name

input

STRING.EXT:ref:*

is a buffer containing the OSS file pathname or ANSI SQL object name to be translated. If passed, an ANSI SQL object name must be fully qualified and not contain wildcards.

name_len

input

INT.value

is the length, in bytes, including a terminating null byte if it is an OSS path name.

filename

output

INT:EXT:ref:12

is an array of 12 words. *filename* returns the internal format local form Guardian name of the referenced file. If the specified OSS file pathname resolves to an OSS directory, the *filename* returned is space-filled. If the OSS file pathname resolves to a remote file name, a local form name is returned in *filename*, and the remote system name is returned in *sysname*.

sysname

output

INT:EXT:ref:4

is an array of four words. If the file is remote, *sysname* returns the Expand system name of the file contained in *filename*. If the file is local, *sysname* returns spaces.

pathid

output

INT:EXT:ref:12

is an array of 12 words. *pathid* returns the OSS internal *pathid* describing the specified OSS file pathname. The *pathid* value returned can be used in MID structures of entity descriptor templates.

pathid is not used for ANSI SQL names.

crvsn

output

INT:EXT:ref:3

is an array of three words. *crvsn* is the creation version serial number associated with the returned *pathid*.

extname

output

STRING.EXT:ref:*

for iterative calls, a buffer in which OSS file pathnames for a particular directory's contents are returned or in which the ANSI SQL partition names for a particular ANSI SQL object are returned.

extname_max

input

INT:value

the maximum length, in bytes, that the *extname* buffer can hold. Required when *extname* is specified.

extname_len

input

INT:EXT:ref:12

for iterative calls, the length, in bytes, of the OSS file pathname or ANSI SQL partition name returned in *extname*. The length includes the null terminator.

index

input, output

FIXED:ref:1

is a context value for iterative calls. On the initial call of MEASLISTGNAME, *index* must be set to -1f. If the translation resolves to an OSS file or an ANSI SQL partition name, *index* is returned as -1f.

If the translation resolves to an OSS directory or an ANSI SQL object name, *index* contains an internal value that must be passed to subsequent MEASLISTGNAME calls. If *index* is returned as -1f, no more data is available about the OSS directory or ANSI SQL object. To explicitly stop the iteration of the MEASLISTGNAME call sequence for OSS names before encountering a -1f termination value, make a final call using a negated index (index := -index).

name_type

input

INT:value

indicates whether the name passed in *name* is to be treated as an OSS pathname or as an ANSI SQL name. If omitted, *name* is treated as an OSS pathname.

1	OSS pathname
2	ANSI SQL name

connectionInfo

input

INT(32) EXT:ref:25

is allocated by the caller to be at least 100 bytes. *connectionInfo* must have been filled in by a call to MEAS_SQL_MAP_INIT_. *connectionInfo* is only needed when the SQL journal is not being used.

Usage Notes

- If iterative calls are initiated, the program should continue calling MEASLISTGNAME until *index* is returned as -1f. Otherwise iterative processing should be terminated by a final call with *index* negated (index := -index). If the iterative processing is interrupted, the calling program retains an open of the OSS file-system directory. Repeated behavior of this type eventually results in the program reaching the limit of concurrent opens for a process.
- Never change a returned index to another positive value. It can result in a never ending loop of calls to MEASLISTGNAME.

Example

These are simplified examples in pTAL pseudo code of how LISTGNAME could handle ANSI SQL names:

Example of iterative calls:

```
index := -1f;
do
begin
  if (error := measlistgname(dfnum,
                             ansi_name,
                             ansi_name_len,
                             guardianname,
                             sys,
                             !pathid!,
                             !crvsn!,
                             !partname!,
                             !partname_max!,
                             !partname_len!,
                             index,
                             MEAS_ANSI_SQL_NAME )) then
    ! handle error
  else if guardianname <> " " then
  begin
    ! display guardian filename
    if partname_len then
      ! display the partition name
  end
  else
    ! display the TABLE or INDEX name
end
until (index < 0f);
```

Example of non-iterative calls:

```
! get the Guardian file name and crvsn for an ANSI SQL name
! with partition
if ( error := measlistgname(-1,
                            ansi_names,
                            ansi_name_len,
                            guardianname,
                            !sysname!,
                            !pathid!,
                            crvsn,
                            !extname!,
                            !extname_max!,
                            !extname_len!,
                            !index!,
                            MEAS_ANSI_SQL_NAME,
                            connectionInfo)) then
   return error;
```

MEASLISTOSSNAMES

Causes the Measure subsystem to list structured OSS file pathname translation information to the file OSSNAMES in the default subvolume.

In Measure G11 and later PVUs, MEASLISTOSSNAMES is still available but is superseded by MEASLISTEXTNAMES (page 411), which supports both OSS file pathnames and ANSI SQL names.

```
error := MEASLISTOSSNAMES ( dfnum              ! i
                          ,volume^subvol);     ! i
```

error

INT

is the error code that indicates the outcome of the operation. Zero means a successful completion. Error codes include:

```
3235 err^buildingossjournal - the OSS journal segment for
this file is still under construction.

3236 err^missingossjournal - an OSS journal segment was
required and not available.
```

dfnum

input

INT:value

is the data file access number returned by the MEASOPEN procedure.

volume^subvol

input

INT .ext:ref:8

is the destination volume and subvolume for the OSSNAMES file.

```
volume^subvol[0]         = '\'
volume^subvol[1]         = system number
volume^subvol[2:7]       = volume name (padded with spaces)
volume^subvol[8:15]      = subvolume name (padded with spaces)
```

Usage Notes

- You must have a measurement data file that contains an OSS journal segment open with the OSS journal segment attached prior to issuing this call. If the measurement data file is

for an active measurement or if MEASFH is still in the process of constructing the OSS journal segment, an error 3235 is returned.

- If an OSSNAMES file already exists in the specified *volume^subvol* field, the new OSSNAMES entries are written to the existing file, and all completions of error 10 (duplicate record) are ignored. A single OSSNAMES file can contain OSSNAMES information for many measurements from many systems.

MEASLISTPNAME

Translates a Guardian file name or an OSS pathid to its OSS file pathname equivalent. The desired OSS file pathname must be valid at the time of the call. If CRVSN is specified, only a translation that matches the specified CRVSN returns a pathname.

In Measure G11 and later PVUs, MEASLISTPNAME is still available but is superseded by MEASLISTENAME (page 408), which supports both OSS file pathnames and ANSI SQL names.

```
error := MEASLISTPNAME  ( dfnum                ! i
                        ,fnameorpathid         ! i
                        ,[ sysname ]           ! i
                        ,[ crvsn   ]           ! i
                        ,[ pathid  ]           ! o
                        ,[ pathname]           ! o
                        ,[ pathname_max]       ! i
                        ,[ pathname_len]       ! o
                        ,[ index  ]);          ! i,o
```

error

INT

is the error code indicating the outcome of the operation. Zero means a successful completion. Other possible error codes include:

```
3204 err^buftoosmall - the pathname returned was larger than
the pathname^max bytes.

3233 err^invalidosspath - the fnameorpathid specified could
not be resolved.

3234 err^crvsnnotspecified - the gname specified was
translated, but may not be correct.

3236 err^missingossjournal - an OSS journal segment was
required and not available.
```

dfnum

input

INT:value

is the data file access number or -1. To access the journal segment of an open Measure data file that contains a journal segment, use the *dfnum* value returned by the MEASOPEN procedure. If the data file access number is omitted, -1 is specified. If no journal segment is available for the specified data file, but the file is from the current system, *dfnum* is retrieved from the OSS file system.

fnameorpathid

input

INT.EXT:ref:12

is an array of 12 words that can contain the internal format of a local form Guardian name of an OSS file or an internal format of an OSS pathid.

sysname

input

INT .EXT:ref:4

is an array of four words. If the file is remote, the array should contain the EXPAND system name of the file specified in *filename*.

crvsn

input

INT .EXT:ref:3

is an array of three words that can contain a CRVSN value for qualifying the pathid or file name specified.

pathid

output

INT.EXT:ref:12

is the internal format pathid associated with the returned OSS file pathname.

pathname

output

STRING.EXT:ref:*

is a buffer where the OSS file pathname is returned.

pathname_max

input

INT:value

is the maximum length in bytes that the pathname buffer will hold.

pathname_len

output

INT:ref:1

is the length in bytes of the OSS file pathname returned in *pathname*.

index

input, output

FIXED:ref:1

is a context value for iterative calls. On the first call, the *index* is initialized to -1f. If no other OSS file pathnames exist for the requested translation, *index* is returned as a -1f. Otherwise, *index* contains an internal value that can be passed to subsequent MEASLISTPNAME() calls for iterative processing.

Usage Notes

- If a PATHID is passed in *fnameorpathid*, the CRVSN must also be provided.
- If a PATHID and *pathname_len* are desired as output, *pathname_max* must be specified.

MEASMONCONTROL

Starts or stops the Measure subsystem. Alternatively, adds a CPU to the Measure subsystem by starting a measurement control process (MEASCTL) in the specified CPU.

NOTE: The calling process must have a super-group user (255,*) process accessor ID to invoke this procedure.

```
error := MEASMONCONTROL ( meascb          ! i,o
                        ,start            ! i
                        ,[ cpunum ] );    ! i
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. Once you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

start

input

INT:value

If specified as true, MEASMONCONTROL starts the Measure subsystem. If specified as false, MEASMONCONTROL stops the Measure subsystem. When the Measure subsystem stops, all measurements stop.

cpunum

input

INT:value

creates a measurement control process (MEASCTL) in the specified CPU. Any currently active measurements automatically begin receiving information from the newly added CPU. This parameter is applicable only if *start* equals true. Under normal operating circumstances, this parameter should not have to be used. The Measure subsystem restarts a MEASCTL automatically in a reloaded CPU.

MEASMONSTATUS

Returns the number of currently active (or configured) measurements and their measurement numbers and data file names. The optional parameter *settings* returns a bit mask that identifies the current settings of several Measure subsystems.

In Measure G11 and later PVUs, the MEASMONSTATUS settings parameter has a new bit (settings.<8>) to indicate the SQL/MX journal status.

```
error := MEASMONSTATUS ( meascb          ! i,o
                       ,measurements     ! o
                       ,measnames        ! o
                       ,[settings ] );   ! o
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. After you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

measurements

output

INT:ref:1

is the number of currently active measurements.

measnames

output

INT.EXT:ref:*

is an array of measurement data file names in 12-word format. The array index of a measurement data file name is its measurement number. The array size is 64 files (64 ¥ 12 words).

settings

output

INT:ref:1

is an array of flags that indicates the current setting of Measure subsystem parameters controlled through defines:

settings.0:7	Reserved for future use.
settings.8	SQL/MX journal segment default ON/OFF.
settings.9	OSS journal segment default ON/OFF.
settings.10	Reserved for internal use.
settings.11	Skew interval copy time is ON/OFF.
settings.12	Lock CID table ON/OFF.
settings.13:15	Maximum CIDs setting: 0 = 32,000 1 = 64,000 2 = 96,000 3 = 128,000 4 = 192,000 5 = 256,000 6 = 512,000 7 = reserved for future use

Usage Notes

CID stands for counter identifier. For each counter record in a processor, Measure allocates and tracks the counter record through a CID. In G-series and earlier RVUs, the CID limit can be specified at START MEASSUBSYS time to be 32,000, 64,000, 96,000, or 128,000 for each processor. In H-series and J-series RVUs, CID sizes of 196,000 and 256,000 are also supported. In Measure H04 and J02 PVUs, a maximum of 512,000 is supported. The CIDs value corresponds to the CID table size. The current number of CIDs in use is reported in the CPU entity report of each processor.

The number of CIDs supported in a processor is limited. In Measure product versions earlier than G05, the CID table has an upper limit of 32,000 CIDs per processor. In product versions D45, G05, and later, you can increase the CID table limit in increments of 32,000 CIDs. In H-series and J-series RVUs, the default size of the CID table is 64,000 CIDs per processor.

MEASOPEN

Obtains read, write, or read and write access to a measurement data file; opens the file: and creates a Measure file-handling (MEASFH) process to access the file. Use the *write* and *read* parameters to specify access type. To close a data file, use the MEASCLOSE procedure.

The best way to access a remote data file is to use the *measfh* parameter. You specify an object file (on the remote system) that contains a Measure file-handling (MEASFH) program. Measure then starts a process on the remote system that executes the MEASFH program to read the data file. MEASOPEN has a parameter for specifying options. Two bits of this field override the default treatment of the OSS journal segment function.

```
error := MEASOPEN ( dfile            ! i
                  ,dfnum             ! o
                  ,write             ! i
                  ,read              ! i
                  ,[measfh ]         ! i
                  ,[swapvol]         ! i
                  ,[cpunum]          ! i
                  ,[options] )       ! i
                  ,[filesize ] ) ;   ! i
```

error

INT

is the error code indicating the outcome of the operation. Possible error codes include:

Error Code	Description
0	Successful completion.
err^buildingsqljournal	The SQL journal segment for this file is still under construction.

dfile

input

INT:ref:12

is an array containing the data file name.

dfnum

output

INT:ref:1

is the data file access number. Use this number in subsequent procedure calls to identify the data file. The Measure subsystem uses the file number of the Measure file-handling (MEASFH) process open as this unique file-access number.

write

input

INT:value

is used with *read* to specify access type:

Write	Read	Result
True	True or False	Obtains write access to the data file and automatically initializes it. During initialization, any data in the file is deleted. It must be an unstructured local disk file with a file code of 175 or a local tape file. If the file does not exist, it is created.
False	True	Obtains read access to the data file. The data file must be a local or remote disk file.

read

input

INT:value

is used with *write* to specify access type. For possible values and results, see *write*.

measfh

input

INT:ref:12

is an array containing the name of an object file for a Measure file-handling (MEASFH) program. By default, MEASFH object files are named $SYSTEM.SYS*nn*.MEASFH.

swapvol

input

INT:ref:4

is an array containing the volume name for MEASFH swap files.

If *swapvol* is not specified, swap files are created on the swap volume of the calling process.

cpunum

input

INT:value

is the number of the CPU where MEASFH is created.

options

input

INT:value

is an array of flags qualifying the MEASOPEN request:

options.0:8		Reserved for future use.
options.9		Override the default mode for writing counter data records to the measurement data file.
	0	Counter data records will be included (default).
	1	Counter data records will be suppressed.
options.10:11		Reserved for future use.
options.12:13		Override the default mode for SQL/MX journal segment handling.
	0	SQL/MX not specified; use the Measure subsystem default setting to determine journal segment handling.
	1	SQL/MX ON; include the SQL/MX journal segment function.
	2	SQL/MX OFF; do not include the SQL/MX journal segment function.
	3	Invalid value; err^badparams is returned.
options.14:15		Override the default mode for OSS journal segment handling.
	0	OSS not specified; use the Measure subsystem default setting to determine journal segment handling.
	1	OSS ON; include the OSS journal segment function.
	2	OSS OFF; do not include the OSS journal segment function.
	3	Invalid value; err^badparams is returned.

filesize

input

FIXED:value

is the desired file size (capacity) in bytes. The minimum value for *filesize* is 133169152 (127 MB) and the maximum is 1099507433472 (1048572 MB) or the maximum disk size (if less than 1048572 MB). The default value is 1073741824 (1024 MB). A value of -1 indicates that no file size was specified. If *filesize* is less than the minimum allowed value or greater than the maximum allowed value, error 3203 (ERR^BADPARAMS) is returned. If the file already exists, the *filesize* parameter is ignored, and the capacity of the existing file is used.

Usage Notes

- A call to MEASOPEN with read access is equivalent to executing an ADD MEASUREMENT command. The MEASOPEN procedure creates a MEASFH process, opens it, and sends the name of the data file to add to the MEASFH process. The MEASFH process scans the data file and builds a set of in-memory table indexes for future access requests (the same as is done in an ADD MEASUREMENT operation). You should know the disk and memory resource costs associated with this procedure.
- A call to MEASOPEN with write access results in a purge of the data file contents. The purge of the data is performed by the MEASFH process.
- These LITERALs are defined in MEASDECS and MEASCHMA:

```
LITERAL OSSNOTSPECIFIED = 0;
LITERAL OSSFORCEDON = 1;
LITERAL OSSFORCEOFF = 2;

LITERAL  SQLNOTSPECIFIED = 0;
LITERAL  SQLFORCEDON = 1;
LITERAL  SQLFORCEDOFF = 2;
```

- If write access is specified, the *options* parameter determines whether the OSS or SQL/MX journal segment is constructed with the measurement. If read access is specified, the *options* parameter determines whether an OSS or SQL/MX journal segment in a Measure data file is attached to the calling process and whether error or warning messages about the OSS or SQL/MX journal segment function are passed to the calling process.
- If the caller of MEASOPEN does not include the *options* parameter, OSSFORCEDOFF or SQLFORCEDOFF is assumed. If write access is specified, no journal segment construction occurs. If read access is specified, no attempt is made to attach an OSS or SQL/MX journal segment from the data file. This handling of the *options* parameter prevents inadvertent performance degradation on the system by applications that cannot utilize the OSS or the SQL/MX journal segment function.
- Programs that start measurements programmatically (for example, programs that usually involve system monitoring, system management, and OSS pathname translation services) are encouraged to use the OSS name server facilities on the system rather than the journal segment function.
- These literals are defined in MEASDECS and MEASCHMA:

LITERAL	SQLNOTSPECIFIED = 0
LITERAL	SQLFORCEDON = 1
LITERAL	SQLFORCEDOFF = 2

- If write access is specified, *options* determines whether to construct the SQL journal segment with the measurement.

- If read access is specified, *options* determines whether the SQL journal segment in a measurement data file is attached to the calling process and whether error or warning messages regarding the SQL journal segment function are passed to the calling process.
- If the caller of MEASOPEN does not include *options*, SQLFORCEDOFF is the default. If write access is specified, no journal segment construction occurs. If read access is specified, no attempt is made to attach the SQL journal segment from the data file. This prevents inadvertent system performance impact by applications that cannot utilize the SQL journal segment function.
- In Measure G11 and later PVUs, the MEASOPEN *options* parameter has two new bits (options.<12:13>) used to override the default treatment of the SQL/MX Journal Segment function, and MEASOPEN returns a new error code if the SQL/MX Journal Segment is under construction.
- In Measure H02 and later PVUs, the MEASOPEN callable procedure allows the caller to select the measurement data file size, suppress counter data records in the measurement data file, or both.

MEASREAD

Reads one or more counter records from a measurement data file. The maximum number of records returned in a single call depends on the size of the destination buffer, *loc*. (Only complete records are returned.) The data file can be associated with a currently active measurement, or it can contain data from an inactive measurement. Before reading from a measurement data file, you must obtain read access to the file using the MEASOPEN procedure.

In Measure H01 and later PVUs, MEASREAD accepts DISCOPEN, DISKFILE, FILE and SQLSTMT entity descriptors for ANSI SQL objects or partitions.

```
error := MEASREAD ( dfnum                    ! i
                  ,entitydesc                ! i
                  ,loc                       ! o
                  ,bufsize                   ! i
                  ,bytesret                  ! o
                  ,firstcall                 ! i,o
                  ,[ nomtime ]               ! i
                  ,[ timetol ]               ! i
                  ,[ version ]               ! i
                  ,[ templateversion ] );    ! i
```

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

entitydesc

input

INT:ref:*

is the entity type and entity specification of the desired counter records. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. If you specify *entitydesc* as a template for multiple specifications, MEASREAD returns all counter records that fit the template.

For ANSI SQL objects or partitions, *entitydesc* can be a DISCOPEN, DISKFILE, FILE, or SQLSTMT entity descriptor.

loc

output

INT.EXT:ref:*

is the destination buffer for the counter records. Counter records are written to the buffer in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

bufsize

input

INT:value

is the size in bytes of the destination buffer, *loc*.

bytesret

output

INT:ref:1

is the number of bytes returned to the destination buffer, *loc*.

firstcall

input, output

FIXED:ref:1

is a context value. The first time you pass *entitydesc* to MEASREAD, specify *firstcall* as 0. MEASREAD modifies and returns *firstcall*. As long as MEASREAD returns a nonzero value in *firstcall*, more counter records are available. Call MEASREAD again with the same *entitydesc* value and the *firstcall* value that was returned to you.

nomtime

input

FIXED:value

is a nominal time. If you omit *nomtime* (or specify it as -1), MEASREAD returns all the counter records written to the file. If you specify *nomtime* and omit *timetol* (or specify it as -1), MEASREAD returns the counter records written from the start of the measurement to the earliest record later than *nomtime*. If you specify both *nomtime* and *timetol*, MEASREAD returns the counter records written from the start of the measurement to *nomtime* plus *timetol* time. The *nomtime* default is -1.

timetol

input

FIXED:value

is a tolerance value. For more information, see *nomtime*. The *timetol* default is -1.

version

input

INT:value

is the Measure product version when the data file was created. The value consists of two parts: bits 0:7 contain an alphabetic character (for example, D), and bits 8:15 contain a numeric value (for example, 30). Together they express the product version (for example, D30).

templateversion

input

FIXED:value

(Measure G11 and later) is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc*, if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

MEASREAD_DIFF_

This procedure is an enhanced version of MEASREAD that reads a window of one or more counter records from a measurement data file.

Before reading from a measurement data file, you must obtain read access to the file using the MEASOPEN procedure. The data file can be associated with a currently active measurement, or it can contain data from an inactive measurement.

Unlike MEASREAD, the caller specifies both start time and stop time. The records returned provide the statistics collected within the specified window.

To get all records for the specified time window might require repeated calls. Only complete records are returned in a given call. The number returned in a given call depends on the size of *loc* (the destination buffer).

In Measure G09 and later PVUs, the parameter if^item has a field template if^item^g09^def that is available for the longer (128 byte) SQL/MX run unit names. The new template coexists with the pre-G09 if^item^def template, thereby preserving compatibility with existing applications and older data files.

```
error := MEASREAD_DIFF_ ( dfnum                    ! i
                        ,entitydesc                ! i
                        ,loc                       ! o
                        ,bufsize                   ! i
                        ,bytesret                  ! o
                        ,firstcall                 ! i,o
                        ,from^time                 ! i
                        ,to^time                   ! i
                        ,[ timetol ]               ! i
                        ,[ version ]               ! i
                        ,[ all^recs ]              ! i
                        ,[ zero^reports ]          ! i
                        ,[ totals ]                ! i
                        ,[ loadid ]                ! i
                        ,[ if^item ]               ! i
                        ,[ totals^num ]            ! o
                        ,[ templateversion ] );    ! i
```

NOTE: The maximum number of USERDEF records in the MEASREAD_DIFF_ reply buffer is 222. If this maximum is exceeded, the error 3204 (ERR^BUFTOOSMALL) is displayed.

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

entitydesc

input

INT.EXT:ref:*

is the entity type and entity specification of the desired counter records. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. If you specify *entitydesc* as a template for multiple specifications, MEASREAD_DIFF_ returns all counter records that fit the template.

For ANSI SQL objects or partitions, *entitydesc* can be a DISCOPEN, DISKFILE, FILE, or SQLSTMT entity descriptor. The SQLSTMT ANSI entity descriptor is used when an ANSI SQL name is to be specified for the retrieval of SQLSTMT counter records.

loc

output

INT.EXT:ref:*

is the destination buffer for the counter records. Counter records are written to the buffer in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

bufsize

input

INT:value

is the size in bytes of the destination buffer, *loc*. If *bufsize* is greater than 32,000, error 3203 (ERR^BADPARAMS) is returned.

bytesret

output

INT.EXT:ref:1

is the number of bytes returned to the destination buffer, *loc*.

firstcall

input, output

FIXED.EXT:ref:1

is a context value. The first time you pass *entitydesc* to MEASREAD_DIFF_, specify *firstcall* as 0. MEASREAD_DIFF_ modifies and returns *firstcall*. As long as MEASREAD_DIFF_ returns a nonzero value in *firstcall*, more counter records are available. Call MEASREAD_DIFF_ again with the same *entitydesc* value and the returned *firstcall* value.

from^time

input

FIXED:value

is the start time of the measurement window. Specify *from^time* as a Julian date based on local civil time in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP. (For a description of the JULIANTIMESTAMP, see the *Guardian Procedure Calls Reference Manual*.) A value of -1 signifies the beginning of the measurement.

to^time

input

FIXED:value

is the stop time of the measurement window. Specify *to^time* as a Julian date based on local civil time in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP. (For a description of the JULIANTIMESTAMP, see the *Guardian Procedure Calls Reference Manual*.) A value of -1 signifies the end of the measurement.

timetol

input

FIXED:value

is a tolerance value to be applied to the time window bounded by *from^time* and *to^time*, effectively decreasing *from^time* by *timetol* and increasing *to^time* by the same amount. A value of -1 widens the time window to include the latest record earlier than *from^time* and the earliest record later than *to^time*.

version

input

INT:value

is the Measure product version when the data file was created. The value returned consists of two parts: bits 0:7 contain an alphabetic character (for example, D), and bits 8:15 contain a numeric value (for example, 30). Together they express the product version (for example, D30).

all^recs

input

INT:value

requests all interval records within the specified window if the value 1 is passed. If omitted or 0, only summary records are returned.

zero^reports

input

INT:value

requests the exclusion of zero records if the value 0 is passed. If omitted or 1, zero records are returned.

totals

input

INT:value

is a value of 0, 1, or 2, which indicates:

0	Omit the total record; include individual counter records (default).
1	Include individual records and the total record.
2	Omit individual counter records; include the total record.

loadid

input

INT.EXT:ref:4

is an 8-byte string to be included in all records returned.

if^item

input

INT.EXT:ref:$LEN(IF^ITEM^DEF) / 2

is a structure (IF^ITEM^DEF, defined in the MEASDECS file):

```
STRUCT IF^ITEM^DEF (*);
Begin
Int      Item^Name[0:15];  !Item name in entity record
Int      Relation^Operator;!Relation Operator
                           !  1 -> Equal        =
                           !  2 -> Not Equal    <>
                           !  3 -> Less Than    <
                           !  4 -> Greater Than >
Int(32)  Value;            !Value used in comparison
Int      Rate;             !Use rated or nonrated value
                           !for condition checking:
                           !  0 -> No-Rate, 1 -> Rated
Int      SQLName[0:15];    !SQL run-unit in Ascii
Int      SQLIndex;         !SQL Index number
End;
```

or

if^item

input

INT .EXT:ref:$LEN(IF^ITEM^G09^DEF) / 2

is a structure (IF^ITEM^G09^DEF, defined in the MEASDECS file):

```
STRUCT IF^ITEM^G09^DEF (*);
Begin
Int     Item^Name[0:15];  !Item name in entity record
Int     Relation^Operator;!Relation Operator
                          !See Usage Notes (page 430)
                          !Bit 0 must be 1
                          !
                          !Bits 13:15
                          !  1 -> Equal        =
                          !  2 -> Not Equal   <>
                          !  3 -> Less Than    <
                          !  4 -> Greater Than >
Fixed Value;              !Value used in comparison
Int     Rate;             !Use rated or nonrated value
                          !for condition checking:
                          !  0 -> No-Rate, 1 -> Rated
Int     SQLName[0:63];    !for SQLSTMTs only
Int     SQLIndex;         !SQL Index number
String  IP^addr[0:15] = SQLName;
Int     port = SQLIndex;
Int     IP^wildcard^flags = value;
End;

DEFINE  IP^addr^wildcard^flag = IP^wildcard^flags.<0>#,
        port^wildcard^flag = IP^wildcard^flags.<1>#;
```

item^name

input

is one of the counter names defined in the entity record (for example, CPU-BUSY-TIME for a CPU record).

relation^operator

input

is a value in the range 1 through 4 that designates the comparison type:

1	Compare for equal
2	Compare for unequal
3	Compare for less-than
4	Compare for greater-than

value

input

is a 32-bit integer to be used for comparison with the value in the given counter. *value* = *n**1000. For example, 17.09 should be stored as 17090.

rate

input

determines whether a nonrated or rated value is used in the comparison:

- 0 indicates a nonrated value (equivalent to REPORT RATE OFF).
- 1 indicates a rated value (equivalent to REPORT RATE ON).

sqlname

input

is a 16-word array specifying the SQL run unit. *sqlindex* specifies the index number. These two fields are used in conjunction with the SQLSTMT entity. If passed, MEASREAD_DIFF_ returns the records with the specified run-unit and index numbers.

sqlindex

input

is the SQL index. See *sqlname*. A value of -1 specifies all SQL indexes.

NOTE: If the *if^item^def* structure is passed, it must first be initialized. To pass the structure without a value, IF^ITEM^DEF.Item^Name must be initialized to 0. Similarly, a value of 0 in IF^ITEM^DEF.SQLName means that no SQL (*run-unit*,*index*) is specified.

totals^num

output

INT(32).EXT:ref:1

is a count of the records used to accumulate the summary record.

templateversion

input

FIXED:value

(Measure G11 and later) is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc*, if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

See TEMPLATE-VERSION (page 143) for further information on obtaining external records corresponding to a format that you can handle.

Usage Notes

- If the longer IF^ITEM^G09^DEF template is used, the define G09^format must be set. If the G09^format is not set, part of the *sqlname* field is interpreted as *sqlindex*, and any records returned are likely to be incorrect.

```
DEFINE G09^format          = relation^operator.0#;
DEFINE G09^rel^operator    = relation^operator.13:15#;
```

  You can use either template format when accessing G09 data files. You can also use the G09 template when accessing G08 and earlier data files if the number passed in the field *value* does not exceed the amount that can be passed in the INT(32) *value* field of the earlier structure, and if the *sqlname* value does not exceed 32 characters.

- In Measure H01 and later PVUs, MEASREAD_DIFF_ accepts DISCOPEN, DISKFILE, FILE, and SQLSTMT entity descriptors for ANSI SQL objects or partitions.

MEASREADACTIVE

Reads data from a currently active counter. Before reading a counter, you must obtain a valid measurement number (generally, by using MEASMONSTATUS (page 419)).

In D-series and pre-G08 Measure PVUs, MEASREADACTIVE cannot read DISCOPEN, DISKFILE, or PROCESSH counters. In Measure G08 and later PVUs, the MEASREADACTIVE procedure cannot read DISCOPEN or PROCESSH counters. To access these counter values, use the

MEASREAD_DIFF_ procedure to read from the currently active measurement data file. The collection interval specified for the measurement determines how recently the counter values were written to the data file.

```
error := MEASREADACTIVE ( meascb                    ! i,o
                         ,measnum                   ! i
                         ,entitydesc                ! i
                         ,loc                       ! o
                         ,bufsize                   ! i
                         ,bytesret                  ! o
                         ,[ entity^index ]          ! i,o
                         ,[ templateversion ] );    ! i
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. After you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

measnum

input

INT:value

is the measurement number. Use the *measnum* value returned by the MEASCONFIGURE procedure or find the *measnum* value using MEASMONSTATUS.

entitydesc

input

INT:ref:*

is the entity type and entity specification of the desired active counter record. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. Because MEASREADACTIVE uses *entitydesc* to directly access the entity control block (which in turn is used to access the counter record in system data space), *entitydesc* must identify exactly one entity. If necessary, use the appropriate operating system information procedures to obtain the names and numbers that let you uniquely identify the entity.

You need not specify all the fields in an entity descriptor if a subset of the fields is sufficient to identify the data to retrieve. For example, if you identify a process by its CPU and PIN, you need not also specify the process name or program file name. Conversely, a call to retrieve data might need to specify more fields than were used to configure the measurement. For example, an *entitydesc* in your measurement configuration might specify a wildcard. If you wanted to retrieve data for only a subset of the objects measured, the *entitydesc* in your MEASREADACTIVE call would probably specify more fields.

entitydesc can be a DISKFILE, FILE, or SQLSTMT entity descriptor for ANSI SQL objects or partitions.

NOTE: Wildcards are not permitted in entity descriptors passed to the MEASREACTIVE callable procedure. If you want to use wildcards, call the MEAS_READACTIVE_MANY_ procedure rather than this one.

For ANSI SQL objects or partitions, *entitydesc* can be a DISKFILE, FILE, or SQLSTMT entity descriptor. Specify the Guardian name of the file, not the ANSI SQL name.

loc

output

INT.EXT:ref:*

is the destination buffer for the counter records. Counter records are written to the buffer in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

bufsize

input

INT:value

is the size in bytes of the destination buffer, *loc*.

bytesret

output

INT:ref:1

is the number of bytes returned to the destination buffer, *loc*. If *bytesret* is larger than *bufsize*, error 3204 (ERR^BUFTOOSMALL) is returned.

entity^index

input, output

FIXED:ref:1

is applicable to G08 and later MEASREADACTIVE requests for DISKFILE records.

On input, *entity^index* is either -1 or a value returned by a previous call to MEASREADACTIVE for the record described by *entity^desc*.

On output, *entity^index* contains an internal value that enables Measure to directly fetch the record described by *entity^desc* on subsequent calls to MEASREADACTIVE. If -1 is specified, Measure searches all DISKFILE records in a processor until a match is found. If *entity^index* is not -1, Measure assumes it to be the index of the DISKFILE record that is described by *entity^desc*. The value of *entity^index* and the contents of *entity^desc* are then examined. If these do not match, error 3402 (errm^cannotaccess) or error 3404 (errm^notmeasuring) is returned.

templateversion

input

FIXED:value

(Measure G11 and later) is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc*, if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

Usage Note

- MEASREADACTIVE supports buffer sizes up to 32 KB only for active measurement of entities. For buffers larger than 32 KB, use MEAS_READACTIVE_ (page 433).
- In Measure H01 and later PVUs, MEASREADACTIVE and MEAS_READACTIVE_ accept DISKFILE, FILE, or SQLSTMT entity descriptors for ANSI SQL objects or partitions.

MEAS_READACTIVE_

Reads data from a currently active counter in Measure G09 and later PVUs. Before reading a counter, you must obtain a valid measurement number (generally, by using the MEASMONSTATUS procedure).

The MEAS_READACTIVE_ procedure cannot read DISCOPEN or PROCESSH counters. To access these counter values, use the MEASREAD_DIFF_ procedure to read from the currently active measurement data file. The collection interval specified for the measurement determines how recently the counter values were written to the data file.

```
error := MEAS_READACTIVE_( meascb                ! i,o
                         , measnum               ! i
                         , entitydesc            ! i
                         , loc                   ! o
                         , bufsize               ! i
                         , bytesret              ! o
                         ,[ entity^index ]       ! i,o
                         ,[ templateversion ]); ! i
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. After you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

measnum

input

INT:value

is the measurement number. Use the *measnum* value returned by the MEASCONFIGURE procedure or find the *measnum* value using MEASMONSTATUS.

entitydesc

input

INT:ref:*

is the entity type and entity specification of the desired active counter record. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. Because MEAS_READACTIVE_ uses *entitydesc* to directly access the entity control block (which in turn is used to access the counter record in system data space), *entitydesc* must identify exactly one entity. If necessary, use the appropriate operating system information procedures to obtain the names and numbers that let you uniquely identify the entity.

entitydesc can be a DISKFILE, FILE, or SQLSTMT entity descriptor for ANSI SQL objects or partitions.

NOTE: Wildcards are not permitted in entity descriptors passed to the MEAS_REACTIVE_ callable procedure. If you want to use wildcards, call the MEAS_READACTIVE_MANY_ procedure rather than this one.

loc

output

INT.EXT:ref:*

is the destination buffer for the counter records. Counter records are written to the buffer in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

bufsize

input

INT(32):value

is the size in bytes of the destination buffer, *loc*.

bytesret

output

INT(32):ref:1

is the number of bytes returned to the destination buffer, *loc*. If *bytesret* is larger than *bufsize*, error 3204 (ERR^BUFTOOSMALL) is returned.

entity^index

input, output

FIXED:ref:1

is applicable to active measurement requests for DISKFILE records.

On input, *entity^index* is either -1 or a value returned by a previous call to MEAS_READACTIVE_ for the record described by *entity^desc*.

On output, *entity^index* contains an internal value which enables Measure to directly fetch the record described by *entity^desc* on subsequent calls to MEAS_READACTIVE_. If -1 is specified, Measure searches all DISKFILE records in a processor until a match is found. If *entity^index* is not -1, Measure assumes it to be the index of the DISKFILE record which is described by *entity^desc*. The value of *entity^index* and the contents of *entity^desc* are then examined. If they do not match, error 3402 (errm^cannotaccess) or error 3404 (errm^notmeasuring) is returned.

templateversion

input

FIXED:value

(Measure G11 and later) is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc*, if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

Usage Notes

In Measure H01 and later PVUs, MEASREADACTIVE and MEAS_READACTIVE_ accept DISKFILE, FILE, or SQLSTMT entity descriptors for ANSI SQL objects or partitions.

MEAS_READACTIVE_MANY_

Retrieves multiple active counter records configured in a processor, without incurring overhead for messages. This procedure is available in H-series and J-series RVUs.

The MEAS_READACTIVE_MANY_ procedure cannot read PROCESSH, SQLSTMT, or USERDEF counters. To access PROCESSH counter values, use the MEASREAD_DIFF_ procedure to read from the currently active measurement data file. To access SQLSTMT or USERDEF counters, use the MEAS_READACTIVE_ procedure.

```
error := MEAS_READACTIVE_MANY_ ( measnum        ! i
                               , mode           ! i
```

```
                              , entitydesc          ! i
                              , buffer              ! o
                              , bufsize             ! i
                              , recs                ! o
                              , context             ! i,o
                              ,[ templateversion ]) ! i
```

measnum

input

INT:value

is the measurement number. Use the *measnum* value returned by the MEASCONFIGURE procedure or find the *measnum* value using MEASMONSTATUS.

mode

input

INT:value

is one of the values READACTIVE_MANY_LIST (0), READACTIVE_MANY_EXACT (1), or READACTIVE_MANY_CHANGED (2). The semantics of these values depends on the value of the *context* parameter.

entitydesc

input

INT:EXT:ref:*

is the entity type and entity specification of the desired counter records. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. If you specify *entitydesc* as a template for multiple specifications, MEAS_READACTIVE_MANY_ returns all counter records that fit the template.

entitydesc can contain wildcards. Using a wildcard descriptor along with LIST *mode* is the best way to get the records of all the instances of an entity configured on a processor. This procedure does not use string-based fields, such as OSS Pathname or program filename, for descriptor matching. Such strings are ignored, if provided; if you need to use them, call the MEASREADACTIVE procedure rather than this one.

NOTE: *entitydesc* can be a DISKFILE, DISCOPEN, or FILE entity descriptor for ANSI SQL objects or partitions. However, there is currently no support for ANSI SQL names or ANS UIDs in this API, so those descriptor fields will be ignored.

buffer

output

INT.EXT:ref:*

is the destination buffer for the counter records. Counter records are written to the buffer in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

bufsize

input

INT(32):value

is the size in bytes of the destination buffer, *buffer*.

recs

output

INT:EXT:ref:1

is the number of records returned.

context

input, output

FIXED:EXT:ref:1

interacts with the value of the mode parameter to determine which records are returned.

On input, if *context* is -1F or 0F:

- The counter type for the entity type specified in *entitydesc* is searched until a counter matches the information in *entitydesc*.
- If *mode* is READACTIVE_MANY_EXACT, one item is returned.
- If *mode* is READACTIVE_MANY_LIST, all items that match *entitydesc* are returned until *bufsize* is exhausted.
- If *mode* is READACTIVE_MANY_CHANGED, all items that match the *entitydesc* are returned, but the ERROR field in the Header part of the record is updated to show whether that record has changed since last requested. Specifically, if the record has not changed and the style is ZMS, the 0th and 5th bits of *entity*.HDR.ERROR are set to 1. Otherwise, the 5th bit is 0. If the style is Legacy, a value of -4 is returned in the *entity*.ERROR field to indicate unchanged records. This Error field is part of the counter record returned (and is specific to it). It is not related to the *error* values returned by this procedure.
- On successful return, the *context* contains information about the next entity to be processed. It should be passed as is for the future calls in that iteration.

On output

- If *mode* is READACTIVE_MANY_EXACT, *context* contains information about the matched entity. Passing this value back the next time you want data for the same entity will increase the performance of the request.
- If *mode* is READACTIVE_MANY_CHANGED or READACTIVE_MANY_LIST, a context of 0F indicates that no more entities matching *entitydesc* are available for retrieval. -1F indicates an error return.

On input, if *context* has a value other than -1F or 0F:

- If *mode* is EXACT, the counter referenced by *context* is returned, provided that it applies to the same object.
- If *mode* is LIST and the *context* value applies to the same object, and the entity type is consistent with *entitydesc*, then counter records of the same type, from this counter onward, are returned until *bufsize* is exhausted. LIST terminates at the end of one pass through the linked counter list. If at any time context fails the check to ensure that it applies to the same counter, an error is returned. A LIST/changed-record operation must restart from the beginning.
- If *mode* is CHANGED-RECORDS, the behavior is similar to LIST, but the ERROR field in the Header part of the record is updated to show whether that record has changed since last requested. If the record has not changed and the style is ZMS, the 0th and 5th bits of *entity*.HDR.ERROR are set to 1. Otherwise, the 5th bit is 0. If the Style is Legacy, a value of -4 is returned in *entity*.ERROR field to indicate unchanged records.
- On successful return, *context* contains information about the next entity to be processed. It should be passed as is for the future calls in that iteration. A return *context* of 1F indicates an error.

templateversion

input

FIXED:value

is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc* if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

Usage Notes

- The application that calls MEAS_READACTIVE_MANY must be running on the same processor as the records to be retrieved. This procedure has high performance precisely because it does not exchange interprocess messages. To retrieve records from multiple processors, run a copy of the application on each processor.
- MEAS_READACTIVE_MANY does not return all the typical header and ID information for an entity. Instead, the procedure returns a subset of descriptive information, such as the PIN, LDEV, OCB or ACB number, and filename.
- In the case of ZMS style records, the procedure returns format and subsystem versions.
- If you specify the TEMPLATE^VERSION parameter to request ZMS style records, counter values are returned according to the external ZMS *type*.CTRS^DEF template. If you omit the TEMPLATE^VERSION parameter, legacy style external records are returned.
- For best performance, use the ZMS `entity-template-version` literal from the MEASDDLS file for the current RVU. Using an earlier template version or requesting legacy style records increases performance cost.
- To use this interface, you must first start a measurement containing the entities to be accessed. Measure needs that measurement and the corresponding *measnum* to identify counter records that have changed.
- In Measure H01 and later PVUs, MEAS_READACTIVE_MANY_ accepts DISKFILE, DISCOPEN, or FILE entity descriptors for ANSI SQL objects or partitions.

MEASREADCONF

Returns the measurement configuration from a data file. Before reading from a measurement data file, you must obtain read access to the file using the MEASOPEN procedure.

In Measure G09 and later PVUs, MEASREADCONF has a parameter for retrieving settings that reports the configuration of the journal segment functions.

```
error := MEASREADCONF ( dfnum           ! i
                      ,[ contab ]       ! o
                      ,[ bufsize ]      ! i
                      ,[ bytesret ]     ! o
                      ,[ starttime ]    ! o
                      ,[ stoptime ]     ! o
                      ,[ interval ]     ! o
                      ,[ entities ]     ! o
                      ,[ ctrspace ]     ! o
                      ,[ version ]      ! i
                      ,[ max^ent ]      ! i
                      ,[ settings]);    ! o
```

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

contab

output

INT.EXT:ref:*

is an array that defines the measurement configuration. For the format of the *contab* array, see MEASCONFIGURE (page 358).

bufsize

input

INT:value

is the size in bytes of the destination buffer, *contab*. If *bufsize* is greater than 32,000 bytes, error 3203 (ERR^BADPARAMS) is returned. To get the *contab* size returned in *bytesret*, *bufsize* must be at least 4 bytes.

bytesret

output

INT:ref:1

is the size in bytes of the *contab* array (and the number of bytes returned to the destination buffer *contab*.) If *bytesret* is larger than *bufsize*, error 3204 (err^buftoosmall) is returned, and no *contab* value is returned.

starttime

output

FIXED:ref:1

is the start time of the measurement.

stoptime

output

FIXED:ref:1

is the stop time of the measurement. A value of -1 indicates that no stop time was specified.

interval

output

FIXED:ref:1

is the collection interval of the measurement. A value of -1 indicates that no collection interval was specified.

entities

output

INT(32):ref:MAX^T+1

is an array that contains the maximum number of entities of each entity type measured concurrently. The array elements are in entity-type order, as shown in this table. A value of 0 indicates the entity type was not included in the configuration. MEASREADCONF does not return the *entities* and *ctrspace* arrays for currently active measurements, MEASSTATUS does. This array should be declared with a zero base. For example: INT(32).ENTITIES[0:MAX^T];

This table lists the entity types and their literal values and numeric identifiers.

Entity Type	Literal Value	Numeric Identifier
CPU	CPU^T	1
PROCESS	PROCESS^T	2
PROCESSH	PROCESSH^T	3
USERDEF	USERDEF^T	4
FILE	FILOP^T	5
DISCOPEN	DFILOP^T	6
DISC	DISC^T	7

Entity Type	Literal Value	Numeric Identifier
DEVICE	IODEV^T	8
LINE	LINE^T	9
NETLINE	NETLINE^T	10
SYSTEM	REMSYS^T	11
CLUSTER	CLUSTER^T	12
TERMINAL	TERM^T	13
TMF	TMF^T	14
SQLPROC	SQLPROC^T	15
SQLSTMT	SQLSTMT^T	16
OPDISK	OPDISK^T	17
CONTROLLER	CTRL^T	18
SERVERNET	SVNET^T	18
DISKFILE	DISKFILE^T	19
OSSCPU	OSSCPU^T	20
OSSNS	OSSNS^T	21
(Max value)	MAX^T	24

ctrspace

output

INT(32):ref:MAX^T+1

is an array that contains the maximum counter space in words used by each entity type. The array elements are in entity-type order, as shown for *entities*. A value of 0 indicates the entity type was not included in the configuration. MEASREADCONF does not return the *entities* and *ctrspace* arrays for currently active measurements, but MEASSTATUS does. Declare this array with a zero base. For example:

INT(32).CTRSPACE[0:MAX^T];

version

input

INT:value

is the Measure product version when the data file was created. The value consists of two parts: bits 0:7 contain an alphabetic character (for example, D), and bits 8:15 contain a numeric value (for example, 30). Together they express the product version (for example, D30). If the parameter is not passed or is passed as zero, the version defaults to the currently installed Measure PVU.

max^ent

input

INT:value

is the maximum number of entities the caller allocated for the *entities* and *ctrspace* arrays. This value should normally be specified as MAX^T. (MAX^T is a literal definition from the MEASDECS file that specifies the maximum entity type code.). If specified as less than MAX^T, it limits (truncates) the amount of *entities* or *ctrspace* array information returned. (For example, a *max^ent* of 1 limits the return to CPU information only.) If *max^ent*

is passed as zero or a negative number, error 3203 (ERR^BADPARAMS) is returned. If *max^ent* is omitted, it defaults to MAX^T.

settings

output

INT:ref:1

is an array of flags that identifies configuration attributes of the measurement represented by the data file:

settings.0:8	Reserved for future use.
settings.9	1 = Counter data records are suppressed in this file.
settings.10	1 = An SQL/MX journal segment is currently attached to the calling process for this file.
settings.11	1 = An SQL/MX journal segment is under construction for this file.
settings.12	1 = An SQL/MX journal segment is in this file.
settings.13	1 = An OSS journal segment is currently attached to the calling process for this file.
settings.14	1 = An OSS journal segment is under construction for this file.
settings.15	1 = An OSS journal segment is in this file.

Usage Notes

- A call to MEASREADCONF is equivalent to performing an INFO * command by using MEASCOM. To obtain return information, you must have previously issued a call to MEASOPEN to establish an open to an executing MEASFH process. MEASREADCONF obtains measurement configuration information through interprocess communication with the MEASFH process.
- The MEASINFO procedure provides an alternative means of obtaining configuration information from a data file.
- MEASREADCONF parameters are optional. However:

  dfnum

  If *dfnum* is omitted, error 3201 (ERR^MISSINGPARAM) is returned.

  contab, bufsize, bytesret

  These parameters are a group. That is, one cannot be passed without the other two, or error 3201 (ERR^MISSINGPARAM) is returned.
- If OSS or SQL/MX journal segment construction is requested for a measurement, the status is always reported as under construction while the measurement is active. When the measurement is stopped, the setting continues to be reported as under construction until updating of the file is complete.

MEAS_RETRIEVE_TIMERCELLS_

Retrieves the values of the specified timer cells. Use this procedure if your application requires fine granularity timers to maintain counters outside of Measure. Measure calls this procedure implicitly when you request values of a timer-cell counter defined for the USERDEF entity.

Retrieving a timer-cell value does not reset the value.

```
error := MEAS_RETRIEVE_TIMERCELLS_ ( count       ! i
                                   , indexes     ! i
                                   , timers);    ! i,o
```

count

INT:VALUE

is the number of timer cells to retrieve. Measure can retrieve multiple timer cell values during a single rendezvous, so it makes sense to retrieve several timer cell values in a single call rather than request each timer cell separately.

indexes

INT:EXT:ref:*count*

is an array containing the indexes of timer cells to retrieve.

timers

FIXED:EXT:ref:*count*

is an array of timer-cell values. On input, the array is empty. On output, it contains the values of the timer cells, in the same order in which their indexes were specified in the previous parameter. Timer-cell values are in microsecond units.

MEAS_SQL_MAP_INIT_

In Measure H02 and later PVUs, MEAS_SQL_MAP_INIT_ initiates the SQL/MX mapping session. This must be done if one of the following APIs is going to be used:

```
meas_getdescinfo_
measlistename
measlistgname
```

and there is no SQL journal in the indicated Measure data file, so resolution comes from the SQL/MX subsystem.

```
error := MEAS_SQL_MAP_INIT_ ( connectionInfo ); !i,o
```

error

INT

is an error code indicating the outcome of the operation. Zero means a successful completion. Possible error codes include:

err^sqlmx^map^dupconnection	The SQL/MX mapping session is already established for this application.
err^sqlmx^map^init	The SQL/MX mapping session cannot be started.

connectionInfo

INT(32) EXT:ref:25

is allocated by the caller to be at least 100 bytes.

Usage Notes

For best efficiency, this API should be called only once per application process.

Example

This a simplified example in pTAL pseudo code to show how to use MEAS_SQL_MAP_INIT_() and MEAS_SQL_MAP_STOP_().

```
int(32)  .connectionInfo[0:24];  ! at least 100 bytes
! at the start of the application
! if (error := MEAS_SQL_MAP_INIT_(connectionInfo)) then
! handle error

! do work including calls to one or more of the APIs that require the
! connectionInfo argument as input unless we are analyzing a data file
! before exiting the application
```

```
! if connectionInfo[0] <> 0d then
! MEAS_SQL_MAP_STOP_(connectionInfo);
```

MEAS_SQL_MAP_STOP_

In Measure H02 and later PVUs, MEAS_SQL_MAP_STOP_ stops the SQL/MX mapping session. This should be done before an application process that has called MEAS_SQL_MAP_INIT_ exits.

```
error := MEAS_SQL_MAP_STOP_ ( connectionInfo );  !i
```

The only error that can be returned is ERR^BOUNDS.

connectionInfo

input

INT(32) EXT:ref:25

is allocated by the caller to be at least 100 bytes.

MEAS_SQLNAME_COMPARE_

In Measure H02 and later PVUs, MEAS_SQLNAME_COMPARE_ compares two fully qualified ANSI SQL names in external format. The names cannot contain wildcards and they must be of the same object type.

```
status := MEAS_SQLNAME_COMPARE_ ( name1            ! i
                                , name1_length     ! i
                                , name2            ! i
                                , name2_length );  ! i
```

status

INT:ref:1

is a status code indicating the outcome of the operation:

-1	*name1* < *name2* (a.b.c < a.bz.c)
0	*name1* = *name2*
1	*name1* > *name2* (a.bz.c > a.b.c)
err^badformatsqlname	One or more of the specified ANSI SQL names contained syntax errors or were not fully qualified.
err^sql^api^internal	One or more of the specified ANSI SQL names could not be parsed.

name1

input

STRING:EXT:ref:*

is a buffer that contains input name 1 in external format. A name space keyword is required.

name1_length

input

INT:value

is the size, in bytes, of input name 1.

name2

input

STRING:EXT:ref:*

is a buffer that contains input name 2 in external format. A name space keyword is required.

name2_length

input

INT:value

is the size, in bytes, of input name 2.

Usage Notes

- ANSI SQL objects have independent name spaces and, if identical names exist in different name spaces, each instance of the name refers to a different object.
- The name spaces used by Measure are TABLE, INDEX and MODULE. TABLE is implied by using the keywords CATALOG, SCHEMA and TABLE. INDEX is implied by using the keyword INDEX. MODULE is implied for SQLSTMT (in MEASCOM there is no MODULE keyword).
- When calling the MEASSQLNAME_* APIs, the input ANSI SQL names must be preceded by a name space keyword.

MEAS_SQLNAME_RESOLVE_

In Measure H02 and later PVUs, MEAS_SQLNAME_RESOLVE_ combines ANSI SQL name parts to create a fully qualified name in normalized external format. This routine resolves default CATALOG and SCHEMA settings and returns the fully qualified ANSI SQL name in a normalized external format.

```
error := MEAS_SQLNAME_RESOLVE_ ( result_name             ! o
                               , result_name_max         ! i
                               , result_name_length      ! o
                               , name );                 ! i
                               , name_length             ! i
                               , [ def_catalog ]         ! i
                               , [ def_catalog_length ]  ! i
                               , [ def_schema ]          ! i
                               , [ def_schema_length ] ;! i
```

error

INT

is an error code indicating the outcome of the operation. Zero means a successful completion. Possible error codes include:

err^badformatsqlname	One or more of the specified ANSI SQL names contained syntax errors.
err^buftoosmall	*result_name_max* is too small to hold the fully qualified name in normalized external format. If indicated, the required size will be returned in *result_name_length*.
err^sql^api^internal	The ANSI SQL name specified could not be parsed.

result_name

output

STRING:EXT:ref:*

is a buffer that contains the returned fully qualified name in normalized external format.

result_name_max

input

INT:value

is the size, in bytes, of the *result_name* buffer.

result_name_length

output

INT:ref:1

if not err^buftoosmall, this is the size, in bytes, of the fully qualified name returned in the *result_name* buffer. If err^buftoosmall, this displays the required size.

name

input

STRING:EXT:ref:*

is a buffer that contains the input name in external format to be expanded to a fully qualified name. If *name* does not contain the catalog and/or schema name, the default catalog (*def_catalog*) and/or default schema (*def_schema*) name is used. In that case, if the necessary *def_catalog* and/or *def_schema* name is missing, an error is returned. Wildcards are permitted. A name space keyword is required.

name_length

input

INT:value

is the size in bytes of the input name.

def_catalog

input

STRING:EXT:ref:*

is a buffer that contains the default catalog name that is used if the input *name* omits the catalog field, in which case it must be specified.

def_catalog_length

input

INT:value

is the size, in bytes, of the default catalog name passed by the *def_catalog* parameter.

def_schema

input

STRING:EXT:ref:*

is a buffer that contains the default schema name that is used if the input *name* omits the schema field, in which case it must be specified.

def_schema_length

input

INT:value

is the size, in bytes, of the default schema name passed by the *def_schema* parameter.

Usage Notes

- ANSI SQL objects have independent name spaces and, if identical names exist in different name spaces, each instance of the name refers to a different object.
- The name spaces used by Measure are TABLE, INDEX and MODULE. TABLE is implied by using the keywords CATALOG, SCHEMA and TABLE. INDEX is implied by using the keyword INDEX. MODULE is implied for SQLSTMT (in MEASCOM there is no MODULE keyword).
- When calling the MEASSQLNAME_* APIs, the input ANSI SQL names must be preceded by a name space keyword.

MEAS_SQLNAME_SCAN_

In Measure H02 and later PVUs, MEAS_SQLNAME_SCAN_ parses a fully qualified, possibly wildcarded, ANSI SQL name in external format. This routine verifies the syntax of an ANSI SQL name.

```
error := MEAS_SQLNAME_SCAN_ ( name            ! i
                            , name_length     ! i
```

error

INT

is an error code indicating the outcome of the operation. Zero means a successful completion. Possible error codes include:

err^badformatsqlname	The specified ANSI SQL name contained syntax errors.
err^sql^api^internal	The specified ANSI SQL name could not be parsed.

name

input

STRING:EXT:ref:*

is a buffer that contains the input name to be scanned in external format. A name space keyword is required.

name_length

input

INT:value

is the size, in bytes, of the input name.

Usage Notes

- ANSI SQL objects have independent name spaces and, if identical names exist in different name spaces, each instance of the name refers to a different object.
- The name spaces used by Measure are TABLE, INDEX and MODULE. TABLE is implied by using the keywords CATALOG, SCHEMA and TABLE. INDEX is implied by using the keyword INDEX. MODULE is implied for SQLSTMT (in MEASCOM there is no MODULE keyword).
- When calling the MEASSQLNAME_* APIs, the input ANSI SQL names must be preceded by a name space keyword.

Example

These are a couple of simplified examples in pTAL pseudo code that call MEAS_SQLNAME_SCAN_:

```
STRING .EXT inbuf[0:1023] :=["table cat.sch.t"];

if (error := MEAS_SQLNAME_SCAN_ (inbuf,
                                 15))
  ! handle error
else
  ! ANSI SQL TABLE name is OK


STRING .EXT inbuf[0:1023] :=["module cat.sch.m"];

if (error := MEAS_SQLNAME_SCAN_ (inbuf,
                                 16))
  ! handle error
```

```
else
  ! ANSI SQL MODULE name is OK
```

MEASSTATUS

Returns information about a currently active measurement.

In Measure G09 and later PVUs, MEASSTATUS has a parameter for retrieving settings that reports the configuration of the journal segment functions.

```
error := MEASSTATUS ( meascb          ! i,o
                    ,measnum          ! i
                    ,cpus             ! o
                    ,starttime        ! o
                    ,stoptime         ! o
                    ,interval         ! o
                    ,entities         ! o
                    ,ctrspace         ! o
                    ,[ max^ent ]      ! i
                    ,[ settings ]); ! o
```

meascb

input, output

INT:ref:$LEN(MEASCB^DEF) / 2

is a control block where the Measure subsystem stores data for subsequent procedure calls. Before calling the first Measure procedure that uses *meascb*, you must allocate space in your global data area for the control block and initialize each element of the control block to -1. After you pass *meascb* to a Measure procedure, do not modify its contents.

The file $SYSTEM.SYS*nn*.MEASDECS contains the structure definition for the control block descriptor (MEASCB^DEF).

measnum

input

INT:value

is the measurement number. Use the *measnum* value returned by the MEASCONFIGURE procedure.

cpus

output

INT:ref:1

indicates the CPUs being measured. Each bit represents a CPU number. If a bit is set to 1, that CPU is being measured. Otherwise, that CPU is not.

starttime

output

FIXED:ref:1

is the start time of the measurement.

stoptime

output

FIXED:ref:1

is the stop time of the measurement.

interval

output

FIXED:ref:1

is the collection interval of the measurement.

entities

output

INT(32):ref:MAX^T+1

is an array that contains the number of entities of each entity type currently being measured. The array elements are in entity-type order, as this table shows. A value of 0 indicates the entity type was not included in the configuration.

Entity Type	Literal Value	Numeric Identifier
CPU	CPU^T	1
PROCESS	PROCESS^T	2
PROCESSH	PROCESSH^T	3
USERDEF	USERDEF^T	4
FILE	FILOP^T	5
DISCOPEN	DFILOP^T	6
DISC	DISC^T	7
DEVICE	IODEV^T	8
LINE	LINE^T	9
NETLINE	NETLINE^T	10
SYSTEM	REMSYS^T	11
CLUSTER	CLUSTER^T	12
TERMINAL	TERM^T	13
TMF	TMF^T	14
SQLPROC	SQLPROC^T	15
SQLSTMT	SQLSTMT^T	16
OPDISK	OPDISK^T	17
CONTROLLER	CTRL^T	18
SERVERNET	SVNET^T	18
DISKFILE	DISKFILE^T	19
OSSCPU	OSSCPU^T	20
OSSNS	OSSNS^T	21
(Max value)	MAX^T	24

ctrspace

output

INT(32):ref:MAX^T+1

is an array that contains the counter space in words currently in use by each entity type. The array elements are in entity-type order, as shown for *entities*. A value of 0 indicates the entity type was not included in the configuration.

max^ent

input

INT:value

is the maximum number of entities the caller allocated for the *entity* and *ctrspace* arrays.

settings

output

INT:ref:1

is an array of flags that identifies configuration attributes of the measurement represented by the data file:

settings.0:8	Reserved for future use.
settings.9	1 = Counter data records are suppressed in this file.
settings.10:11	Reserved for future use.
settings.12	1 = An SQL/MX journal segment is under construction for this file.
settings.13	1 = An SQL/MX journal segment is present in this file.
settings.14	1 = An OSS journal segment is under construction for this file.
settings.15	1 = An OSS journal segment is present in this file.

Usage Notes

If OSS or SQL/MX journal segment construction is requested for a measurement, the status is always reported as under construction while the measurement is active. When the measurement is stopped, the setting continues to be reported as under construction until updating of the file is complete.

MEASWRITE_DIFF_

Reads a window of one or more counter records from a measurement data file and places the results in a structured file.

Before reading from a measurement data file, you must obtain read access to the file using the MEASOPEN procedure. The data file can be associated with a currently active measurement, or it can contain data from an inactive measurement.

The convention used for naming the structured file is identical to that of MEASCOM. The procedure uses the specified entity name as a file name in the volume and subvolume specified in the procedure call. If the file already exists, the retrieved information is appended to it. For example, if the call requests CPU information be put in $USER.MEAS, the information is put in $USER.MEAS.CPU.

The procedure call specifies both the start time and the stop time. The records returned provide the statistics collected within the interval. You can request interval records, summary records, or both.

The procedure call can specify options (*all^recs*, *zero^reports*, *totals*, *loadid*, and *if^item*) to obtain additional processing

In Measure G09 and later PVUs, the parameter if^item has a field template if^item^g09^def that is available for the longer (128 byte) SQL/MX run unit names. This template coexists with the pre-G09 if^item^def template, thereby preserving compatibility with existing applications and older data files.

```
error := MEASWRITE_DIFF_( dfnum                    ! i
                        ,entitydesc                ! i
                        ,volume^subvol             ! i
                        ,[ from^time ]             ! i
                        ,[ to^time ]               ! i
                        ,[ timetol ]               ! i
                        ,[ version ]               ! i
                        ,[ all^recs ]              ! i
                        ,[ zero^reports ]          ! i
```

```
                          ,[ totals ]                   ! i
                          ,[ loadid ]                   ! i
                          ,[ if^item ]                  ! i
                          ,[ templateversion ] );       ! i
```

dfnum

input

INT:value

is the data file access number. Use the *dfnum* value returned by the MEASOPEN procedure.

entitydesc

input

INT.EXT:ref:*

is the entity type and entity specification of the desired counter records. Specify *entitydesc* as one of the descriptors listed in the *contab* description of the MEASCONFIGURE procedure. If you specify *entitydesc* as a template for multiple specifications, MEASWRITE_DIFF_ returns all counter records that fit the template.

For ANSI SQL objects or partitions, *entitydesc* can be a DISCOPEN, DISKFILE, FILE, or SQLSTMT entity descriptor. The SQLSTMT ANSI entity descriptor is used when an ANSI SQL name is to be specified for the retrieval of SQLSTMT counter records.

volume^subvol

input

INT.EXT:ref:8

is the destination volume and subvolume for the counter records. Counter records are written to the file in DDL record format. For the DDL record definitions for each entity type, see Chapter 3: Entities and Counters (page 133).

For a local system, the format of *volume^subvol* (when redefined as a string [0:15]) is:

```
Volume^SubVol[0]    = '$'
Volume^SubVol[1:7]  = volume name (padded with spaces)
Volume^SubVol[8:15]= subvolume name (padded with spaces)
```

For a remote system, the format is:

```
Volume^SubVol[0]    = '\'
Volume^SubVol[1]    = system number
Volume^SubVol[2:7]  = volume name (padded with spaces)
Volume^SubVol[8:15]= subvolume name (padded with spaces)
```

from^time

input

FIXED:value

is the start time of the measurement window. Specify *from^time* as a Julian date based on local civil time in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP (see the *Guardian Procedure Calls Reference Manual*). A value of -1 signifies the beginning of the measurement. If no *from^time* is specified, the default is -1.

to^time

input

FIXED:value

is the stop time of the measurement window. Specify *to^time* as a Julian date based on local civil time in four-word-by-one-microsecond format as provided by the system procedure JULIANTIMESTAMP. (See the *Guardian Procedure Calls Reference Manual*.) A value of -1 signifies the end of the measurement. If no *to^time* is specified, the default is -1.

timetol

input

FIXED:value

is a tolerance value to be applied to the time window bounded by *from^time* and *to^time*, effectively decreasing *from^time* by *timetol* and increasing *to^time* by the same amount. A value of -1 widens the time window to include the latest record earlier than *from^time* and the earliest record later than *to^time*. If no *timetol* is specified, the default is -1.

version

input

INT:value

is the Measure product version when the data file was created. The value consists of two parts. Bits 0:7 contain an alphabetic character (for example, D), and bits 8:15 contain a numeric value (for example, 30). Together they express the product version (for example, D30).

all^recs

input

INT:value

requests all interval records within the specified window if the value 1 is passed. If omitted or 0, only summary records are returned.

zero^reports

input

INT:value

requests the exclusion of zero records if the value 0 is passed. If omitted or 1, zero records are returned.

totals

input

INT:value

is a value of 0, 1, or 2, which indicates:

0	(Default) Omit total record; include individual counter records.
1	Include individual records and total record.
2	Omit individual counter records; include total record.

loadid

input

INT.EXT:ref:4

is an 8-byte string to be included in all records returned. If no *loadid* is specified, the default is blank (spaces).

if^item

input

INT .EXT:ref:$LEN(IF^ITEM^DEF) / 2

is a structure (IF^ITEM^DEF, defined in the MEASDECS file):

```
STRUCT IF^ITEM^DEF (*);
Begin
Int     Item^Name[0:15];  !Item name in entity record
Int     Relation^Operator;!Relation Operator
                          !  1 -> Equal        =
                          !  2 -> Not Equal    <>
```

```
                              !  3 -> Less Than    <
                              !  4 -> Greater Than >
   Int(32) Value;             !Value used in comparison
   Int     Rate;              !Use rated or nonrated value
                              !for condition checking:
                              !  0 -> No-Rate, 1 -> Rated
   Int     SQLName[0:15];     !SQL run-unit in Ascii
   Int     SQLIndex;          !SQL Index number
   End;
```

or

if^item

input

INT .EXT:ref:$LEN(IF^ITEM^G09^DEF) / 2

is a structure (IF^ITEM^G09^DEF, defined in the MEASDECS file):

```
STRUCT IF^ITEM^G09^DEF (*);
Begin
Int     Item^Name[0:15];  !Item name in entity record
Int     Relation^Operator;!Relation Operator
                          !See Usage Notes (page 452)
                          !Bit 0 must be 1
                          !
                          !Bits 13:15
                          !  1 -> Equal        =
                          !  2 -> Not Equal    <>
                          !  3 -> Less Than    <
                          !  4 -> Greater Than >
Fixed   Value;            !Value used in comparison
Int     Rate;             !Use rated or nonrated value
                          !for condition checking:
                          !  0 -> No-Rate, 1 -> Rated
Int     SQLName[0:63];    !For SQLSTMTs only
Int     SQLIndex;         !SQL Index number
String  IP^addr[0:15] = SQLName;
Int     port = SQLIndex;
Int     IP^wildcard^flags = value;
End;

DEFINE  IP^addr^wildcard^flag = IP^wildcard^flags.<0>#,
        port^wildcard^flag = IP^wildcard^flags.<1>#;
```

item^name

input

is one of the counter names defined in the entity record (for example, CPU-BUSY-TIME for a CPU record).

relation^operator

input

is a value in the range 1 through 4, which designates comparison type:

1	Compare for equal.
2	Compare for unequal.
3	Compare for less-than.
4	Compare for greater-than.

value

input

is a 32-bit integer to be used for comparison with the value in the given counter. *value* = *n* * 1000. For example, 17.09 should be stored as 17090.

rate

input

determines whether a nonrated or rated value is used in the comparison:

- 0 indicates a nonrated value (similar to REPORT RATE OFF).
- 1 indicates a rated value (similar to REPORT RATE ON).

sqlname

input

is a 16-word array specifying the SQL run unit. *sqlindex* specifies the index number. These two fields are used in conjunction with the SQLSTMT entity. If passed, MEASWRITE_DIFF_ returns the records with the specified run unit and index numbers.

sqlindex

input

is the SQL index. A value of -1 specifies all SQL indexes.

NOTE: If the *if^item^def* structure is passed, it must first be initialized. To pass the structure without a value, IF^ITEM^DEF.Item^Name must be initialized to 0. Similarly a value of 0 in IF^ITEM^DEF.SQLName means that no SQL (*run-unit*,*index*) is specified.

IP^addr

input

is a bound or connected stream socket in the server.

templateversion

input

FIXED:value

(Measure G11 and later) is the appropriate template version literal from MEASDDLS for the entity type requested in *entitydesc*, if you want ZMS style external records. If omitted or passed as 0F, legacy style records are returned.

If passed as -1F, the *templateversion* for the current release will be used, starting with the H06.15/J06.04 RVUs. Note that returned external records, in that case, may not match the counter record definitions with which the application was compiled.

See TEMPLATE-VERSION (page 143) for further information on obtaining external records corresponding to a format that you can handle.

Usage Notes

- If the longer IF^ITEM^G09^DEF template is used, the G09^format must be set (see the DEFINE example). If the G09^format is not set, part of the *sqlname* field is interpreted as *sqlindex*, and any records returned are likely to be incorrect.

```
DEFINE G09^format       = relation^operator.0#;
DEFINE G09^rel^operator = relation^operator.13:15#;
```

- You can use either template format when accessing G09 data files. You can also use the longer template when accessing G08 and earlier data files if the actual *sqlname* value does not exceed 32 characters.

- In Measure H01 and later PVUs, MEASWRITE_DIFF_ accepts DISCOPEN, DISKFILE, FILE, and SQLSTMT entity descriptors for ANSI SQL objects or partitions.
- Measure H04, J02, and later PVUs will create a format 1 or format 2 structured file, depending upon the measurement data file size.

A Error Messages

The Measure subsystem produces error messages and error codes. The error messages generated by MEASCOM, the Measure command language interface, are described in this appendix. For descriptions of the error codes generated by Measure callable procedures, see Appendix B: Error Codes (page 471).

Each message in this appendix is preceded by MEAS and the message number. If you encounter an error while using MEASCOM, enter:

```
HELP error-number
```

MEAS 2000

```
Comment. For more information type HELP number.
```

Cause MEASCOM lets you get more information on a particular MEASURE message by entering HELP *number*.

Effect This message has no effect on the current processing.

Recovery None necessary; to suppress this comment, enter COMMENTS SUPPRESS 2000.

MEAS 3000

```
WARNING. System system-name does not exist; check system name.
```

Cause The indicated system was not known to the network.

Effect The command is executed, but the corresponding system number or name could not be added to the descriptor.

Recovery Check that you correctly indicated the desired system.

MEAS 3001

```
WARNING. Device dev-name does not exist; check spelling.
```

Cause The indicated device was not found or is not a valid device type for the command.

Effect The command is executed, but the device is not measured given the current state of the system.

Recovery Check that you correctly indicated the desired device or that the device is the correct type.

MEAS 3002

```
WARNING. Measurement was not started due to reason listed above.
```

Cause The START MEASUREMENT command failed at some point during its processing.

Effect Effect. The reason for failure is listed as an error returned from the callable procedures.

Recovery Enter HELP *error code number* to determine the appropriate corrections, and retry the operation. (For information on ALLOCATESEGMENT, see the *Guardian Procedure Calls Reference Manual*.)

MEAS 3003

```
WARNING. Measurement was not stopped due to reason listed above.
```

Cause The STOP MEASUREMENT command failed at some point during its processing.

Effect The reason for failure is listed as an error returned from the callable procedures.

Recovery Enter HELP *error code number* to determine the appropriate corrections, and retry the operation. (For information on ALLOCATESEGMENT, see the *Guardian Procedure Calls Reference Manual*.)

MEAS 3004

```
WARNING. Logging was not on so command had no effect.
```

Cause LOG STOP was issued, but logging was not in effect.

Effect The request is ignored.

Recovery Informative message only; no corrective action needed.

MEAS 3005

```
WARNING. Changing current plot file to current data file.
```

Cause ADD PLOT was issued, but the old data file associated with the plot was not the same as the one just listed from the current data file.

Effect The contents of the previous plot are lost.

Recovery Informative message only; no corrective action needed.

MEAS 3006

```
WARNING. Volume not network-accessible with current system default.
```

Cause A 7-character volume default was in effect with a remote system default.

Effect All file names expanded with the current defaults are invalid.

Recovery Change the system or volume default.

MEAS 3007

```
WARNING. No entities present to ADD; LIST records before trying ADD.
```

Cause An ADD PLOT was performed without any records available to be added.

Effect The request is ignored.

Recovery Perform a LIST *entity* command to obtain a list of the desired records before trying to add to the plot.

MEAS 3008

```
WARNING. Plot became full during ADD; some entities were not added.
```

Cause An attempt was made to add more than 26 items to a plot.

Effect Some of the requested items were not added.

Recovery Delete any unneeded items so there are 26 or fewer before adding the desired counters.

MEAS 3009

```
WARNING. Item not found -- item-name.
```

Cause The indicated plot item was not found or, if the plot character was also used in the command (for example, DELETE PLOT cpu-busy-time (A)), counter A was not an appropriate item name.

Effect The requested operation is ignored.

Recovery Check that you spelled the item correctly and entered the correct plot character.

MEAS 3010

```
WARNING. Measurement not found -- filename.
```

Cause The indicated data file was not found.

Effect The requested operation is ignored.

Recovery Check that you spelled the file name correctly and that the indicated file was added.

MEAS 3011

```
WARNING. Counter not found -- counter-name.
```

Cause The indicated counter was not found.

Effect The requested operation is ignored.

Recovery Check that you spelled the counter name correctly and that the indicated counter was added.

MEAS 3012

```
WARNING. Counter not found -- counter-name.
```

Cause The indicated counter was not found.

Effect The requested operation is ignored.

Recovery Check that you spelled the counter name correctly and that the indicated counter was added.

MEAS 3013

```
WARNING. Measurement in the current data file is from
current-datafile-version, which is a later release than this version
(current-meascom-version) of MEASCOM.
```

Cause Measurement in the current data file is from a later RVU than the version of the running MEASCOM. The later RVU might have changes in the external record format.

Effect The data file is added.

Recovery If you are certain there are no differences in the external record format between the two RVUs, you can disregard this warning message. Otherwise, use a MEASCOM from the same or later RVU of that in the data file.

MEAS 3014

```
WARNING. The specified Loadid has more than 5 characters, sequencing
will be suppressed.
```

Cause A 3-digit sequence-ID is added to LOADID to designate each interval of the measurement when LISTALL was used. It is suppressed if three characters are not available for use.

Effect Sequencing is suppressed.

Recovery Use five or fewer characters for LOADID.

MEAS 3015

```
WARNING. MEASCOM is attempting to allocate or resize its extended segment
to be number bytes.
```

Cause MEASCOM is requesting an extended segment larger than 16 MB.

Effect The command continues unless a system error is encountered, in which case the error is displayed.

Recovery If the segment is considered to be too large, stop MEASCOM, and perform the LIST in a way that decreases the number of records returned to MEASCOM. (For example, use the IF clause.)

MEAS 3016

```
WARNING. The item item-name is obsolete for the current datafile version;
results of the operation may be unexpected.
```

Cause The item name entered is no longer maintained in the data file being processed.

Effect The command continues, but the operation might produce unexpected results.

Recovery Use an item name that is valid for the PVU of the data file being processed.

MEAS 3017

```
WARNING. The specified file name was resolved using the current default
system (current-system-name), which is different from the system where
the data file was collected (system-name). Therefore, this command may
not LIST any records.
```

Cause The file name resolved in the LIST FILE command used the current default system, which is different from the system where the data file was collected.

Effect The command continues, but probably no data will be returned.

Recovery Enter the system name with the file name in the LIST command. For example, use LIST FILE \A.$B.C.D, or perform a SYSTEM \A and repeat the LIST command.

MEAS 3018

```
WARNING. TO and FOR are both set -- FOR will be ignored; if FOR is
desired then RESET TO.
```

Cause If displayed during a SHOW or SET command, the default attributes contain a value for both the TO and FOR values. If displayed during a LIST command, the default attributes plus the command line attributes caused the TO and FOR to both be set.

Effect The command continues, but the FOR option is ignored.

Recovery To use the TO option, no recovery is necessary. Otherwise, use the RESET command to clear the TO value so the FOR value is used.

MEAS 3019

```
WARNING. MEASCTL's current SWAP volume is volume; do NOT bring volume
down while MEASURE is still running.
```

Cause MEASCTL was started with a SWAP volume other than $SYSTEM. If MEASCTL's SWAP volume is unavailable at any time (for example, if a PUP DOWN operation is performed), the entire system halts.

Effect The command is executed.

Recovery No recovery is mandatory. But if disk space permits, stop the MEASURE subsystem and restart it using $SYSTEM (or a mirrored volume) as the SWAP volume. Use this command:

```
MEASCOM /SWAP $SYSTEM/ START MEASSUBSYS
```

MEAS 3020

```
WARNING. No current data file; select a current data file using ADD.
```

Cause The command executed required a current measurement data file to perform its function.

Effect The requested operation is ignored.

Recovery Perform an ADD MEASUREMENT on the data file desired.

MEAS 3021

```
WARNING. Measurement not active; unable to perform request.
```

Cause This command is valid only when the measurement is currently active.

Effect The requested operation is ignored.

Recovery Informative message only; no corrective action is needed.

MEAS 3022

```
WARNING. No more configuration data is available.
```

Cause No more entities are configured for the type requested.

Effect The requested operation is ignored.

Recovery Restart the search or change to another entity type.

MEAS 3023

```
WARNING. Crvsn was not specified, OSS file pathname translation may not
be correct.
```

Cause A Guardian file name was specified in a LISTPNAME command without specifying the CRVSN qualifier.

Effect Translation of the Guardian file name to an OSS file pathname is attempted, but the reported name might be incorrect if the file name specified is a transient entity.

Recovery Include the CRVSN value as reported in Measure entity reports.

MEAS 3024

```
WARNING. OSS Journal Segment is under construction, not attached.
```

Cause An ADD MEASUREMENT command was entered for a data file that was still in the process of OSS journal segment construction at measurement shutdown.

Effect The file is opened successfully, but the OSS journal segment is not attached.

Recovery Generally, no action is necessary. Translation can be processed from the OSS file-system name space on the current system. If you need access to the OSS journal segment at a later time, use the DELETE MEASUREMENT and ADD MEASUREMENT commands to add the Measure data file again after OSS journal segment construction is complete.

MEAS 3025

```
WARNING. Default Catalog not set.
```

Cause The SQLSCHEMA command has been issued without specifying a default SQL catalog and the default SQL catalog has not yet been set.

Effect Although the default SQL schema is set, if no SQLCATALOG command is issued, the ANSI names used in the subsequent commands might fail with an error.

Recovery Issue the SQLCATALOG command to set the default SQL catalog.

MEAS 3026

```
WARNING. Counter data records have been suppressed; active counter data
records are not available for PROCESSH.
```

Cause A START MEASUREMENT command was entered for a measurement that contains PROCESSH entities, and counter data record suppression was selected.

Effect The command continues, but PROCESSH records are not available for the measurement.

Recovery If PROCESSH records are desired, stop the measurement and restart it without counter data record suppression. If PROCESSH records are not desired, no corrective action is needed.

MEAS 3027

```
WARNING. Counter data records have been suppressed; OSS and/or SQL
journal segments will not be produced.
```

Cause A START MEASUREMENT command was entered for a measurement for which OSS and/or SQL journal segments were selected, and counter data record suppression was selected.

Effect The command continues, but OSS and/or SQL journal segments are not produced for the measurement.

Recovery If OSS and/or SQL journal segments are required, stop the measurement and restart it without counter data record suppression. If OSS and/or SQL journal segments are not required, no corrective action is needed.

MEAS 3050

```
Exceeded depth of 4 OBEY files; rework OBEY nesting.
```

Cause An attempt was made to include more than four OBEY files.

Effect The command is aborted, and all OBEY files are closed.

Recovery Rework the nesting of your OBEY files so no more than four files are used.

MEAS 3051

```
Expecting a name of nn characters or less; shorten name.
```

Cause The string entered must contain, at most, the number of characters indicated by the error message.

Effect The requested operation is ignored.

Recovery Shorten the string appropriately and retry the command.

MEAS 3052

```
File is not network-accessible; shorten volume name.
```

Cause A file name with a volume name containing more than seven characters was indicated for remote access.

Effect The requested operation is ignored.

Recovery For remote access, a volume name can contain no more than seven characters. Shorten the volume name accordingly.

MEAS 3053

```
Value out of range; value must be range.
```

Cause The number entered was out of the acceptable range for the current item.

Effect The requested operation is ignored.

Recovery Enter a value within the range indicated by the two values in the error message.

MEAS 3054

```
System does not exist; check system name.
```

Cause The system indicated in the request does not exist.

Effect The requested operation is ignored because the file name cannot be expanded.

Recovery Check that you spelled the system name correctly.

MEAS 3055

```
Illegal syntax; see HELP or the Measure Reference Manual.
```

Cause The parser encountered a syntax error.

Effect The requested operation is ignored.

Recovery Correct the syntax according to HELP or the syntax in this manual.

For command syntax descriptions, see Chapter 2: MEASCOM Commands (page 37). For descriptions of syntax for identifying entities to be measured, see Chapter 3: Entities and Counters (page 133).

MEAS 3056

```
Subsystem error; please report to Tandem Computers Incorporated. Trap
%n, CS = %n, P = %nnnnn, E = %nnnnn, L = %nnnnn, S = %nnnnn.
```

Cause A trap was encountered by MEASCOM.

Effect The requested operation is ignored.

Recovery Retain a copy of the information provided in the error message and contact your HP representative.

MEAS 3057

```
Ambiguous DELETE command; specify which occurrence to delete.
```

Cause A DELETE PLOT was requested where the counter name entered occurs more than once in the plot definition.

Effect The requested operation is ignored.

Recovery Repeat the command and add either the plot character to indicate a particular occurrence or an asterisk (*) to indicate all occurrences.

MEAS 3058

```
Expecting a valid process or file name.
```

Cause MEASCOM was expecting either a valid process name (for example, $XMM) or a valid file name (for example, $*disk.subvol.file*, or $*disk.subvol.**).

Effect The requested operation is ignored.

Recovery Reissue the request with a valid process or file name.

MEAS 3059

```
Only an individual file can be specified; replace *'s.
```

Cause A single file name was not entered for the command.

Effect The requested operation is ignored.

Recovery Replace the wild-card character (*) with a specific file name.

MEAS 3060

```
Expecting a disk file name; ensure filename is a valid disk file name
```

Cause A permanent disk file was not specified in the command.

Effect The requested operation is ignored.

Recovery Ensure the name entered is of the form $*a.b.c*.

MEAS 3061

```
The USERDEF for this counter was not found; ADD the USERDEF.
```

Cause The process specification entered in the current ADD COUNTER command was not previously added by an ADD USERDEF command.

Effect The requested operation is ignored.

Recovery Check that the process specification is correct, or add the USERDEF and retry the operation.

MEAS 3062

```
Counter already exists; DELETE and then ADD to change counter.
```

Cause An attempt was made to add a counter that already exists.

Effect The requested operation is ignored.

Recovery To change the counter type or array index, delete the counter and then add it.

MEAS 3063

```
No more data files can be added; DELETE unnecessary files.
```

Cause An attempt was made to add more than 64 data files.

Effect The requested operation is ignored.

Recovery Delete any unnecessary data files and then add the ones desired.

MEAS 3064

```
Expecting a disk file name; ensure file is a valid disk filename.
```

Cause A valid or unique disk file was not specified in the command.

Effect The requested operation is ignored.

Recovery Ensure that the name entered is of the form $*a.b.c* or $*a.#x*.

MEAS 3065

```
File must be local; see Measure Reference Manual.
```

Cause The file name given or expanded was remote, and the command required a local file name.

Effect The requested operation is ignored.

Recovery For a discussion of file names that are used with the command, see the description of the specific command in Chapter 2: MEASCOM Commands (page 37). Then specify a local file and retry the operation.

MEAS 3066

```
The name specified in the IF condition (item-name) was not found; check
the spelling of the name.
```

Cause The item name used in the IF condition was not an item name associated with the entity type being listed.

Effect The requested operation is ignored.

Recovery Check the spelling of the item and verify the item name exists, then retry the operation.

MEAS 3067

```
The name specified in the BY clause (item-name) was not found; check
the spelling of the name.
```

Cause The item name used in the BY condition was not an item name associated with the entity type being listed.

Effect The requested operation is ignored.

Recovery Check the spelling of the item and verify the item name exists, then retry the operation.

MEAS 3068

```
No current data file; select a current data file using ADD.
```

Cause The command executed required a current measurement data file to perform its function.

Effect The requested operation is ignored.

Recovery Perform an ADD MEASUREMENT on the data file desired.

MEAS 3069

```
Command option has been repeated; delete one and reissue command.
```

Cause An option was specified more than once in the same command.

Effect The requested operation is ignored.

Recovery Delete all but one occurrence of the option and repeat the command.

MEAS 3070

```
Both FOR and TO have been specified; delete one and reissue command.
```

Cause Both the FOR and TO option were specified in the same command. These options are mutually exclusive.

Effect The requested operation is ignored.

Recovery Delete one of the options and repeat the command.

MEAS 3071

```
item-name not found; check the spelling and/or the appropriate entity
report.
```

Cause The plot counter name entered was not found.

Effect The requested operation is ignored.

Recovery Check the spelling and verify, using the appropriate report, that the name entered is a valid item name. Legacy style report items that are no longer present in the equivalent ZMS style report can only be plotted in the legacy style interface. And ZMS style report items that are no longer present in the equivalent legacy style report can only be plotted in the ZMS style interface.

MEAS 3072

```
Measurement not active; unable to perform request.
```

Cause An attempt was made to issue a command that is valid only when the measurement is currently active.

Effect The requested operation is ignored.

Recovery Informative message only; no corrective action is needed.

MEAS 3073

```
A remote filename is not allowed when dealing with DISCOPENs or
DISKFILEs.
```

Cause The file name entered or expanded (using the current SYSTEM) for a DISCOPEN or DISKFILE command was for a remote file.

Effect The requested operation is ignored.

Recovery DISCOPENs and DISKFILEs are always local. For DISCOPENs, a remote opener can be specified after the file name (for example, $a.b.c (\sys)). For DISKFILE, no opener information is associated with the entity, so using a remote opener is not allowed.

MEAS 3074

```
Plot list is full; must DELETE an item before ADD.
```

Cause An attempt was made to add to an already full plot.

Effect The requested operation is ignored.

Recovery Delete some counters and then retry the ADD.

MEAS 3075

```
Expecting a single alpha character; see HELP.
```

Cause The plot character was not a single alphabetic character.

Effect The requested operation is ignored.

Recovery Enter only a single alphabetic character. For more information, enter HELP.

MEAS 3079

```
Measurement space overflow; DELETE unneeded descriptors.
```

Cause The measurement space became full during an ADD.

Effect The requested operation is ignored.

Recovery Delete unneeded descriptors from the measurement space and retry the operation.

MEAS 3080

```
Allocate segment error number on work segment.
```

Cause An error was encountered and returned from the allocate segment on the MEASCOM primary work segment.

Effect The requested operation is ignored.

Recovery MEASCOM can run without the segment, but it can perform in only a limited capacity. Correct the problem encountered with the allocate segment if possible and rerun MEASCOM.

MEAS 3081

```
The IF clause on an identifier field (i.e. name) is not supported for
the current datafile version.
```

Cause Use of the IF clause on a noncounter item is not supported for data files prior to D10.

Effect The requested operation is aborted.

Recovery Use an item name that corresponds to a counter item, or delete the IF clause.

MEAS 3083

```
Internal buffer full; try making command less complex.
```

Cause When Measure expanded the command, a buffer became full.

Effect The requested command is ignored.

Recovery Break the command into several commands and execute the commands separately, or qualify any file names present so MEASCOM performs a minimum of expansion.

MEAS 3084

```
The DISCOPEN and DISKFILE entities require the filename to be either a
permanent or temporary disk filename (e.g. $a.b.* or $system.#*).
```

Cause The current file name is not a disk file name

Effect The requested command is not executed.

Recovery Change the file name entered to a valid permanent or temporary disk file name.

MEAS 3085

```
"char" not allowed in current string; delete symbol and retry.
```

Cause The symbol displayed by the error message is not a valid word separator for the string entered.

Effect The requested operation is ignored.

Recovery Either remove the character or replace it with a valid word separator, and then retry the command.

MEAS 3086

```
START attempted on an ADDed datafile; DELETE and then START.
```

Cause START was attempted on a data file that was added for listing. Because the START command causes the data to be purged, the command is aborted. The commands START, STOP, or ADD cause the data file to be added in the MEASCOM environment. This message is an attempt to prevent accidental purging of data.

Effect The requested operation is aborted.

Recovery If you still need the START operation, DELETE the data file and retry the START.

MEAS 3087

```
Expecting Help key length of 30 characters or less; shorten string.
```

Cause The key string entered for the Help command exceeds 30 characters.

Effect The requested operation is not executed.

Recovery Shorten the string.

MEAS 3089

```
Expecting a number with 3 or less decimal digits.
```

Cause The number entered has more than three decimal digits.

Effect The requested operation is not executed.

Recovery Reduce the number of decimal digits, then reenter the command.

MEAS 3090

```
Incompatible PROCESSH records in the filename data file. Please use a
proper version of MEASCOM.
```

Cause The PROCESSH record format in the data file is incompatible with the MEASCOM currently running.

Effect The requested operation is not executed.

Recovery Obtain a proper version of MEASCOM.

MEAS 3091

```
The existing DDL format structured file for the entity specified in the
LIST command has a different record size.
```

Cause The record size is different from the size in the data file.

Effect The requested operation is not executed.

Recovery Remove the existing structured file and let MEASCOM create it, or use a different subvolume. Be sure to regenerate the corresponding DDL dictionary.

MEAS 3092

```
SYSTEM-PROCESSES is not allowed in this command.
```

Cause SYSTEM-PROCESSES is not allowed in SQLPROC and SQLSTMT commands.

Effect The requested operation is not performed.

Recovery Change the command to a suitable format and retry.

MEAS 3094

```
Incompatible datafile version. The LISTALL command is not supported for
this version of the datafile.
```

Cause LISTALL command was issued on a C10 or previous data file.

Effect Command aborts; specified MEASFH cannot satisfy the request.

Recovery Try another command. To retrieve records, use the LIST command.

MEAS 3095

```
Command option "report-option" not allowed with LISTALL command when
STRUCTURED format is specified.
```

Cause Illegal *report-option* was specified with LISTALL.

Effect Command aborts; specified option is not satisfied.

Recovery Retry the command, and do not include the *report-option* causing the error.

MEAS 3096

```
Command option "TOLERANCE OFF" not allowed with LIST command.
```

Cause The TOLERANCE OFF option was specified on a C10 or previous data file.

Effect Command aborts; MEASFH cannot satisfy request.

Recovery Retry the command, and do not include the TOLERANCE option.

MEAS 3097

```
Error number was reported earlier on this data file. Operation cannot
be carried out.
```

Cause When data file was added, MEASFH reported an error or warning. Problems might exist with the data file.

Effect Command aborts; command could not be carried out correctly.

Recovery Try a different command on this data file.

MEAS 3098

```
FC and ! are not allowed on input longer than 132 characters; reenter
the desired command in full.
```

Cause Although an effective record can contain up to 2100 characters, only effective records of 132 characters or fewer are candidates for modification using the FC command or for reexecution using the ! command.

Effect No command is executed.

Recovery The desired command must be entered in full (use the editor and the OBEY command to execute long commands).

MEAS 3099

```
Line specified is not in the HISTORY buffer; check buffer and retry.
```

Cause The line that Measure is attempting to be reexecute is not in the HISTORY buffer.

Effect No command is executed.

Recovery Perform a HISTORY command to make sure the command is still in the HISTORY buffer. Otherwise you must reenter the command in full.

MEAS 3100

```
The command is too long, the maximum number of characters an effective
record can contain is 2100 (nnnn were entered); try to enter multiple
commands or remove embedded spaces as possible to shorten the command(s)
entered.
```

Cause Even with the use of the ampersand (&), there is a limit to the number of characters that can be entered in one record (for example, before MEASCOM scans the record and executes the commands). The current input stream exceeded the limit of 2100 characters.

Effect The command is not executed.

Recovery If more than one command was entered, try entering each command separately. If only one command was entered, try breaking the command into several commands and execute the commands separately. If the command cannot be decomposed, try removing some embedded spaces to make the command shorter.

MEAS 3101

```
filename was not started due to Process_Create_ error number; correct
error and retry command.
```

Cause The program file indicated was not started due to the Process_Create_ error indicated.

Effect The command is not executed.

Recovery Correct the cause of the Process_Create_ error and retry the command.

MEAS 3102

```
filename could not be opened due to File System error number.
```

Cause The program file indicated could not be opened due to the file-system error indicated.

Effect The command is not executed.

Recovery Correct the cause of the file-system error and retry the command.

MEAS 3103

`filename did not read the startup message due to File System error number.`

Cause The program file indicated did not read the startup message sent due to the file-system error indicated.

Effect The command is not executed.

Recovery Correct the cause of the file-system error and retry the command.

MEAS 3104

`SCALE-FROM must be less than SCALE-TO; make sure the values are correct before retrying the LIST PLOT command.`

Cause The effective value for the SCALE-FROM attribute was greater than or equal to the SCALE-TO attribute of the PLOT object.

Effect The command is not executed.

Recovery Perform a SHOW PLOT to determine the default values for each attribute. Any attribute entered on the LIST PLOT command line overrides the default value. Ensure the effective values are consistent when you retry the command.

MEAS 3105

`Expecting one of the following keywords keyword-list.`

Cause The word encountered is not a valid keyword.

Effect The command is not executed.

Recovery Enter a valid keyword (some are displayed in the error text, but not all if the list is too long).

MEAS 3106

`item-name is a valid item name, but it can only be used in the BY clause; use a different item name, or a counter name that appears within the entity report.`

Cause An item name was used in the IF clause that can only be used in the BY clause.

Effect The command is not executed.

Recovery Use a different item name, or use a counter name that appears within the entity report.

MEAS 3107

`item-name is a valid item name, but it can only be used in the By and IF clauses; use a counter name that appears within the entity report.`

Cause An item name was used in a PLOT command that can only be used in the BY and IF clauses.

Effect The command is not executed.

Recovery Use a counter name that appears within the entity report.

MEAS 3108

`item-name is not a unique item-name; do a HELP entity-type COUNTERS (e.g. HELP CPU COUNTERS) to determine the number of characters required to make the abbreviation unique.`

Cause The item name entered matched more than one item name.

Effect The command is not executed.

Recovery Add more characters to the name entered.

MEAS 3109

`item-name is a new item name that is not valid for the current datafile version; use a different item name.`

Cause An item name was entered that is not contained in the (down-rev) data file being processed.

Effect The command is not executed.

Recovery Use a different item name.

MEAS 3110

`The current string of digits contains a digit which is greater than the current base of number; delete the digit and retry the command.`

Cause The digit entered is too large for the current base (for example, ADD CPU %8 is incorrect because an octal digit must be 7 or under).

Effect The command is not executed.

Recovery Change the number entered to be consistent with the base you are using.

MEAS 3111

`Native Mode code maps do not have space numbers; if you wish to sample native mode maps specify either UCr, ULr, SCr, or SLr.`

Cause The code map specified is incorrect for native mode code maps.

Effect The command is not executed.

Recovery To measure native mode code, specify UCr, ULr, SCr, or SLr.

MEAS 3112

`The datafile version is` *`version`* `and cannot be analyzed with F40 Meascom.`

Cause When you are using a G-series RVU, the version of the data file precedes F40.00.

Effect The command is not executed.

Recovery To analyze a pre-F40.00 data file, use a pre-F40.00 version of MEASCOM that matches the version of the data file as well as the appropriate MEASFH.

MEAS 3113

`Unable to open data file.`

Cause MEASFH was unable to open the data file because there was too much data for the system to handle (file-system error 21).

Effect The command is not executed.

Recovery Retry the command.

MEAS 3114

`A journal segment is required but not available for request.`

Cause A command was entered for a data file that was not collected on the current node, and for which no journal segment is currently available.

Effect The command is not executed.

Recovery If the translation desired is NOT from the added data file, then DELETE the measurement and reissue the command so that the translation is done from the current system. If translation is desired for data collected on another node, specify the OSS and/or SQL option when starting the measurement.

MEAS 3115

`PAGESIZE must be specified in the range of 6 and 128.`

Cause An invalid PAGESIZE setting was specified.

Effect The requested operation is aborted.

Recovery Specify a value from 6 through 128.

MEAS 3116

```
Unable to translate the specified OSS file pathname in the current
context.
```

Cause The requested translation was not found in the OSS name space or in the current OSS journal segment.

Effect The requested operation is aborted.

Recovery If the request is part of an ADD data file that contains an OSS journal segment, you can DELETE the data file and attempt translation from the current system if it is part of the same ADD data file.

MEAS 3117

```
Unable to translate an ANSI SQL name.
```

Cause The requested translation was not found in the ANSI SQL name space or in the current ANSI SQL journal segment.

Effect The command is not executed.

Recovery If within the context of an ADD data file that contains an ANSI SQL journal segment, delete the data file and attempt translation from the current system if it is the same as that of the data file. For the command LISTACTIVE DISKFILE, make sure that a partition is specified. Else there is no recovery possible.

MEAS 3118

```
Syntax errors in specified ANSI SQL name.
```

Cause The ANSI SQL name contained syntax errors.

Effect The requested operation is aborted.

Recovery Reenter command with correctly formed ANSI SQL name.

MEAS 3121

```
Default Catalog or Schema missing.
```

Cause There is not enough information available to create a fully qualified SQL ANSI name. Either the default SQL catalog or the default SQL schema or both are missing.

Effect The command is not executed.

Recovery Issue the ENV command to see what the settings are for default SQL catalog and SQL schema. Issue the appropriate SQLCATALOG and/or SQLSCHEMA command(s) to add the missing default setting(s).

MEAS 3122

```
The data file format does not support CLIM devices.
```

Cause A LIST command was given for the DISC or DEVICE entity that specified a CLIM device or a path selection on a pre-H03 data file.

Effect The command is not executed.

Recovery Change the LIST command to not specify a CLIM device, path selection or lun.

MEAS 3175

```
System is not TANDEM NonStop; see your system manager.
```

Cause An attempt was made to run MEASCOM on a system other than a NonStop server.

Effect MEASCOM stops abruptly.

Recovery See your system manager.

MEAS 3176

`System option number is not T2035; see your system manager.`

Cause An attempt was made to run MEASCOM with an option number other than T2035.

Effect MEASCOM stops abruptly.

Recovery See your system manager.

MEAS 3177

`Guardian version is not B30 or greater; see your system manager.`

Cause The NonStop operating system PVU must be B30 or later.

Effect MEASCOM stops abruptly.

Recovery See your system manager.

MEAS 3178

`Internal consistency check` *`nnn`* `failed; report error.`

Cause The internal check *nnn* failed.

Effect MEASCOM stops abruptly.

Recovery If the command involves a great deal of file-name expansion, rerun MEASCOM and try to perform the same command in several steps. If this attempt fails or does not address the original problem, contact your HP representative.

B Error Codes

The Measure subsystem produces error messages and error codes. The error codes generated by Measure callable procedures are described in this appendix. For descriptions of the error messages generated by MEASCOM, the Measure command interface, see Appendix A: Error Messages (page 455).

3200

(ERR^BADMEASCB)

Cause The MEASCB was not set to -1s on the initial procedure call, or its contents were modified after being set by Measure.

Effect The requested operation aborts.

Recovery Initialize MEASCB to all -1s prior to your first Measure call and do not write to it afterwards.

3201

(ERR^MISSINGPARAM)

Cause A required parameter was missing.

Effect The requested operation aborts.

Recovery Add the missing parameter to your procedure call statement.

3202

(ERR^BOUNDS)

Cause An invalid parameter address was supplied.

Effect The requested operation aborts.

Recovery Check and correct the address initialization of your parameters.

3203

(ERR^BADPARAMS)

Cause A bad parameter value was supplied to a callable procedure.

Effect The requested operation aborts.

Recovery Check and correct the values of your parameters.

3204

(ERR^BUFTOOSMALL)

Cause The buffer size was too small.

Effect The requested operation aborts.

Recovery Correct the buffer size. The buffer size needed is contained in the *bytesret* (byte size of record returned) parameter. If MEASREADACTIVE encountered this error and requires a buffer larger than 32 KB, use the MEAS_READACTIVE_ Procedure.

3205

(ERR^NOTENOUGHSTACK)

Cause Not enough data-stack space was available to send or receive the buffer.

Effect The requested operation aborts.

Recovery Move some of the stack data into an extended segment to increase the size of the available working stack.

3206

(ERR^BADREPLY)

Cause An unexpected reply was received from a Measure process.

Effect The requested operation aborts.

Recovery Retain a copy of the files and commands needed to duplicate this error and contact your HP representative.

3207

(ERR^NEWDESCFIELD)

Cause A down-rev MEASFH being used cannot support the new field used in the descriptor.

Effect The requested operation aborts.

Recovery Reissue the request without using the new field. In MEASCOM, avoid the new fields. For example, change LIST FILE * ($PNAME) to LIST FILE * (1,52), assuming 1,52 was $PNAME. If you are using the programmatic interface directly, use the old version of the descriptor, or place only the appropriate default values (for example, $*) in the new fields.

3208

(ERR^WRONGSYSTEM)

Cause The MEASFH process, which indexes and builds structured records from the data file, and the data file are located on different systems.

Effect The requested operation aborts.

Recovery Move either the MEASFH program file or the data file so they are both on the same node. Only read access is allowed for remote data files.

3209

(ERR^REMOTEDFILE)

Cause The data file was not on the same system as the calling process.

Effect The requested operation aborts.

Recovery Because write access is not allowed for remote data files, use a data file on the same system as the calling process.

3210

(ERR^BADDFNUM)

Cause The data file access number (DFNUM) parameter value did not match a valid DFNUM returned by a successful MEASOPEN call.

Effect The requested operation aborts.

Recovery Check that the value used is the same as the value returned by a MEASOPEN procedure.

3211

(ERR^BADMEASNUM)

Cause The measurement number (MEASNUM) parameter value did not match a valid MEASNUM returned by a successful MEASCONFIGURE call.

Effect The requested operation aborts.

Recovery Check that the value used is the same as the value returned by a MEASCONFIGURE procedure.

3212

(ERR^MEASNOTACTIVE)

Cause Measurement was not configured and active.

Effect The requested operation aborts.

Recovery Check that the measurement is configured and started, and that it did not stop during the operation.

3213

(ERR^NOMEASCTL)

Cause A CPU was up, but there was no MEASCTL to allocate and maintain counters in the CPU.

Effect The requested operation aborts.

Recovery Use the MEASMONCONTROL procedure, indicating the CPU number of the missing MEASCTL.

3214

(ERR^MEMLOCKFAILURE)

Cause Insufficient physical memory is available.

Effect The requested operation aborts.

Recovery Redistribute your processes to other processors or install more memory.

3215

(ERR^LINK)

Cause A call to LINK failed.

Effect The requested operation aborts.

Recovery Retry the operation.

3216

(ERR^NOTAVAIL)

Cause PROCESSH, DISCOPEN, and DISKFILE records are not available from MEASREADACTIVE.

Effect The requested operation aborts.

Recovery Call MEASREAD to obtain these records.

3217

(ERR^NOTSUPER)

Cause The attempted call requires a super-group user ID (255,*n*).

Effect The requested operation aborts.

Recovery Have a super-group user execute the process that calls this procedure.

3218

(ERR^MONALREADYALIVE)

Cause The Measure subsystem was already started and cannot be started again.

Effect The requested operation aborts.

Recovery Verify that the Measure subsystem is not running prior to calling MEASMONCONTROL with a start subsystem request.

3219

(ERR^NOMEASMON)

Cause The Measure subsystem was stopped.

Effect The requested operation aborts.

Recovery Verify that the Measure subsystem is running prior to calling this procedure.

3220

(ERR^UDCNOTPRESENT)

Cause A user-defined counter was not installed and cannot be bumped.

Effect The requested operation aborts.

Recovery Prior to this call, configure a measurement that includes the user-defined counters to be bumped.

3221

(ERR^UDCBADBUMP)

Cause An invalid BUMPTYPE was used for a user-defined counter.

Effect The requested operation aborts.

Recovery Check and correct the value of the BUMPTYPE parameter.

3222

(ERR^UDCBADINDEX)

Cause An invalid INDEX was used for a user-defined counter.

Effect The requested operation aborts.

Recovery Check and correct the value of the INDEX parameter.

3223

(ERR^INVALIDENTITY)

Cause A new or invalid entity type was supplied.

Effect The requested operation aborts.

Recovery Check that the entity type is valid and that it is supported in the RVU of the data file being processed.

3224

(ERR^INCONSISTENCY)

Cause MEASCTL internal consistency check failed.

Effect SQL activities are not measured.

Recovery Contact your HP representative.

3225

(ERR^INCOMPLETEMSG)

Cause The callable procedure was unable to complete a message to the MEASCTL process.

Effect The requested operation was ignored.

Recovery Try the operation again. If the operation continues to fail, contact your HP representative.

3226

(ERR^LOCALSYSNAME)

Cause The callable procedure cannot retrieve the caller's local system name from Nodenumber_to_Nodename_.

Effect The requested operation was ignored.

Recovery Try the operation again. If the operation continues to fail, contact your HP representative.

3227

(ERR^MEASFH^INVALID^NAME)

Cause Measure encountered an internal error while working on the MEASFH file name.

Effect The requested operation aborted.

Recovery Check that the MEASFH object file exists and is properly secured.

3228

(ERR^MEASFH^CREATION)

Cause An error was returned from Process_Create_.

Effect The requested operation aborted.

Recovery Check that the MEASFH object file exists and is properly secured.

3229

(ERR^MEASFH^OPEN)

Cause An error was returned from File_Open_.

Effect Unable to open the MEASFH process. The requested operation aborted.

Recovery Check that the MEASFH object file exists and is properly secured.

3230

(ERR^MEASFH^SWAPFILE)

Cause The SWAPVOL value specified was in error.

Effect The MEASFH process or swap file is not created; the requested operation aborts.

Recovery Use a different SWAPVOL value, or clear the SWAPVOL and use the default value.

3231

(ERR^CONFIG^NOT^AVAIL)

Cause You specified an unsupported entity type to the MEASLISTCONFIG procedure. The procedure supports only the DEVICE, LINE, NETLINE, OPDISK, and SERVERNET entities.

Effect The requested operation aborts.

Recovery Issue the request again, specifying a supported entity type.

3232

(ERR^NOTENOUGH^PFSBUFFER) (ERR^NOTENOUGH^FPMEMORY)

Cause The API was unable to allocate enough global buffer space from the flexible memory pool of the process.

Effect The operation is not performed.

Recovery Ensure that the application does not exhaust its flexible pool memory.

3234

(ERR^CRVSNNOTSPECIFIED)

Cause You called MEASLISTPNAME without specifying the CRVSN qualifier.

Effect Translation of the Guardian fielname to an OSS file pathname is attempted, but the reported name might be incorrect if the file name specified is a transient entity.

Recovery Include the CRVSN value in MEASLISTPNAME calls.

3235

(ERR^BUILDINGOSSJOURNAL)

Cause MEASOPEN was called for a data file that was still in the process of OSS journal segment construction at measurement shutdown.

Effect The file is opened successfully, but the OSS journal segment is not attached.

Recovery Generally, no action is necessary. Translation can be processed from the OSS file system name space on the current system. If you need access to the OSS journal segment at a later time, the program must close and reopen the Measure data file after OSS journal segment construction is complete. You might notice that data access is slower for files opened during journal segment construction even after the journal segment construction step is completed. Closing and reopening the file improves the data-access speed.

3236

(ERR^MISSINGOSSJOURNAL)

Cause A request was made that required an OSS journal segment for translation, but no OSS journal segment is attached in the current Measure data file context.

Effect The requested operation is aborted.

Recovery If you need translation from the OSS file-system name server, omit the *dfnum* parameter or pass -1 as its value. If you need translation from a journal segment, ensure that one was collected with the data file and that *err^buildingossjournal* was not returned on the MEASOPEN request.

3237

(ERR^OSSDIRNOTALLOWED)

Cause A DISKFILE command contained an OSS directory name.

Effect The requested operation is aborted.

Recovery Only fully qualified OSS file pathnames are allowed in DISKFILE entity specification. If you need transient file activity or specific sets of files, use Guardian file naming. For example: $DATA.*.*, or $*.ZYQ00000.*.

3238

(ERR^UNKNOWNSQLNAME)

Cause The specified ANSI SQL name could not be translated, or the specified Guardian name could not be translated to an ANSI SQL name.

Effect The requested operation is aborted.

Recovery The name is no longer current in the ANSI SQL name space or is not present in the SQL journal.

3239

(ERR^BADFORMATSQLNAME)

Cause The specified ANSI SQL name could not be parsed due to syntax errors or was not fully qualified.

Effect The requested operation aborts.

Recovery In the entity specification syntax, check that the ANSI SQL name is syntactically correct and, if necessary, fully qualified.

3240

(ERR^BUILDINGSQLJOURNAL)

Cause MEASOPEN was called for a data file that was still in the process of SQL journal segment construction when the measurement shut down.

Effect The data file opens successfully, but the SQL journal segment is not attached.

Recovery Generally, no action is necessary. Translations are processed through the SQL/MX services. If you later need access to the SQL journal segment, close and reopen the data file after SQL journal segment construction completes.

3241

(ERR^MISSING^SQLJOURNAL)

Cause A request was made that required the SQL journal segment for translation, but no SQL journal segment is attached in the current Measure data file context.

Effect The requested operation aborts.

Recovery If you require translation from the ANSI SQL name server, omit the *dfnum* parameter or pass -1 as its value. If you require translation from a journal segment, ensure that you requested such a segment and that the MEASOPEN request did not return the error err^buildingsqljournal.

3242

(ERR^NOT^IMPLEMENTED)

Cause A request was made for Measure functionality that does not exist in this version of Measure.

Effect The requested Measure function is not performed.

Recovery Upgrade your version of Measure to the required level for the specified function.

3243

Err^nospaceossjournal

Cause Disc space was not sufficient or segment allocation failed for building OSS or ANSI SQL journal segment.

Effect File is opened successfully, but journal segment is not attached.

Recovery OSS name translation requests will be processed from the OSS file system name server on the current system. ANSI SQL names translation will be processed via SQL/MX on the current system. If access to the OSS or SQL journal segment is desired then start a measurement on a disk having enough space.

3244

(ERR^NO^TIMERCELL^CONTEXT)

Cause The application called a timercell function, but no timercell context is currently allocated. For example, the application called another timercell function without first invoking MEAS_ALLOCATE_TIMERCELLS_. Alternatively, all timercells have been deallocated, or the context was corrupt (as indicated by error code 3245).

Effect The requested operation is not performed.

Recovery Call MEAS_ALLOCATE_TIMERCELLS_ before invoking other timercell functions.

3245

(ERR^BAD^TIMERCELL^CONTEXT)

Cause The application called a timercell function, but the timercell context is corrupted.

Effect The context is cleared, and all allocated timer cells for this process are lost.

Recovery Call MEAS_ALLOCATE_TIMERCELLS_ to create a new timercell context before invoking other timercell functions.

3246

(ERR^NOT^ALLOCATED)

Cause The resource specified as the target of a callable function is not allocated.

Effect The operation is not performed.

Recovery Call whatever procedure allocates the resource before you call any procedure that requires the resource.

3247

(WARN^TIMERCELL^ACCURACY)

Cause The application attempted to retrieve more than 128 timer cells, and because of insufficient system resources, Measure could not adjust the reported values of some counters to reflect the advance of time for repeated rendezvous operations.

Effect Data is returned for all timers, but timers in the penultimate group (modulo 128) will have returned values slightly less than their true values.

Recovery Retrieve fewer than 128 timer cells at a time.

3248

(ERR^NO^MORE^TIMERCELLS)

Cause The application attempted to allocate timer cells, but timercell pool for the processor was insufficient to satisfy the request.

Effect The operation does not occur.

Recovery Use the timer cells that have already been allocated. Alternatively, deallocate some existing timer cells, and allocate new ones.

3251

(ERR^BADDESC)

Cause The CONTAB format, which contains the entity descriptors to be measured, was incorrect.

Effect The requested operation aborts.

Recovery Correct any bad or missing CONTAB header, entity descriptor, or trailer record. Examine the CONTAB prior to the call.

3252

(ERR^FILECODE)

Cause The data file did not have a file code of 175.

Effect The requested operation aborts.

Recovery Purge the file and create a new file with a file code of 175.

3253

(ERR^NOTDISCFILE)

Cause The data file was not a disk file.

Effect The requested operation aborts.

Recovery Copy the measurement data file from tape to disk prior to read access.

3254

(ERR^WRONGVERSION)

Cause The MEASFH version did not match the data file version.

Effect The requested operation aborts.

Recovery Specify the MEASFH program file name from the RVU used to generate the data file.

3255

(ERR^FILEINUSE)

Cause An attempt was made to start a measurement on a measurement data file that was already open.

Effect The requested operation aborts.

Recovery Close the file and retry the operation.

3256

(ERR^ISEGOVERFLOW)

Cause The MEASFH process received an error while attempting to sort a data file index segment.

Effect The requested operation aborts.

Recovery Close the file and retry the operation.

3257

(ERR^RSEGOVERFLOW)

Cause The datafile record address array segment overflowed.

Effect The requested operation aborts.

Recovery Try reducing the number of items to be LISTed by requesting specific items rather than using the asterisk (*) to list all items. For the PROCESSH entity, try specifying `if code-range > 0` in the LIST command.

3258

(ERR^CSEGOVERFLOW)

Cause MEASFH's external counter record segment overflowed.

Effect The requested operation aborts.

Recovery Try reducing the number of items to be LISTed by requesting specific items rather than using the asterisk (*) to list all items. For the PROCESSH entity, try specifying `if code-range > 0` in the LIST command.

3259

(ERR^MEASNOTDONE)

Cause The measurement was not complete.

Effect The requested operation aborts.

Recovery Stop the measurement and retry the operation.

3260

(ERR^NOOPEN)

Cause The data file was not open.

Effect The requested operation aborts.

Recovery Call MEASOPEN for the data file and retry the operation that failed.

3261

(ERR^INTTOOSMALL)

Cause The time interval was too small.

Effect The requested operation aborts.

Recovery Retry the operation with a time interval equal to or greater than one second.

3262

(ERR^NOCTRREC)

Cause A counter record for the requested entity could not be found.

Effect The requested operation aborts.

Recovery Inspect the measurement CONTAB parameter containing the entity descriptors to be measured. Check for an entity descriptor that includes the current descriptor and verify that this entity was alive during the measurement.

3263

(ERR^CRFILETYPE)

Cause The PROCESSH code-range file was not a disk file.

Effect The requested operation aborts.

Recovery Move the code-range file to disk.

3264

(ERR^CRFILECODE)

Cause The PROCESSH code-range file was not an object or edit file.

Effect The requested operation aborts.

Recovery Move the code ranges to a file in edit format, or use an object-format file to specify the code ranges.

3265

(ERR^CRFILE)

Cause A PROCESSH code-range file open or read error occurred.

Effect The requested operation aborts.

Recovery Check the file name and verify that the file can be read (FUP COPY). Make any needed changes and retry the operation.

3266

(ERR^CRFMT)

Cause There was an error in the format of the PROCESSH code-range file.

Effect The requested operation aborts.

Recovery Correct the format and retry the operation.

3267

(ERR^NOCRS)

Cause The PROCESSH code-range file did not specify any code ranges, or the existing code ranges do not occur in the specified code space.

Effect The requested operation aborts.

Recovery Add at least one code range to the file or correct the code space specification and retry the operation.

3268

(ERR^TOOMANYCRS)

Cause The PROCESSH code-range file specified more than 500 code ranges.

Effect The requested operation aborts.

Recovery Delete enough code ranges from the file so that there are 500 or fewer and then retry the operation.

3269

(ERR^CRTABLES)

Cause The PROCESSH code-range table overflowed. Either the data file has been corrupted, or there is a problem with MEASFH.

Effect The requested operation aborts.

Recovery If the data file was modified since the measurement was done, repeat the measurement and then retry the operation. Otherwise, retain the data file and contact your HP representative.

3270

(ERR^EMPTYDATAFILE)

Cause Read access was requested for a data file that contains no data.

Effect The requested operation aborts.

Recovery Either use another data file with measurement data in it or start a measurement using this file.

3271

(ERR^CONTABFULL)

Cause Too many PROCESSH code ranges are configured for measurement, and the CONTAB is full.

Effect The requested operation aborts.

Recovery Reconfigure the PROCESSH measurement by reducing the number of code ranges (or taking out some code spaces).

3272

(ERR^INVALID^TAPE^OP)

Cause An invalid request was made on a tape data file.

Effect The requested operation aborts.

Recovery The request can only be made on a disk file. Use FUP COPY to copy the measurement data from the tape to a disk file of code 175 and then retry the operation.

3273

(ERR^TRAP^OCCURRED)

Cause The MEASFH process has trapped out.

Effect The MEASFH process stops abruptly, and a saveabend file is created.

Recovery If the trap number is 12 (%14), check the disk where the MEASFH work files were created. If the disk is full or very fragmented, use the MEASCOM SWAPVOL command or the SWAP parameter in the TACL RUN command to request that the MEASFH work files be created on another disk. If the trap is not caused by the inability to create an extent, the CPU is short of memory. In that case, try running the MEASFH process in a CPU with more memory. For traps other than trap 12, copy the displayed information and report the problem to your service provider.

3274

(ERR^UNKNOWNMSG)

Cause The MEASFH version is incompatible. The MEASFH process received a request it could not understand. Either the request is not implemented in the current MEASFH PVU, or the request is of an invalid type.

Effect The MEASFH process ignored the request.

Recovery If the request is valid, use a MEASFH file that supports the new request. Otherwise, supply a valid request.

3275

(ERR^WRONGRELEASE)

Cause An incompatible MEASFH version was specified.

Effect The requested operation aborts.

Recovery Specify the correct MEASFH version and retry the command.

3276

(ERR^MEAS^NEVER^STARTED)

Cause Measurement was not started correctly on this data file (error 3410 returned by MEASMON), or START time was not written to the data file.

Effect The data file does not contain valid data. Further operations on the data file could result in error.

Recovery Try to start another measurement. Check the EMS log for error 3410 (measurement not started correctly). Use a data file big enough that the START time can be written to the data file.

3277

(ERR^DATAFILE^CORRUPT)

Cause MEASFH encountered an invalid data record when adding the data file.

Effect The data file is analyzed only up to the invalid record.

Recovery None possible. MEASFH cannot analyze the data file past the corrupted record.

3278

(ERR^CONTAB^NOT^COMPLETED)

Cause The callable MEASREADCONF is an earlier PVU than MEASFH.

Effect MEASREADCONF could not handle an incomplete CONTAB from MEASFH. It cannot read large measurement configurations.

Recovery Use a C20 or later PVU of the callable MEASREADCONF.

3279

(ERR^CONVERTING^TO^RISC)

Cause MEASFH was unable to convert a TNS address to a TNS/R address while processing the EDIT file for SC or SL.

Effect The measurement is not configured.

Recovery Verify that each TNS address in the EDIT file is valid for the code space being configured. To do so, check the CONFLIST in SYSTEM.SYS*nn*. Although Measure accepts any number from 0 through %177777 for a TNS address, if the code space was accelerated, only addresses mapped by the NonStop operating system into a TNS/R address are accepted in the EDIT file.

3280

(ERR^FH^OLD^STRUCT^FILE)

Cause The record size is different from the size in the data file.

Effect No records were written to the structured file.

Recovery Remove the existing structured file and let MEASFH create it, or use a different subvolume. If using MEASCOM, use the VOLUME command. Be sure to regenerate the proper DDL dictionary.

3281

(ERR^FH^CTR^NAME^NOT^FOUND)

Cause The counter name passed in the IF^ITEM structure was not found. Perhaps the name is misspelled or does not exist in the PVU being processed.

Effect The operation was not performed.

Recovery Pass a valid counter name.

3282

(ERR^FH^BY^CTR^ONLY)

Cause The counter name in the IF^ITEM structure is valid but cannot be used for the IF item.

Effect The operation was not performed.

Recovery Pass a different counter name.

3283

(ERR^RESIZESEGMENT)

Cause The MEASFH process was not able to obtain the extended segment size needed to perform the requested operation.

Effect The requested operation aborts.

Recovery Designate a CPU with more available memory and retry the command.

3284

(ERR^ALLOCATE^SEG)

Cause The MEASFH process was not able to obtain the extended segment size needed to perform the requested operation.

Effect The requested operation aborts.

Recovery Designate a CPU with more available memory and retry the command.

3285

(ERR^NOT^TNDM)

Cause The MEASFH process was not able to read the COFF file further.

Effect The operation was not performed.

Recovery Designate a proper COFF file and retry the MEASCOM ADD PROCESSH command.

3286

(ERR^EDIT^NOT^ASCEND)

Cause The MEASFH process was not able to perform the requested operation because the native address in the edit file was not in ascending order.

Effect The requested operation aborts.

Recovery Make changes in the edit file and retry the command.

3287

(ERR^NO^POOLSPACE)

Cause MEASFH was not able to perform the requested operation. The MEASMON process cannot obtain the memory space required to start this measurement.

Effect The operation was not performed.

Recovery Split the measurement into multiple measurements and retry the command.

3288

(ERR^NO^GINFO)

Cause The MEASFH process was not able to read the COFF file further.

Effect The operation was not performed.

Recovery The COFF file is partially linked. Link the COFF file completely by passing it through the Native Mode Linker, NLD without the -r option, and then retry the operation with the output of the Linker process.

3289

(ERR^DATAFILE^INCOMPLETE)

Cause One or more CPUs ran out of CIDs, counter space, or both. Consequently, the data file is incomplete.

Effect The data file does not contain complete information for the measurement.

Recovery Retry the measurement with fewer entities specified. For more information, see the EMS log.

3290

(ERR^VERSION^NOT^SUPPORTED)

Cause The Measure PVU is not supported.

Effect The data file is not read.

Recovery Supply a valid Measure PVU and retry the operation.

3291

(ERR^NO^EGINFO)

Cause MEASFH was not able to read the ELF file further.

Effect The operation was not performed.

Recovery The ELF file is partially linked. Pass the ELF file through the Native Mode Linker, NLD without the -r option and then retry the operation with the output of the linker process.

3292

(ERR^CANNOTMAKEJOURNAL)

Cause An ADD MEASUREMENT command (or implicit ADD MEASUREMENT command) was entered for a data file that was still in the process of journal segment construction at measurement shutdown, or a START MEASUREMENT command was entered with insufficient temporary file space for the MEASFH process to establish a journal segment.

Effect The data file is opened or the measurement is started, but without the OSS or ANSI SQL Journal Segment or name translation.

Recovery When adding a data file, use the DELETE MEASUREMENT and ADD MEASUREMENT commands to re-add the data file and attach the journal segment after construction is complete. When starting a measurement, verify adequate security for temp file creation, suitable space for the file (up to 64MB).

3293

(ERR^NOT^REPORTFORMATNOTSTRUCTURED)

Cause LIST OSSNAMES or LIST EXTNAMES was issued without having set REPORT FORMAT STRUCTURED.

Effect No records are written to the OSSNAMES or EXTNAMES file.

Recovery Prior to issuing a LIST OSSNAME or LIST EXTNAMES command, specify REPORT FORMAT STRUCTURED.

3294

(ERR^CONFIGFILECONFLICT)

Cause Two files were specified for configuring the same code space range of the same object in a measurement configuration. MEASFH could not complete the measurement configuration because of the invalid specification.

Effect The operation was not performed.

Recovery Remove one of the two files from the specification.

3295

(ERR^SQL^API^INTERNAL)

Cause The MEASURE API internal check failed due to insufficient buffer space.

Effect The ANSI SQL name operation could not be performed.

Recovery Contact your HP representative.

3296

(ERR^SQLMX^MAP^PROCESS)

Cause A call to the SQL/MX mapping API failed because no mapping process has been started.

Effect The ANSI SQL name operation could not be performed.

Recovery Verify that SQL/MX and TMF are installed and active. If that is the case, contact your HP representative.

3297

(ERR^SQLMX^MAP^DUPCONNECTION)

Cause There is already an existing SQL/MX mapping session for this application process.

Effect No new mapping session is started.

Recovery If MEAS_SQL_MAP_STOP was just called, wait and try again, else it can probably be ignored and an already existing *connectionInfo* can be used to call the appropriate APIs.

3298

(ERR^SQLMX^MAP^INIT)

Cause A SQL/MX mapping session could not be started.

Effect No new mapping session is started.

Recovery Check if SQL/MX and TMF are installed and active. If that is the case, contact your HP representative.

3300

(ERRM^INVALIDMSG)

Cause The type or length of the request was invalid.

Effect The requested operation aborts.

Recovery Correct the type or length and retry the request.

3301

(ERRM^BADMEASNUM)

Cause The measurement number supplied in the request was invalid for the requested service.

Effect The requested operation aborts.

Recovery Supply the correct measurement request number and retry the operation.

3302

(ERRM^SECURITYVIOLATION)

Cause The requester did not have proper security rights to perform the requested operation.

Effect The requested operation aborts.

Recovery Verify your request and retry the operation.

3303

(ERRM^OPENOVERFLOW)

Cause There were already 100 processes open.

Effect The requested operation aborts.

Recovery There can be no more than 100 open processes. Wait and retry the operation.

3304

(ERRM^INTERNAL)

Cause An internal error occurred.

Effect The requested operation aborts.

Recovery Contact your HP representative.

3305

(ERRM^DATAFILEINUSE)

Cause An attempt was made to use a data file that was already being used by another measurement.

Effect The requested operation aborts.

Recovery Specify another data file.

3306

(ERRM^MEASTABLEFULL)

Cause An attempt was made to configure more than 64 measurements.

Effect The requested operation aborts.

Recovery Wait until some measurements have stopped and retry the operation.

3307

(ERRM^NOCONTABSPAC)

Cause The MEASMON CONTAB pool, containing entity descriptors, was full.

Effect The requested operation aborts.

Recovery Wait until some measurements have stopped and retry the measurement.

3308

(ERRM^INVALIDINTERVALTIME)

Cause The interval time is invalid. It cannot be less than one second or greater than one year.

Effect The requested operation aborts.

Recovery Correct the interval time value and retry the request.

3309

(ERRM^INVALIDSTARTSTOPTIME)

Cause In a control-measurement request, the stop time was less than the start time, an attempt was made to change the start time of an active measurement, or there was some other invalid start or stop time request.

Effect The requested operation aborts.

Recovery Correct the start and stop time values and retry the request.

3310

(ERRM^INVALIDDATAFILE)

Cause An incorrect data file type was specified.

Effect The requested operation aborts.

Recovery Specify a tape file or an unstructured disk file with a file code of 175.

3311

(ERRM^INVALIDCONTAB)

Cause The CONTAB format was not correct.

Effect The requested operation aborts.

Recovery Correct your CONTAB format and retry the operation.

3312

(ERRM^DATAFILEOPENFAILED)

Cause An attempt by MEASMON to open the data file failed.

Effect The requested operation aborts.

Recovery Check the current state of the data file and take appropriate action.

3313

(ERRM^ADOPTFAILED)

Cause An attempt by MEASMON to adopt an existing MEASCTL or MEASIP process failed.

Effect The requested operation aborts.

Recovery Notify your system operator.

3314

(ERRM^ALREADYOPEN)

Cause The caller process has already opened MEASMON.

Effect The requested operation aborts.

Recovery The MEASMON process can be opened only once by the caller. It is already open. Check the MEASCB structure passed to the callable procedure. The caller should not have modified it.

3315

(ERRM^START^MEASSUBSYS)

Cause The backup for MEASMON did not start, one or more executing CPUs does not contain an active MEASCTL or MEASIP process, or both.

Effect The MEASURE subsystem is not fully operational.

Recovery Correct the problems indicated by the console messages. Then stop the MEASURE subsystem and restart it.

3316

(ERRM^MEMLOCKFAILURE)

Cause The call to the LOCKMEMORY procedure failed. MEASMON was not able to perform the security check needed to access the data file.

Effect The measurement was not started.

Recovery Retry the request. If the problem persists, notify your system operator.

3317

(ERRM^INVALIDBINADDR)

Cause An invalid code range was given in the PROCESSH entity.

Effect The measurement does not run.

Recovery Supply a valid code range address and retry.

3318

(ERRM^INVALIDBINPART)

Cause The PROCESSH configuration was not sent to MEASMON. MEASMON was unable to configure the measurement.

Effect The measurement was not started.

Recovery Retry the request. If the problem persists, notify your HP representative.

3319

(ERRM^NOBINADDRSPACE)

Cause The pool space for code-range addresses is full in the MEASCTL process.

Effect The measurement does not run.

Recovery Stop some measurements and then try to start your measurement again.

3320

(ERRM^BACKUP^DATAFILEOPENFAILED)

Cause An attempt by backup MEASMON to open the data file failed.

Effect The requested operation aborts.

Recovery Check the file system error number and take appropriate action.

3400

(ERRM^READLINKFAILED)

Cause The complete request could not be read because the processor did not have sufficient physical memory.

Effect The reply file does not contain any data.

Recovery Install more memory.

3401

(ERRM^BUFFERTOOSMALL)

Cause The requestor's buffer was too small to complete the request.

Effect The reply file does not contain any data.

Recovery Allocate more space for the reply and retry the operation.

3402

(ERRM^CANNOTACCESS)

Cause The entity could not be accessed because of the current state. The CPU, group, module, slot number, SCSI ID, SAC name, lun, or path entered for the device may not be the correct value. Find out the correct value from SCF and retry the command.

Effect The reply does not contain any data.

Recovery If the device is inaccessible, contact your system operator.

3403

(ERRM^ILLEGALDESC)

Cause The entity descriptor in the request was invalid.

Effect The reply does not contain any data.

Recovery Correct the entity descriptor and retry the request.

3404

(ERRM^NOTMEASURING)

Cause The measurement number (MEASNUM) parameter did not measure the entity, or the entity does not exist.

Effect The reply does not contain any data.

Recovery Check that the entity exists in the measurement CONTAB parameter.

3405

(ERRM^REPLYTOOBIG)

Cause The reply did not fit in the MEASCTL reply buffer. This can result when there are too many user-defined counters to be returned in a reply to a READACTIVE request.

Effect The reply does not contain any data.

Recovery Eliminate some of the user-defined counters.

3406

(ERRM^ILLEGALREQ)

Cause The request was invalid.

Effect The reply does not contain any data.

Recovery Check and correct the entity-descriptor type, check the actual request and then retry the operation.

3407

(ERRM^ILLEGALMEASNUM)

Cause The measurement number (MEASNUM) in the request was not a valid measurement number, or there was no active measurement with this measurement number.

Effect The reply does not contain any data.

Recovery Supply the correct measurement number.

3408

(ERRM^MEASNUMINUSE)

Cause An attempt was made to start an already active measurement.

Effect The operation is aborted.

Recovery Contact your HP representative.

3409

(ERRM^NOCONTABSPACE)

Cause The MEASCTL CONTAB pool was full.

Effect The operation is aborted.

Recovery Wait until some other measurements have stopped and then retry the operation.

3410

(ERRM^INSTALL)

Cause A resource problem occurred with MEASCTL.

Effect The operation is aborted.

Recovery Wait until some other measurements have stopped or reduce the number of measurement entities, and then retry the operation.

3411

(ERRM^DFOPENFAILED)

Cause The data file could not be opened.

Effect The operation is aborted.

Recovery Check the current open state of the data file and its file security. Retry the operation with the right data file.

3420

(ERRM^NOCIDENTRY)

Cause The MEASCTL process in the specified CPU ran out of internal counter space while starting a measurement.

Effect Some entities are not measured.

Recovery For all measurements in the system, reduce the amount of transient entities to be measured by configuring only needed entities. For example, avoid using wild-card (*) specifications in the configurations. Use the DELETE command to exclude entities that are not needed.

3421

(ERRM^NOCOUNTERSPACE)

Cause The MEASCTL process in the specified CPU ran out of internal counter space while starting a measurement.

Effect Some entities are not measured.

Recovery For all measurements in the system, reduce the amount of transient entities to be measured by configuring only needed entities. For example, avoid using wild-card (*) specifications in the configurations. Use the DELETE command to exclude entities that are not needed.

3430

(ERRM^ILLEGALINDEX)

Cause An illegal SQL statement index number was supplied in the LISTACTIVE SQLSTMT command.

Effect The operation is aborted.

Recovery Check that the number supplied is a legal statement index number for the desired SQL process.

C Subsystem Files

This list describes the files that make up the Measure performance monitor:

MEASCOM process	MEASCOM is the Measure subsystem command interpreter. For descriptions of the commands, see Chapter 2: MEASCOM Commands (page 37). Typically started from TACL, a MEASCOM process lets the user who started it carry on an interactive session with the Measure subsystem. A super-group user (member of user group 255) can use MEASCOM to start the Measure subsystem. Thereafter, any user can use MEASCOM to configure, run, and examine measurements. Concurrent sessions as well as concurrent measurements within a session are allowed. MEASCOM is stored in $SYSTEM.SYS*nn*.
MEASFH process	Created by the callable procedures whenever you run a measurement or later examine the resulting data file, one MEASFH process presides over one user-designated data file: formatting it, adding data to it, and retrieving data as needed. MEASFH is release-dependent in that a data file generated in one version can be examined only by using a MEASFH process of the same version. MEASFH is stored in $SYSTEM.SYS*nn*. In the H01 PVU, the default record style changed from Legacy to ZMS. Accordingly, in G-series and earlier RVUs, fields that appear only in ZMS style records and reports cannot be used in "IF" clauses, even in the ZMS style report mode. In H-series and J-series RVUs, fields that appear only in legacy style records and reports cannot be used in "IF" clauses, even in the legacy style report mode.
MEASMON process	Serving as the subsystem monitoring and coordinating process, a single MEASMON process (running as a process pair) is created when the Measure subsystem is started. MEASMON creates a MEASCTL process in each CPU and coordinates the activity of these processes, sending them measurement configuration requests and gathering information from them. MEASMON is stored in $SYSTEM.SYS*nn*.
MEASCTL process	MEASMON creates one MEASCTL process in each CPU when the Measure subsystem is started. The job of a MEASCTL process is to handle a description of counter space and to engage the relevant subsystems responsible for maintaining the counters. Among the tasks performed by MEASCTL are those that initialize counters when a measurement starts and those that copy counters to the data file. MEASCTL also retrieves for display the counters a user requests for online examination. MEASCTL is stored in $SYSTEM.SYS*nn*.
MEASIP process	MEASIP is the Measure Interrupt Process. It does the sampling for a PROCESSH measurement. The MEASMON process creates one or more MEASIP processes that run in each CPU of the local system when the Measure subsystem starts. This process is not present in RVUs prior to H-series and J-series. MEASIP is stored in $SYSTEM.SYS*nn*.
MEASDDLS file	The MEASDDLS file contains the record definitions (DDLs) of the optional structured files produced by the commands and procedures that display measurement results. You can use these record definitions to produce language-specific record declaration files for programmatic use of data or as the basis for user-developed reports using the Enform product. MEASDDLS is stored in $SYSTEM.SYS*nn*.
MEASDDLF	The MEASDDLF file contains FORTRAN DDL definitions. MEASDDLF is stored in $SYSTEM.SYS*nn*.
MEASDDLB	The MEASDDLB file contains COBOL85 DDL definitions. MEASDDLB is stored in $SYSTEM.SYS*nn*.

MEASDDLZ file	The MEASDDLZ file lets applications access ZMS style data records with minimal impact. The ZMS style record structure adds a level of naming to each component of the record. The legacy format does not have this additional naming level. For example, the field cpu.dispatches in a legacy style record is equivalent to the zmscpu.ctr.dispatches in the ZMS style record structure. Converting existing applications and ENFORM reports from the legacy naming format to the ZMS style naming format is a significant edit of record references. It also involves removal of fields that might be referenced by programs, and the possible combination of equivalent fields that appeared in two sizes (for example, process.sent-bytes and process.sent-bytes-f). To simplify this process, use the MEASDDLZ file as described in the *Measure User's Guide*.MEASDDLZ presents the ZMS style counter widths and locations in the record using the legacy style record naming convention. Record template names use legacy style names (for example, PROCESS rather than ZMSPROC). Fields that have been combined are referenced by redefines.
MEASDECS file	The MEASDECS file contains the structure declarations and literal value definitions used in the Measure callable procedures. MEASDECS is stored in $SYSTEM.SYS*nn*. See C and C++ Language Usage Notes.
MEASCHMA file	The MEASCHMA file contains the structure declarations for C and TAL. Its output is similar to the output of the MEASDECS file. See C and C++ Language Usage Notes.
MEASIMMU file	The MEASIMMU file contains the Measure error messages and the online help text accessible through the MEASCOM HELP command. MEASIMMU is stored in $SYSTEM.SYS*nn*.

NOTE: MEASCTL, MEASFH, MEASIP, and MEASMON must be on the active system subvolume (SYS*nn*). The remaining Measure files need not be located on the active subvolume.

C and C++ Language Usage Notes

To see whether a field is within a union, refer to the C declarations in files MEASCHMA and MEASDECS. If the field is within a union, include the union name when writing code that uses the field.

Existing C or C++ programs that use Measure data might not compile using the C structure declarations in new PVUs of Measure's MEASCHMA or MEASDECS. References to fields that now are within a union must be changed to include the union name in the reference.

Process Identification Numbers

Process identification numbers (PINs) on systems running D-series, G-series, and later RVUs are divided into these two ranges:

- A low PIN range from 0 through 254
- A high PIN range from 256 through the maximum number supported for the CPU

The PIN 255 is used only for a synthetic process ID. For more information on PIN 255, see the *Guardian Application Conversion Guide*.

Processes on C-series systems run at low PINs. A process running at a high PIN cannot communicate with a process or a file that resides on a system running a C-series RVU. To let a D-series, G-series, or later process access a C-series process, you must run the D-series, G-series, or later process at a low PIN.

Table C-1 PIN Information for the Measure Product

Process	Can run at high PIN	Defaults to high PIN
MEASCOM	No	N.A.
MEASMON	Yes	Yes
MEASCTL	Yes	Yes
MEASFH	Yes, in Measure H02 and later PVUs	Yes, in Measure H02 and later PVUs

When you run the Measure product on a system running a D-series, G-series, or later RVU, you need not be concerned about PIN ranges. However, to access a measurement data file on a system running a C-series RVU, you must run the Measure product at a low PIN.

To run a Measure process at a low PIN, use the /HIGHPIN OFF/ option in the TACL RUN command, or set the TACL HIGHPIN built-in variable to OFF.

NOTE: To examine data files from D-series or previous RVUs on systems running G-series or later RVUs, you must use a D*nn* PVU of MEASCOM as well as the appropriate MEASFH. For maximum compatibility with a variety of data files, always use the most recent D*nn* PVU of MEASCOM.

D Measure Data File Tool (MEASFT)

The Measure Data File Tool (MEASFT) lets you split a measurement data file into multiple data files for consumption and display by MEASCOM or through Measure APIs. MEASFT is especially useful in the case of the very large data files that can result from measuring large numbers of entities or measuring at very small intervals.

MEASFT is part of H06.07 and later RVUs, but you can use it to split measurement data files created by G10 or later G-series PVUs. To split a G-series data file, you must first move the file to a system running an H-series and J-series RVU. After running MEASFT, either move the output data files back to the original system for processing and display, or process them on the system running an H-series and J-series RVU, using the MEASFH option of the ADD command to specify the appropriate MEASFH object file version.

NOTE: In Measure G12 or later G-series PVUs, it is no longer necessary to move a G-series data file to a system running an H-series and J-series RVU in order to split the file.

MEASFT has two major functions, each invoked by a command:

- SPLIT. This command splits one measurement data file into multiple measurement data files. You specify whether to split the file by entity type(s) or by processor number. Measurement configuration, START/STOP measurement records, and any applicable OSS and SQL journal data are replicated in each output data file.
- INFO. This command reads a measurement data file and displays a report indicating how much file space is used for data associated with each entity type and processor number. This information can help you decide how best to split the data file.

MEASFT also has a HELP command, which displays syntax requirements for MEASFT commands.

INFO Command

This command produces a report that indicates how much space in a data file is associated with each entity type and processor type. You can use this information to help you determine the best split criteria for the file.

Configuration, START/STOP records, and journal data are listed separately and not included in the entity and processor usage statistics.

Syntax

```
INFO filename
```

filename

identifies the Measure data file to be examined. Use a full or partially qualified Guardian filename.

Example

```
$PERF CAPAPP 28> measft info mdata5
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
INFO MDATA5
$PERF.CAPAPP.MDATA5 (H01) File Size 133980160 bytes

Data file usage (in bytes) by Entity

Entity            Space       %
------          ----------  ------
CPU                133056    0.10
DISC              2031486    1.52
DISCOPEN         25678814   19.17
DISKFILE         17085880   12.75
FILE             35348390   26.38
```

```
NETLINE             173644    0.13
OSSCPU              155626    0.12
OSSNS                23394    0.02
PROCESS           41238946   30.78
SERVERNET         10585470    7.90
SQLPROC               9956    0.01
SQLSTMT              82328    0.06
TMF                  29766    0.02
------          ----------  ------
Total            132576756   98.95

Data file usage (in bytes) by Processor Number

CPU          Space       %
---       ----------  ------
 0          28409802   21.20
 1          19602142   14.63
 2          41878388   31.26
 3          42686424   31.86

Start time: March 10, 2006 08:19:05
Stop time:  March 10, 2006 08:38:53
```

SPLIT Command

This command splits the specified data file into multiple data files. It can also be used to produce a single file comprising selected records from the original data file.

After splitting the data file, MEASFT displays a summary of the resulting split, showing the output data file names, size and criteria.

Syntax

```
SPLIT filename [, out-file-set [, split-criteria] ]
```

filename

specifies the data file to be split. It can be a partially or fully qualified Guardian filename. (That is, you can specify or omit the node, volume and subvolume.)

out-file-set

specifies the names of the files to be created from the split. Specify the set in one of these ways:

```
*
name-prefix*
(name1[,name2]...)
name
```

If you omit both the *out-file-set* and the *split-criteria*, MEASFT divides the input data file into two files, named ZMSRF000 and ZMSRF001, using split criteria that make the split as even as possible.

If you specify an asterisk (*), MEASFT creates the output data files with names of the form ZMSRF*nnn*, beginning with ZMSRF000. The number of output data files created depends upon *split-criteria*. If *split-criteria* is not specified, MEASFT behaves as described in Usage Note (page 498).

If you specify a string of characters ending in an asterisk, MEASFT regards the value as a filename prefix and appends a number, starting with "00" for the first file. The number of output data files created depends upon the*split-criteria*. If *split-criteria* is not specified, MEASFT behaves as described in Usage Note (page 498).

If you specify a comma-separated list of filenames, enclosed in parentheses, MEASFT uses those names for the files it creates. You must specify at least one name. If the number of specified names is different from the number of output data files dictated by the split-criteria, MEASFT terminates with an error message.

If you specify only one name, you must specify *split-criteria* to indicate what should go into the single output data file.

name-prefix, *namen* and *name* can be partially or fully qualified names. For example, all of the following specifications are valid:

```
\node.$vol.subvol.*
```

```
subvol.pref*
```

```
(split1, \node.$vol.subvol.rsplit2, othersv.split3)
```

If a file with the specified name already exists and it is not the data file to be split--the file identified by *filename*--MEASFT displays a warning message and purges the data before writing to the file. If the existing file is the data file to be split, MEASFT displays an error message and terminates.

split-criteria

specifies how the data file will be split. Its syntax is one of:

BY ENTITY *
BY ENTITY (*entity-list1*[, *entity-list2*]...)
BY CPU *
BY CPU (*cpu-list1*[, *cpu-list2*]...)

BY ENTITY followed by an asterisk (*) causes MEASFT to create one output data file per entity type present in the data file.

BY ENTITY followed by one or more entity lists causes MEASFT to create one output data file for each list of entities. The syntax for entity-list*n* is:

```
entity-name [entity-name]...
```

that is, a space-separated list of *entity-name*. The permissible values for *entity-name* are:

CLUSTER
CPU
DEVICE
DISC or DISK
DISCOPEN or DISKOPEN
DISKFILE
FILE
LINE
NETLINE
OPDISK
OSSCPU
OSSNS
PROCESS
PROCESSH
SERVERNET
SQLPROC
SQLSTMT
SYSTEM
TERMINAL
TMF
USERDEF

The last entity-list*n* can also be an asterisk (*), which instructs MEASFT to put data for all the remaining entities into the last output data file. If only *entity-list1* is specified, a single output data file is created.

BY CPU followed by an asterisk (*) causes MEASFT to create one output data file per processor number present in the data file.

BY CPU followed by one or more lists of processor numbers causes MEASFT to create one output data file for each list of processor numbers. The syntax for cpu-list*n* is:

```
cpu-num [cpu-num]...
```

that is, a space-separated list of *cpu-num*.

The last *cpu-listn* can also be an asterisk (*) which means to put all of the rest of the processor data into the last output data file. If only *cpu-list1* is specified, a single output data file is created, to contain data for the processor numbers in the list.

Usage Note

If you specify an asterisk (*) as the *out-file-set* and omit the *split-criteria* parameter, MEASFT chooses the number of output data files by dividing the Measure data file size by 1024 MB (1,073,741,824 bytes) and rounding down. It uses this number or 2, whichever is larger, as the number of output data files. It then groups the entities to enable as even a split as possible. For example:

```
$PERF KMZMFT 113> measft split tests.bigdata,*
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
SPLIT TESTS.BIGDATA,*
File $PERF.TESTS.BIGDATA was split into the following files:
File                                  Size       %    Entity(s)
---------------------------------- ----------  ------  --------------------
$PERF.KMZMFT.ZMSRF000                27391786    2.69  CPU,DISC,NETLINE,
                                                       SYSTEM,TMF,SQLPROC,
                                                       SQLSTMT,OSSCPU,OSSNS
$PERF.KMZMFT.ZMSRF001                51509214    5.06  DISKFILE
$PERF.KMZMFT.ZMSRF002                86759610    8.52  SERVERNET
$PERF.KMZMFT.ZMSRF003               145165294   14.26  DISCOPEN
$PERF.KMZMFT.ZMSRF004               165726444   16.28  FILE
$PERF.KMZMFT.ZMSRF005               190483848   18.71  PROCESS
$PERF.KMZMFT.ZMSRF006               337090908   33.11  PROCESSH
```

Examples

- Split Measure data file MDATA5 without specifying any output data filenames or split criterion. MEASFT attempts to create a two-way split that is as even as possible. In this case it puts PROCESS and FILE records in one file and all other records in another file.

```
$PERF CAPAPP 7> measft split mdata5
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
SPLIT MDATA5
File $PERF.CAPAPP.MDATA5 was split into the following files:
File                                  Size       %    Entity(s)
---------------------------------- ----------  ------  --------------------

$PERF.CAPAPP.ZMSRF000                55990206   41.79  CPU,DISCOPEN,DISC,
                                                       NETLINE,TMF,SQLPROC,
                                                       SQLSTMT,SERVERNET,
                                                       DISKFILE,OSSCPU,OSSNS
$PERF.CAPAPP.ZMSRF001                76588122   57.16  PROCESS,FILE
```

- Split Measure data file MDATA5 by entity, with each entity going into a separate output data file. Let MEASFT name the output data files.

```
$PERF CAPAPP 8> measft split mdata5,*,by entity *
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
SPLIT MDATA5,*,BY ENTITY *
File $PERF.CAPAPP.MDATA5 was split into the following files:
File                                  Size       %    Entity(s)

---------------------------------- ----------  ------  --------------------
$PERF.CAPAPP.ZMSRF000                  133842    0.10  CPU
$PERF.CAPAPP.ZMSRF001                41239732   30.78  PROCESS
$PERF.CAPAPP.ZMSRF004                35349176   26.38  FILE
$PERF.CAPAPP.ZMSRF005                25679600   19.17  DISCOPEN
$PERF.CAPAPP.ZMSRF006                 2032272    1.52  DISC
$PERF.CAPAPP.ZMSRF009                  174430    0.13  NETLINE
$PERF.CAPAPP.ZMSRF013                   30552    0.02  TMF
```

```
$PERF.CAPAPP.ZMSRF014                    10742    0.01  SQLPROC
$PERF.CAPAPP.ZMSRF015                    83114    0.06  SQLSTMT
$PERF.CAPAPP.ZMSRF017                 10586256    7.90  SERVERNET
$PERF.CAPAPP.ZMSRF018                 17086666   12.75  DISKFILE
$PERF.CAPAPP.ZMSRF019                   156412    0.12  OSSCPU
$PERF.CAPAPP.ZMSRF020                    24180    0.02  OSSNS
```

- Split Measure data file MDATA5 by entity, with FILE data going into a file named SFFILE and the rest going into SFREST.

```
$PERF CAPAPP 10> measft split mdata5,(sffile,sfrest),by entity (file,*)
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
SPLIT MDATA5,(SFFILE,SFREST),BY ENTITY (FILE,*)
File $PERF.CAPAPP.MDATA5 was split into the following files:

File                                    Size       %    Entity(s)
----------------------------------- ----------  ------  --------------------
$PERF.CAPAPP.SFFILE                   35349176   26.38  FILE
$PERF.CAPAPP.SFREST                   97229152   72.57  CPU,PROCESS,DISCOPEN,
                                                        DISC,NETLINE,TMF,
                                                        SQLPROC,SQLSTMT,
                                                        SERVERNET,DISKFILE,
                                                        OSSCPU,OSSNS
```

- Split Measure data file MDATA5 by entity, with CPU and PROCESS data going into a file named SFFILE1 and SERVERNET data going into SFFILE2.

```
$PERF CAPAPP 14> measft split mdata5,(sffile1,sffile2),
                              by entity (cpu process,servernet)
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD

Copyright 2006 Hewlett-Packard Company

SPLIT MDATA5,(SFFILE1,SFFILE2),BY ENTITY (CPU PROCESS,SERVERNET)
File $PERF.CAPAPP.MDATA5 was split into the following files:
File                                    Size       %    Entity(s)
----------------------------------- ----------  ------  --------------------
$PERF.CAPAPP.SFFILE1                  41372788   30.88  CPU,PROCESS
$PERF.CAPAPP.SFFILE2                  10586256    7.90  SERVERNET
```

- Split Measure data file MDATA5 by processor number, with each processor's data going into a separate output data file. The name of the output data files start with "MCPU".

```
$PERF CAPAPP 16> measft split mdata5,mcpu*,by cpu *
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company
SPLIT MDATA5,MCPU*,BY CPU *
File $PERF.CAPAPP.MDATA5 was split into the following files:
File                                    Size       %    CPU(s)
----------------------------------- ----------  ------  --------------------
$PERF.CAPAPP.MCPU00                   28410588   21.21  0
$PERF.CAPAPP.MCPU01                   19602928   14.63  1
$PERF.CAPAPP.MCPU02                   41879174   31.26  2
$PERF.CAPAPP.MCPU03                   42687210   31.86  3
```

HELP Command

This command displays help information for the published commands.

Syntax

```
HELP | empty
```

Enter "MEASFT HELP" or just "MEASFT" to display help text for the MEASFT command.

MEASFT Error and Warning Messages

Any of the following messages can appear on the console when you run MEASFT. This example illustrates the format of a typical error and warning:

```
MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company

SPLIT KMEAS3, ,by cpu *
              ^
*** Error 3232: SPLIT <name> Resolution failed, File System Error = #13

$PERF KMZMFT 45> measft split kmeas2,*,by entity (cpu terminal,*)

MEASURE File Tool - T9086H01 - (01AUG06) - \YOSPRD
Copyright 2006 Hewlett-Packard Company

SPLIT KMEAS2,*,BY ENTITY (CPU TERMINAL,*)

*** Warning 2007: No data for this specified entity, TERMINAL

File $PERF.KMZMFT.KMEAS2 was split into the following files:

File                              Size       %    Entity(s)
------------------------------ ----------  ------  ----------------------
$PERF.KMZMFT.ZMSRF000              118098    0.60  CPU
$PERF.KMZMFT.ZMSRF001            18679336   95.33  PROCESS,PROCESSH,
                                                   SERVERNET
```

If a message indicates a syntax error, a circumflex (^) shows where the syntax error was detected.

MEASFT 2000 (WARN_SPLIT_FILE_NOT_CREATED)

`Specified file, out-file, not created because of lack of data records.`

Cause The specified output data file was not created because MEASFT found no data records to put into it.

Effect The SPLIT command continues, but the specified output file is not created.

Recovery Remove the specified output file and its corresponding split criteria from the command line.

MEASFT 2001 (WARN_SPLIT_FILE_PURGED)

`Specified output filename, out-file, already existed, previous data was overwritten.`

Cause User specified an output filename (either directly or indirectly through a wildcard) that already exists and is not the same as the input data file, the file that is to be split.

Effect The SPLIT command continues, and previous data is overwritten.

Recovery It is too late to recover the data in the overwritten file. In the future, specify an output filename that doesn't conflict with existing files.

MEASFT 2002 (WARN_NO_XRTIMES_AT_EOF)

`No Stop Measurement record, file possibly truncated/corrupted.`

Cause There is no Stop Measurement record in the input data file. The file is either a truncated copy or the file was corrupted.

Effect The INFO or SPLIT command continues.

Recovery Determine the cause of the missing record and decide whether this is a significant issue.

MEASFT 2003 (WARN_BAD_SVNET_ENT_LEN)

`Bad length in SVNET entity, entity-name, length #length, record #number`

Cause A record for the SERVERNET entity, *entity-name*, had an unexpected length, *length*. The record number is *number*.

Effect The INFO or SPLIT command continues.

Recovery Please contact HP about this warning and save the data file that caused it.

MEASFT 2004 (WARN_BAD_ENT_LEN)

`Bad length in entity, entity-name, length #length, record #number`

Cause A record for the entity, *entity-name*, had an unexpected length, *length*. The record number is *number*.

Effect The INFO or SPLIT command continues.

Recovery Please contact HP about this warning and save the data file that caused it

MEASFT 2005 (WARN_TOO_MANY_SPLIT_FILES)

`More output files specified than entities present; # files lowered from mm to nn`

Cause The Measure data file contains data records for fewer entities than split files specified, so MEASFT cannot do the split as specified. This warning occurs only when the user specifies a definite number of output files but no split criteria, forcing MEASFT to determine the split criteria.

Effect The SPLIT command continues. MEASFT ignores the extra output filenames.

Recovery Decrease the number of output data files specified.

MEASFT 2006 (WARN_NO_SPLIT_FILES_CREATED)

`No output files were created, no records satisfied split criteria.`

Cause None of the records in the Measure data file satisfied any of the split criteria so no output files were created.

Effect The SPLIT command continues.

Recovery Determine the cause of the missing records and decide whether this is a significant issue.

MEASFT 2007 (WARN_ENTITY_HAS_NO_DATA)

`No data for this specified entity, entity`

Cause No data records were encountered for an entity that was specified in the split criteria.

Effect The SPLIT command continues. If the data file contained no data for any entity to be represented in a given output data file--for example, if an output file is to contain data for PROCESS and DISK entities, but the data file being split contains no records for either of those entities--MEASFT also displays message 2000, indicating that that output file was not created. If the data file contained no data for any of the entities to be included in any output data file, MEASFT also displays message 2006, indicating that no output data files were created.

Recovery Determine the reason for the missing data and decide whether this is a significant issue.

MEASFT 2008 (WARN_REMAINING_ENTITY_HAS_NO_DATA)

`Output file for the remaining entities (*) was not created due to lack of data.`

Cause No data records were encountered for the last output file, whose *entity-list* was specified as a wildcard (*) in the SPLIT command.

Effect The SPLIT command continues. The specified output file is not created.

Recovery Determine the reason for the missing data and decide whether this is a significant issue.

MEASFT 2009 (WARN_CPU_HAS_NO_DATA)

`No data for this specified CPU, nn`

Cause No data records were encountered for CPU *nn*, which was specified in the SPLIT command split criteria.

Effect The SPLIT command continues. If the data file contained no data records for any CPU to be represented in an output file, message 2000 is also displayed.

Recovery Determine the reason for the missing data and decide whether this is a significant issue.

MEASFT 2010 (WARN_REMAINING_CPU_HAS_NO_DATA)

`Output file for the remaining CPUs (*) was not created due to lack of data.`

Cause No data records were encountered for the last output file, whose *cpu-list* was specified as a wildcard (*) in the SPLIT command.

Effect The SPLIT command continues. The specified output file is not created.

Recovery Determine the reason for the missing data and decide whether this is a significant issue.

MEASFT 2011 (WARN_SPLIT_FILE_NOT_CREATED_2)

`An output file was not created due to lack of data.`

Cause An output filename was not created because MEASFT found no data records that satisfied the split criteria for that file. This warning is always preceded by message 2007 and occurs only when an asterisk or a prefix followed by an asterisk was specified in the output file set.

Effect The SPLIT command is continues; the output file is not created.

Recovery Remove the split criteria for the output file associated with this message. See the preceding occurrence of message 2007 to know which split criterion caused the problem.

MEASFT 2032 (WARN_MISSING_JOURNAL)

journal-type `Journal missing.`

Cause Expected Journal was not encountered; journal-type can be either OSS or SQL.

Effect The command continues.

Recovery Check the filesize of the Measure data file to see whether the file was truncated; truncation can sometimes arise when you transfer a very large file. If the file was not truncated, it might be corrupted. If neither of these cases seems to apply, report this message to HP.

MEASFT 3000 (ERR_FNAME_RESOLVE)

`Measfile Name Resolution failed, File System Error = #`*nn*

Cause User specified a Measure data filename that could not be resolved by FILENAME_RESOLVE_.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine why the specified Measure data filename could not be resolved. Bad filename syntax is the most likely cause.

MEASFT 3001 (ERR_FNAME_UNRESOLVE)

`Measfile Name UnResolution failed, File System Error = #`*nn*

Cause User specified a Measure data filename that could not be unresolved by FILENAME_UNRESOLVE_.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine why the specified Measure data filename could not be unresolved. This is an unlikely error since the filename had already been resolved by FILENAME_RESOLVE_.

MEASFT 3002 (ERR_OPEN_MEASFILE)

`Measfile Open failed, File System Error = #nn`

Cause User specified a Measure data filename that could not be opened.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine why the specified Measure data file could not be opened. Likely reasons are the data file is missing, in use by another user, or not secured for read access for the user.

MEASFT 3005 (ERR_CREATE_SPLIT_FILE)

`Creation of output file, datafile, failed, File System Error = #nn`

Cause The output file, *datafile*, could not be created.

Effect The SPLIT command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Use the displayed File System error number, *nn*, to determine why the specified output file could not be created. Likely reasons are that the volume name was misspelled, the subvolume is not secured for write access for the user, or the volume is full.

MEASFT 3006 (ERR_OPEN_SPLIT_FILE)

`Open of output file, datafile, failed, File System Error = #nn`

Cause The output file, *datafile*, could not be opened.

Effect The SPLIT command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Use the displayed File System error number, *nn*, to determine why the specified output file could not be opened. This is an unlikely error since the output file had just been created successfully.

MEASFT 3007 (ERR_FILE_SETPOSITION)

`FILE_SETPOSITION_ failed, File System Error = #nn`

Cause The Measure data file could not be repositioned at the beginning of the file or near the end of the file if the data file has a journal section.

Effect The SPLIT command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Use the displayed File System error number, *nn*, to determine why the Measure data file could not be repositioned.

MEASFT 3008 (ERR_WRITEX_SPLIT_FILE)

`Write of output file, datafile, failed, File System Error = #nn`

Cause MEASFT encountered an error writing to the output file, *datafile*.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will be incomplete.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3009 (ERR_CLOSE_SPLIT_FILE)

`Close of output file, datafile, failed, File System Error = #nn`

Cause MEASFT encountered an error in closing the output file, *datafile*.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will probably be incomplete.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3010 (ERR_FILE_SETPOS_SPLIT_FILE)

`FILE_SETPOSITION_ of output file, datafile, failed, File System Error = #nn`

Cause MEASFT encountered an error in repositioning the output file, *datafile*, to the beginning of the file in order to update the journal pointer(s) in the first record.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will not have correct journal pointers.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3011 (ERR_READX_SPLIT_FILE)

`Read of output file, datafile, failed, File System Error = #nn`

Cause MEASFT encountered an error in reading the first record of the output file, *datafile*, so that it could update the journal pointer(s) in it.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will not have correct journal pointers.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3012 (ERR_WRUPDX_SPLIT_FILE)

`Writeupdate of output file, datafile, failed, File System Error = #nn`

Cause MEASFT encountered an error in writing back the first record of the output file, *datafile*, so that it could update the journal pointer(s) in it.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will probably not have correct journal pointers.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3013 (ERR_READX_MEASFILE)

`Measfile READX failed, File System Error = #nn`

Cause MEASFT encountered an error in reading the journal records of the Measure data file.

Effect The SPLIT command does not complete, MEASFT terminates in error. The created output files will not be complete.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3014 (ERR_FILE_NOT_FOUND)

`Measure data file not found.`

Cause MEASFT could not find the Measure data file.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Enter the correct location of the Measure data file.

MEASFT 3016 (ERR_FILE_GETINFO)

`FILE_GETINFOLISTBYNAME_ of datafile failed, File System Error = #nn.`

Cause MEASFT could not get info on the specified data file.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3017 (ERR_FILE_BAD_TYPE)

`Measure file not unstructured, file type encountered #n`

Cause The specified Measure data file is not unstructured.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Measure data files are unstructured files, whereas the one specified has file type *n*. Enter the correct file name.

MEASFT 3018 (ERR_FILE_BAD_FILECODE)

`Measure file code not 175, file code encountered #n`

Cause The specified Measure data file does not have a 175 file code.

Effect The SPLIT or INFO command is not executed. MEASFT terminates in error.

Recovery Measure data files have a file code of 175, whereas the one specified has file code *n*. Enter the correct file name.

MEASFT 3019 (ERR_SPLIT_FILE_NAME_ERR)

`Specified output filename, outfile, is the same as the file to be split.`

Cause User specified an output filename (either directly or indirectly through a wildcard) that is the same as the input data file, the file that is to be split.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Re-specify an output filename that doesn't conflict with the input data file name.

MEASFT 3020 (ERR_CREATE_SPLIT_FILE_DUP)

`Specified output filename, outfile, already exists and is not a measure data file.`

Cause User specified an output filename (either directly or indirectly through a wildcard) that already exists and it isn't a Measure data file. MEASFT will not overwrite a file that it is not an existing Measure data file.

Effect The SPLIT command is not executed. MEASFT terminates in error. Some output files may have been created.

Recovery Re-specify a output filename that doesn't conflict with existing file names.

MEASFT 3021 (ERR_PURGE_SPLIT_FILE)

`Purging output filename, datafile, failed, File System Error = #nn`

Cause The output file, *datafile*, could not be purged. MEASFT overwrites existing Measure data files if an output file with the same name is specified.

Effect The SPLIT command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Use the displayed File System error number, *nn*, to determine why the specified output file could not be purged.

MEASFT 3022 (ERR_SEG_ALLOCATE_)

`SEGMENT_ALLOCATE_ error #nn.`

Cause MEASFT could not allocate enough memory to run.

Effect The command is not executed. MEASFT terminates in error.

Recovery Use the SEGMENT_ALLOCATE_ error code, *nn*, to determine why memory could not be allocated. MEASFT requires about 3MB of memory to run.

MEASFT 3023 (ERR_USESEGMENT)

`USESEGMENT error.`

Cause MEASFT could not access enough memory to run.

Effect The command is not executed. MEASFT terminates in error.

Recovery Please report this message to HP.

MEASFT 3024 (ERR_CRL_BAD_REC_TYPE)

`Bad record type #`*`mm`* `in this context #`*`nn`*

Cause MEASFT encountered an unexpected record type.

Effect The command does not complete, MEASFT terminates in error. Some output files might have been created.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP, citing the record type, *mm*, and context, *nn*, displayed in this message.

MEASFT 3025 (ERR_CONTAB_TRLR_WRONG_CTXT)

`Unexpected CONTAB^TRAILER record in MF_NORMAL context.`

Cause MEASFT encountered a record type it did not expect.

Effect The command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP.

MEASFT 3027 (ERR_BAD_RECORD_STATE)

`Bad record state in record #`*`number`*

Cause MEASFT encountered a record state it did not expect in record *number*.

Effect The command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP.

MEASFT 3028 (ERR_BAD_DFFIRST_REC)

`Measure data file corrupt, 1st record wrong length, file version` *`Xnn`*

Cause MEASFT did not recognize the first record in the Measure data file, whose version was *Xnn*.

Effect The command is not executed. MEASFT terminates in error.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP.

MEASFT 3029 (ERR_UNEXPECTED_NONCTR_REC)

`Unexpected non-counter record #`*`number`*

Cause MEASFT encountered a record type it did not expect in record *number*.

Effect The command does not complete, MEASFT terminates in error. Some output files might have been created.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP.

MEASFT 3030 (ERR_UNSUPPORTED_VERSION)

`Unsupported Measure data file version,` *`Xnn`*

Cause MEASFT encountered a Measure data file version created by the *Xnn* version of Measure.

Effect The command does not complete, MEASFT terminates in error

Recovery MEASFT does not support data files created by Measure versions earlier than G10.

MEASFT 3031 (ERR_NO_DFFIRST_REC)

`Measure data file corrupt, 1st record not present.`

Cause MEASFT did not recognize the first record in the Measure data file.

Effect The command is not executed. MEASFT terminates in error.

Recovery Either the Measure data file has been corrupted or MEASFT has a defect. Please report this message to HP.

MEASFT 3033 (ERR_REC_LENGTH_COMP_ERR)

`Record length could not be computed, record #`*`nnnnn`*

Cause MEASFT could not compute the length of record number *nnnnn*. This is either a MEASFT defect or a corrupt Measure data file.

Effect The command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery If the Measure data file seems correct, contact HP.

MEASFT 3034 (ERR_REC_LENGTH_TOO_BIG)

`Record length too big, record #`*`nnnnn`*

Cause The length of record number *nnnnn* is too big. This is either a MEASFT defect or a corrupt Measure data file.

Effect The command does not complete, MEASFT terminates in error. Some output files may have been created.

Recovery If the Measure data file seems correct, contact HP.

MEASFT 3035 (ERR_BAD_RECORD_TYPE)

`Invalid record type, record #`*`nnnnn`*

Cause MEASFT encountered an invalid record type in record number *nnnnn*. The cause may be a corrupt Measure data file or the specified input data file is not a Measure data file.

Effect The command does not complete, MEASFT terminates in error. Some output files might have been created.

Recovery Verify that the specified Measure data file is correct.

MEASFT 3036 (ERR_READ_NO_DATA_ERR)

`File error, read no data, File System Error = #`*`nn`*`, record #`*`nnnnn`*

Cause MEASFT read no data and got File System error *nn*, at record number *nnnnn*. The cause may be a corrupt Measure data file or the specified input data file is not a Measure data file.

Effect The command does not complete, MEASFT terminates in error. Some output files might have been created.

Recovery Verify that the specified Measure data file is correct.

MEASFT 3200 (ERR_PARSE_NOT_KEYWORD)

`XXX`` is not a recognized keyword.`

Cause The string, *XXX*, in the command line is not a valid keyword.

Effect The command is not executed. MEASFT terminates in error.

Recovery Enter the correct keyword or command syntax.

MEASFT 3201 (ERR_PARSE_NO_FILENAME)

`<filename> parameter is missing.`

Cause The Measure data file was not specified in the command line.

Effect The command is not executed. MEASFT terminates in error.

Recovery Enter the Measure data file name.

MEASFT 3202 (ERR_PARSE_FILENAME_TOO_LONG)

`<filename> parameter too long.`

Cause The Measure data file name entered in the command line is too long.

Effect The command is not executed. MEASFT terminates in error.

Recovery Enter the correct Measure data file name

MEASFT 3230 (ERR_PARSE_SPLIT_NAME_TOO_LONG)

`SPLIT <name> parameter too long.`

Cause The indicated output file name is too long. This occurs when an unqualified output file name greater than 8 bytes is entered.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Enter the correct output file name.

MEASFT 3231 (ERR_PARSE_SPLIT_PREFIX_TOO_LONG)

`SPLIT <name-prefix> parameter too long.`

Cause The indicated output file name prefix is too long. Maximum prefix length is 6 bytes.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Enter a shorter output file prefix.

MEASFT 3232 (ERR_PARSE_SPLIT_FNAME_RESOLVE)

`SPLIT <name> Resolution failed, File System Error = #`*`nn`*

Cause User specified an output filename that could not be resolved by FILENAME_RESOLVE_.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error.

MEASFT 3233 (ERR_PARSE_SPLIT_FNAME_UNRESOLVE)

`SPLIT <name> UnResolution failed, File System Error = #`*`nn`*

Cause User specified an output filename that could not be unresolved by FILENAME_UNRESOLVE_.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Use the displayed File System error number, *nn*, to determine what caused the error. This is an unlikely error since the output filename had already been resolved by FILENAME_RESOLVE_.

MEASFT 3234 (ERR_PARSE_SPLIT_RPAREN_MISSING)

`SPLIT file specs right parenthesis missing.`

Cause The SPLIT file specs entry starts with a left parenthesis and is missing a right parenthesis.

Effect The SPLIT command does not complete. MEASFT terminates in error

Recovery Correct the SPLIT file specs syntax.

MEASFT 3235 (ERR_PARSE_SPLIT_BY_EXPECTED)

`SPLIT 'BY' keyword missing.`

Cause The SPLIT split criteria entry must start with "BY".

Effect The SPLIT command does not complete. MEASFT terminates in error

Recovery Correct the SPLIT split criteria syntax.

MEASFT 3236 (ERR_PARSE_SPLIT_BY_KW_EXPECTED)

`SPLIT 'BY CPU' or 'BY ENTITY' expected.`

Cause The SPLIT split criteria entry must start with "BY CPU" or "BY ENTITY".

Effect The SPLIT command does not complete. MEASFT terminates in error

Recovery Correct the SPLIT split criteria syntax.

MEASFT 3237 (ERR_PARSE_SPLIT_SC_RPAREN_MISSING)

`SPLIT split criteria right parenthesis missing.`

Cause The SPLIT split criteria entry starts with a left parenthesis and is missing a right parenthesis.

Effect The SPLIT command does not complete. MEASFT terminates in error

Recovery Correct the SPLIT split criteria syntax.

MEASFT 3238 (ERR_PARSE_SPLIT_SC_MISMATCH)

`SPLIT file specs and split criteria mismatch.`

Cause The SPLIT file specs and split criteria entries conflict in the number of output files specified.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT files specs and/or split criteria syntax so that they specify the same number of output files.

MEASFT 3239 (ERR_PARSE_SPLIT_BAD_CPU_NUM2)

`SPLIT criteria CPU number out of range.`

Cause The CPU specified in the SPLIT command split criteria is either out of range (0 to 15) or is not numeric.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the invalid CPU number.

MEASFT 3240 (ERR_PARSE_SPLIT_SC_CPU_WC_ERR1)

`SPLIT criteria CPU '*' must be in last <cpu-list>.`

Cause The wildcard (*) specified in the SPLIT command split criteria was not in the last *`cpu-list`*.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT BY CPU entry so that the wildcard *`cpu-list`* is the last one.

MEASFT 3241 (ERR_PARSE_SPLIT_SC_CPU_WC_ERR2)

`SPLIT criteria CPU '*' must be only item in last <cpu-list>.`

Cause The wildcard (*) specified in the SPLIT command split criteria was not the only item in the last *`cpu-list`*.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT BY CPU entry so that the wildcard *`cpu-list`* is the last one and that the wildcard is the only entry in that list.

MEASFT 3242 (ERR_PARSE_SPLIT_WC_ERR1)

`SPLIT criteria '*' can not be prepended to anything.`

Cause The wildcard (*) specified in the SPLIT command split criteria cannot be prepended to anything; it must be the last entry in the command line when no lists are specified.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT split criteria so that the wildcard is the last element on the command line.

MEASFT 3243 (ERR_PARSE_SPLIT_WC_ERR2)

`SPLIT criteria '*' requires that <out-file-set> be a '*'.`

Cause If wildcard (*) is specified in the SPLIT command split criteria, then the file set must also be a wildcard.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT split criteria so that either both file specs and split criteria have a wildcard or neither has one.

MEASFT 3244 (ERR_PARSE_SPLIT_SC_EMPTY)

`SPLIT criteria empty.`

Cause Nothing was specified between the parentheses in the SPLIT command split criteria.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Enter the desired SPLIT split criteria .

MEASFT 3245 (ERR_PARSE_SPLIT_SC_ENT_WC_ERR1)

`SPLIT criteria Entity '*' must be in last <entity-list>.`

Cause The wildcard (*) specified in the SPLIT command split criteria was not in the last *`entity-list`*.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT BY ENTITY entry so that the wildcard *`entity-list`* is the last one.

MEASFT 3246 (ERR_PARSE_SPLIT_SC_ENT_WC_ERR2)

`SPLIT criteria Entity '*' must be only item in last <entity-list>.`

Cause The wildcard (*) specified in the SPLIT command split criteria was not the only item in the last *`entity-list`*.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the SPLIT BY ENTITY entry so that the wildcard *`entity-list`* is the last one and that the wildcard is the only entry in that list.

MEASFT 3247 (ERR_PARSE_NOT_ENT_KEYWORD)

`XXX` `is not a recognized Entity.`

Cause The string, *`XXX`*, in the command line is not a valid entity name.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Enter a correct entity name.

MEASFT 3248 (ERR_PARSE_SPLIT_ONE_FILE)

`SPLIT criteria required (no wildcard) for a single output file.`

Cause Only one output filename was specified but no split criterion was specified.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Add split criteria to the specified SPLIT command.

MEASFT 3249 (ERR_PARSE_FS_SYNTAX)

```
SPLIT <out-file-set> syntax error.
```

Cause Multiple output file names were specified in the file specs with no parentheses.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Correct the SPLIT file specs syntax.

MEASFT 3250 (ERR_PARSE_SPLIT_FS_WC_ERR)

```
SPLIT <out-file-set> filename in list cannot have an '*'.
```

Cause Entries in the SPLIT command file specs cannot use a wildcard if names are enclosed in parentheses. In fact, the wildcard is only permitted by itself or as a output file name prefix.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Correct the SPLIT file specs syntax.

MEASFT 3251 (ERR_PARSE_SPLIT_BAD_CPU_NUM1)

```
SPLIT criteria CPU not numeric or out of range.
```

Cause The CPU specified in the SPLIT command split criteria is not numeric or is out of range.

Effect The SPLIT command does not complete. MEASFT terminates in error.

Recovery Correct the CPU number in error.

MEASFT 3252 (ERR_PARSE_SPLIT_SC_BAD)

```
Syntax error after 'BY CPU/ENTITY', expected '*' or '('.
```

Cause Something other than a wildcard or left parenthesis followed the "BY CPU" or "BY ENTITY" phrase.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Correct the SPLIT split criteria syntax.

MEASFT 3253 (ERR_PARSE_SPLIT_MISSING_DEL)

```
SPLIT file specs missing a delimiter (',' or ')').
```

Cause A delimiter (either a comma or a right parenthesis) is missing in the SPLIT file specs.

Effect The SPLIT command is not executed. MEASFT terminates in error.

Recovery Correct the SPLIT file specs syntax

MEASFT 3300 (ERR_CANNOT_SPLIT)

```
The specified measure file cannot be split by entity, try splitting by
CPU.
```

Cause The Measure data file contains data records for only one entity, so it cannot be split by entity. This error occurs only when no split criterion is specified; by default, MEASFT attempts to split the data file by entity.

Cause The SPLIT command is not executed. MEASFT terminates in error.

Recovery Try splitting by CPU.

MEASFT 4000 (ERR_PROCHAND_GETMINE)

```
PROCESSHANDLE_GETMINE_ Error = #nn
```

Cause MEASFT encountered error *nn* in PROCESSHANDLE_GETMINE_

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4001 (ERR_PROCHAND_DECOMP)

`PROCESSHANDLE_DECOMPOSE_ File System Error = #nn`

Cause MEASFT encountered File System error *nn* in PROCESSHANDLE_DECOMPOSE_

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4002 (ERR_NOMEMORY_RECBUF)

`Not enough memory, bytes remaining = nnnn`

Cause MEASFT ran out of memory and reported *nnnn* bytes left. This is a MEASFT defect.

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4003 (ERR_NOMEMORY_SPLITBUF)

`Not enough memory for output buffer, bytes remaining = nnnn`

Cause MEASFT ran out of memory and reported *nnnn* bytes left. This is a MEASFT defect.

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4004 (ERR_NUM_SPFILES_ERR1)

`Internal error with variable, Num_spfiles = nn`

Cause MEASFT encountered an internal error with the internal Num_spfiles variable whose value is reported as *nn*. This is a MEASFT internal error; the text is the same as that for error 4005, but the underlying error differs.

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP

MEASFT 4005 (ERR_NUM_SPFILES_ERR2)

`Internal error with variable, Num_spfiles = nn`

Cause MEASFT encountered an internal error with the internal Num_spfiles variable whose value is reported as *nn*. This is a MEASFT internal error; the text is the same as that for error 4004, but the underlying error differs.

Effect The SPLIT command is not executed. MEASFT abends.

Recovery Contact HP

MEASFT 4007 (ERR_BAD_SPF_EOF)

`Output file EOF bad, expected = nnnnnn`

Cause MEASFT miscalculated the output file EOF. It expected *nnnnnn*, which is a number consisting of five to ten decimal digits. This is a MEASFT internal error.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4010 (ERR_BAD_MF_CONTEXT)

`Bad Measfile_context #nn`

Cause MEASFT internal logic error.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4012 (ERR_PARSE_KW)

`Internal error, bad keyword index = #nn`

Cause MEASFT internal logic error in parsing the command line. This is a MEASFT defect.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4013 (ERR_DIDNT_GET_INFO_DATA)

```
Internal error, didn't get the INFO data.
```

Cause MEASFT internal logic error in the INFO or SPLIT command. This is a MEASFT defect.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4014 (ERR_RB_STATE_ERROR)

```
Bad record buffer state, record #nnnnn
```

Cause MEASFT internal logic error in the INFO or SPLIT command occurred at record number *nnnnn*. This is a MEASFT defect.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

MEASFT 4015 (ERR_BAD_JOURNAL_SELECTION)

```
Internal error, bad Journal selection = #n
```

Cause MEASFT internal logic error in the INFO or SPLIT command. This is a MEASFT defect.

Effect The command is not executed. MEASFT abends.

Recovery Contact HP.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z