SMARTS POSIX Layer Configuration

The parameters described in this section are POSIX parameters only.

The POSIX start-up options for the SMARTS environment are specified as keyword parameters (so-called "sysparms"). These SYSPARM (z/OS) or SYSIPT (VSE) specifications must be entered according to established keyword coding conventions. See *Sysparm Format*

The description of parameters is organized under the following headings:

- SMARTS POSIX Log and Trace Parameters
- SMARTS POSIX Tracing Parameters
- SMARTS POSIX Recovery Parameters
- SMARTS POSIX Statistics Collection Parameters
- SMARTS POSIX Miscellaneous Parameters
- Standard CDI Definitions

SMARTS POSIX Log and Trace Parameters

Both the Logging and Tracing configurations are controlled by three parameters respectively. They define the ’size’ of the Data Collection structures. Data Collection is a SMARTS mechanism for buffering output in dataspaces to reduce the perfomance implications of outputting large volumes of Trace or Log records. If any of the following parameters are set to zero or not specified, Data Collection will not occur and all data will be output directly to the DD names specified, with possible performance degradation resulting.

LOG_DATA_COLL_ELEMENT_SIZE

Parameter	Use	Possible Values	Default
LOG_DATA_COLL_ELEMENT_SIZE	The size (in bytes) of a data element within the log data collection block.	16 - 32767	0

The element contains the data collection prefix area (DCPA) in the first 64 bytes; followed by the data collected by the user.

LOG_DATA_COLL_BLOCK_SIZE

Parameter	Use	Possible Values	Default
LOG_DATA_COLL_BLOCK_SIZE	The size of a block within the log data collection data space.	LOG_DATA_COLL_ELEMENT_SIZE - 32767	1024

The size of the block will determine how many elements it will contain. The more elements a block contains, the less IO required to harden the output, as the data is written out by block.

LOG_DATA_COLL_BLOCK_COUNT

Parameter	Use	Possible Values	Default
LOG_DATA_COLL_BLOCK_COUNT	Number of blocks in the log data collection data space.	1 - n where n * blocksize <= 2GB	8

The block count will determine how many blocks the data collection structure will contain. This in turn determines how many concurrent threads the data collections mechanism can process concurrently.

If statistics are being collected and the data is to be hardened (`STATISTICS_INCLUDE=LOG`), then the element and block values will be overridden by the length of the longest statistics block, if greater than the block value specified.

TRACE_DATA_COLL_ELEMENT_SIZE

Parameter	Use	Possible Values	Default
TRACE_DATA_COLL_ELEMENT_SIZE	The size (in bytes) of a data element within the trace data collection block.	16 - 32767	128

The element contains

- the data collection prefix area (DCPA) in the first 64 bytes; followed by
- the data collected by the user.

TRACE_DATA_COLL_BLOCK_SIZE

Parameter	Use	Possible Values	Default
TRACE_DATA_COLL_BLOCK_SIZE	The size of a block within the trace data collection data space.	TRACE_DATA_COLL_ELEMENT_SIZE - 32767	1024

The size of the block will determine how may elements it will contain. The more elements a block contains, the less IO required to harden the output, as the data is written out by block.

TRACE_DATA_COLL_BLOCK_COUNT

Parameter	Use	Possible Values	Default
TRACE_DATA_COLL_BLOCK_COUNT	Number of blocks in the trace data collection data space.	1 - n where n * blocksize <= 2 GB	8

The block count will determine how may blocks the data collection structure will contain. This in turn determines how many concurrent threads the data collections mechanism can process concurrently.

SMARTS POSIX Tracing Parameters

Tracing parameters are processed in the order in which they are entered. No effort is made to process all includes before excludes or vice versa.

SYSTEM_TRACE_LEVEL

Parameter	Use	Possible Values	Default
SYSTEM_TRACE_LEVEL	Granularity of tracing to be collected.	1 - 5	1

Five (5) levels of tracing are possible; level 1 provides the least amount of tracing information, and level 5 provides the maximum amount of tracing information.

Use the following guidelines to determine what to trace for a given trace level:

Level	Description
1	The minimum amount of information needed to identify why the trace occurred and the event in question. Only main events are traced. The trace information is formatted to fit on one print line. Use this level to gather trace information with a minimum of overhead.
2	Same as level 1 except that all events are traced.
3	Same as level 2 with additional trace records for each event that may include parameter lists and single values including pointers. Control blocks are not included.
4	Same as level 3 with additional trace records for each event that may include control blocks or parts of control blocks that are relevant to the trace event.
5	Same as level 4 with all relevant information related to the trace event: control blocks, buffers, and any other data that may be useful. This level will have a severe impact on system performance.

When the APSTRCE identifier is provided in a SMARTS job stream, the trace data collection mechanism attempts to open the file identified by APSTRCE and write unformatted trace data to it. The file is generally a blocked dataset with the ability to hold block-size/element-size records per block.

The element size determines the amount of data from a single request that the trace collection mechanism can handle. If the element size is set to 128 bytes, for example, the collection mechanism accepts a DCPA and up to 64 bytes of additional information. If the DCPALEN field value is greater than 64 bytes in this case, anything after the 64th byte of information in the additional data is not logged. Although the element size can be increased, the larger the element size, the fewer the elements that will fit into the trace buffer and the greater the impact on system performance.

When the identified APSTRCF is provided in the SMARTS job stream, the trace mechanism formats the provided DCPA and any additional data in a generic format and writes the formatted data to the dataset identified by APSTRCF. The trace logic must format and write this data immediately; thus if large amounts of data are traced, system performance slows significantly. Each additional piece of data to be written slows performance even more. You can manage the situation by writing the code that builds requests to the trace subsystem so that it properly restricts the amount of data that is traced.

TRACE_SYSTEM_INCLUDE

Parameter	Use	Possible Values	Default
TRACE_SYSTEM_INCLUDE	Specifies system trace options to include in trace.	see table	none

One trace option may be specified per parameter. To activate more than one option, the parameter must be specified multiple times:

```
TRACE_SYSTEM_INCLUDE = CFUNCTION
TRACE_SYSTEM_INCLUDE = CONDVAR
TRACE_SYSTEM_INCLUDE = MUTEX
```

Value	Description
CFUNCTION	Trace entry to and exit from each C function in the application running on SMARTS.
CONDVAR	Trace all activity in the SMARTS system related to condition variables.
MUTEX	Trace all activity in the SMARTS system related to mutex.
PTHREADS	Trace all activity in the SMARTS system related to pthreads.
INDEPENDENT_SOCKETS	Trace generic sockets activity.
STACK_DEPENDENT_SOCKETS	Trace low level stack requests.
CONTAINER_IO	Trace all internal IO calls to SAGIOS (inactive for BS2000).
ALL	Trace all of the above parameters.

TRACE_SYSTEM_EXCLUDE

Parameter	Use	Possible Values	Default
TRACE_SYSTEM_EXCLUDE	Specifies system trace parameters to exclude in the trace.	see tables and discussion for TRACE_SYSTEM_INCLUDE	none

TRACE_FUNCTION_INCLUDE

Parameter	Use	Possible Values	Default
TRACE_FUNCTION_INCLUDE	Include a specific function in the trace.	function name	none

The function name is case-sensitive.

A list of functions with tracing switched on is produced unless the list contains more than 50% of all functions. In that case, a list of the functions with tracing switched off is produced.

TRACE_FUNCTION_EXCLUDE

Parameter	Use	Possible Values	Default
TRACE_FUNCTION_EXCLUDE	Exclude a specific function from the trace.	function name	none

The function name is case-sensitive.

TRACE_GROUP_INCLUDE

Parameter	Use	Possible Values	Default
TRACE_GROUP_INCLUDE	Include a specific group of functions in the trace.	see table of groups | ALL	none

Value	Description
ALL	Switch tracing on for all functions.

TRACE_GROUP_EXCLUDE

Parameter	Use	Possible Values	Default
TRACE_GROUP_EXCLUDE	Exclude a specific group of functions from the trace.	table of groups | ALL	none

Value	Description
ALL	Switch tracing off for all functions.

Table of Tracing Groups

Group	Functions
ASYNC_IO	aio_cancel, aio_error, aio_fsync, aio_read, aio_return, aio_suspend, aio_write, lio_listio
DATABASE	dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store
DEVICE	grantpt, isatty, ptsname, unlockpt
FILE_ DIRECTORY	__check, access, basename, chdir, chmod, chown, chroot, close, closedir, creat, dirname, dlclose, dlerror, dlopen, dlsym, dup, dup2, fattach, fchdir, fchmod, fchown, fcntl, fdatasync, fdetach, fnmatch, fpathconf, fstat, fstatvfs, fsync, ftruncate, ftw, getcwd, getdtablesize, getwd, glob, globfree, lchown, link, lockf, lseek, lstat, mkdir, mkfifo, mknod, mkstemp, mktemp, nftw, open, opendir, pathconf, pwrite, read, readdir, readdir_r, readlink, readv, realpath, remove, rename, rewinddir, rmdir, seekdir, stat, statvfs, symlink, sync, telldir, truncate, umask, unlink, utime, utimes, write, writev, flockfile, pread, tempnam, tmpfile, tmpnam, ttyname, ttyname_r

Group	Functions
INTER_ PROCESS_COMMS	execl, execle, execlp, execv, execve, execvp, fork, ftok, pipe
INTERNAL	__aett, __eatt, __xlt, apslog, apstrace, ENVINIT, ENVTERM, EXTATTCH, EXTCDICHCK, EXTCDICNCL, EXTCDISLCT, EXTDEL, EXTDETCH, EXTFREEG, EXTFREET, EXTFSIOS, EXTGETG, EXTGETT, EXTLOAD, EXTMSG, EXTOPCMD, EXTPOST, EXTPPCBG, EXTPPCBS, EXTTRACE, EXTWAIT, EXTWAITL, hlli, SAGIOR
IO	cfgetispeed, cfgetospeed, cfsetispeed, cfsetospeed, clearerr, ctermid, cuserid, delenv, fclose, fdopen, feof, ferror, fflush, fgetc, fgetpos, fgets, fgetwc, fgetws, fileno, flockfile, fmtmsg, fopen, fprintf, fputc, fputs, fputwc, fputws, fread, freopen, fscanf, fseek, fseeko, fsetpos, ftell, ftello, ftrylockfile, funlockfile, fwide, fwprintf, fwrite, getc, getc_unlocked, getchar, getchar_unlocked, getmsg, getopt, getpass, gets, getsubopt, getw, getwc, getwchar, ioctl, isastream, optarg, pclose, poll, popen, pread, printf, putc, putc_unlocked, putchar, putchar_unlocked, putmsg, putpmsg, puts, putw, rewind, scanf, select, setbuf, setvbuf, snprintf, sprintf, sscanf, stdin, system, tcdrain, tcflow, tcflush, tcgetattr, tcgetsid, tcsendbreak, tcsetattr, ungetc, vfprintf, vprintf, vsnprintf, vsprintf, putwc, putwchar, swprintf, swscanf, tempnam, tmpfile, tmpnam, ttyname, ttyname_r, ungetwc, vfwprintf, vswprintf, vwprintf, wprintf, wscanf
JUMP	_longjmp, _setjmp, longjmp, setjmp, siglongjmp, sigsetjmp
LANGUAGE_ LOCALE	localeconv, nl_langinfo, setlocale
LOGGING	closelog, openlog, setlogmask, syslog
MATH	abs, acos, acosh, asin, asinh, atan, atan2, atanh, cbrt, ceil, cos, cosh, div, drand48, erand48, erf, erfc, exp, expm1, fabs, floor, fmod, frexp, gamma, hypot, ilogb, initstate, isnan, j0, j1, jn, jrand48, labs, lcong48, ldexp, ldiv, lgamma, log, log10, log1p, logb, lrand48, modf, mrand48, nextafter, nrand48, pow, rand, rand_r, random, remainder, rint, scalb, seed48, setstate, signgam, sin, sinh, sqrt, srand, srand48, srandom, tan, tanh, y0, y1, yn
MEMORY	brk, bzero, calloc, free, getpagesize, malloc, memccpy, memchr, memcmp, memcpy, memmove, memset, mlock, mlockall, mmap, mprotect, msync, munlock, munlockall, munmap, realloc, sbrk, shm_open, shm_unlink, shmat, shmctl, shmdt, shmget, valloc, bcmp, bcopy
MESSAGES	catclose, catgets, catopen, mq_close, mq_getattr, mq_notify, mq_open, mq_receive, mq_send, mq_setattr, mq_unlink, msgctl, msgget, msgrcv, msgsnd, perror, putmsg, putpmsg
MISCELLANEOUS	__environ, __errno, _assert, clrenv, confstr, getenv, iconv, iconv_close, iconv_open, putenv, qsort, swab, sysconf, ualarm, uname, usleep, wordexp, wordfree

Group	Functions
NETWORK_ SOCKETS	__h_errno, accept, bind, connect, endhostent, endnetent, endprotoent, endservent, gethostbyaddr, gethostbyname, gethostent, gethostid, gethostname, getnetbyaddr, getnetbyname, getnetent, getpeername, getprotobyname, getprotobynumber, getprotoent, getservbyname, getservbyport, getservent, getsockname, getsockopt, givesocket, htonl, htons, inet_addr, inet_lnaof, inet_makeaddr, inet_netof, inet_network, inet_ntoa, listen, ntohl, ntohs, recv, recvfrom, recvmsg, send, sendmsg, sendto, sethostent, setnetent, setprotoent, setservent, setsockopt, shutdown, socket, socketpair, takesocket
PROCESS	_exit, _spawn, atexit, exit, getegid, geteuid, getgid, getgroups, getlogin, getlogin_r, getpgid, getpgrp, getpid, getppid, getsid, getuid, nice, setegid, seteuid, setgid, setpgid, setpgrp, setregid, setreuid, setsid, setuid, spawnl, spawnle, spawnlp, spawnv, spawnve, spawnvp, tcgetpgrp, tcsetpgrp, ulimit, vfork, wait, waitid, waitpid

Group	Functions
PTHREAD	pause, pthread_atfork, pthread_attr_destroy, pthread_attr_getdetachstate, pthread_attr_getguardsize, pthread_attr_getinheritsched, pthread_attr_getschedparam, pthread_attr_getschedpolicy, pthread_attr_getscope, pthread_attr_getstackaddr, pthread_attr_getstacksize, pthread_attr_init, pthread_attr_setdetachstate, pthread_attr_setguardsize, pthread_attr_setinheritsched, pthread_attr_setschedparam, pthread_attr_setschedpolicy, pthread_attr_setscope, pthread_attr_setstackaddr, pthread_attr_setstacksize, pthread_cancel, pthread_cleanup_pop, pthread_cleanup_push, pthread_cond_broadcast, pthread_cond_destroy, pthread_cond_init, pthread_cond_signal, pthread_cond_timedwait, pthread_cond_wait, pthread_condattr_destroy, pthread_condattr_getpshared, pthread_condattr_init, pthread_condattr_setpshared, pthread_create, pthread_detach, pthread_equal, pthread_exit, pthread_getconcurrency, pthread_getschedparam, pthread_getspecific, pthread_join, pthread_key_create, pthread_key_delete, pthread_mutex_destroy, pthread_mutex_getprioceiling, pthread_mutex_init, pthread_mutex_lock, pthread_mutex_setprioceiling, pthread_mutex_trylock, pthread_mutex_unlock, pthread_mutexattr_destroy, pthread_mutexattr_getprioceiling, pthread_mutexattr_getprotocol, pthread_mutexattr_getpshared, pthread_mutexattr_gettype, pthread_mutexattr_init, pthread_mutexattr_setprioceiling, pthread_mutexattr_setprotocol, pthread_mutexattr_setpshared, pthread_mutexattr_settype, pthread_once, pthread_rwlock_destroy, pthread_rwlock_init, pthread_rwlock_rdlock, pthread_rwlock_tryrdlock, pthread_rwlock_trywrlock, pthread_rwlock_unlock, pthread_rwlock_wrlock, pthread_rwlockattr_destroy, pthread_rwlockattr_getpshared, pthread_rwlockattr_init, pthread_rwlockattr_setpshared, pthread_self, pthread_setcancelstate, pthread_setcanceltype, pthread_setconcurrency, pthread_setschedparam, pthread_setspecific, pthread_testcancel, pthread_kill, pthread_sigmask
PWD_GRP_ACC	endgrent, endpwent, endutxent, getgrent, getgrgid, getgrgid_r, getgrnam, getgrnam_r, getpmsg, getpwent, getpwnam, getpwnam_r, getpwuid, getpwuid_r, getutxent, getutxid, getutxline, pututxline, setgrent, setpwent, setutxent, ttyslot
REGULAR_ EXPRESSIONS	advance, compile, loc1, locs, re_comp, re_exec, regcmp, regcomp, regerror, regex, regexec, regexp, regfree, step
RESOURCES	getpriority, getrlimit, getrusage, setpriority, setrlimit
SCHEDULING	sched_get_priority_max, sched_get_priority_min, sched_getparam, sched_getscheduler, sched_rr_get_interval, sched_setparam, sched_setscheduler, sched_yield

Group	Functions
SEARCH	bsearch, hcreate, hdestroy, hsearch, insque, lfind, lsearch, remque, tdelete, tfind, tsearch, twalk
SEMAPHORE	sem_close, sem_destroy, sem_getvalue, sem_init, sem_open, sem_post, sem_trywait, sem_unlink, sem_wait, semctl, semget, semop
SIGNAL	abort, alarm, bsd_signal, kill, killpg, pthread_kill, pthread_sigmask, raise, sigaction, sigaddset, sigaltstack, sigdelset, sigemptyset, sigfillset, sighold, sigignore, siginterrupt, sigismember, signal, sigpause, sigpending, sigprocmask, sigqueue, sigrelse, sigset, sigstack, sigsuspend, sigtimedwait, sigwait, sigwaitinfo
STRING	a64l, atof, atoi, atol, bcmp, bcopy, crypt, ecvt, encrypt, fcvt, ffs, gcvt, index, l64a, rindex, setkey, strcasecmp, strcat, strchr, strcmp, strcoll, strcpy, strcspn, strdup, strerror, strfmon, strftime, strlen, strncasecmp, strncat, strncmp, strncpy, strpbrk, strrchr, strspn, strstr, strtod, strtok, strtok_r, strtol, strtoul, strxfrm, -wcsftime, wcscat, wcschr, wcscmp, wcscoll, wcscpy, wcscspn, wcslen, wcsncat, wcsncmp, wcsncpy, wcspbrk, wcsrchr, wcsrtombs, wcsspn, wcsstr, wcstod, wcstok, wcstol, wcstombs, wcstoul, wcswcs, wcswidth, wcsxfrm, wcsftime
TIME	asctime, asctime_r, clock, clock_getres, clock_gettime, clock_settime, ctime, ctime_r, daylight, difftime, ftime, getdate, getitimer, gettimeofday, gmtime, gmtime_r, localtime, localtime_r, mktime, nanosleep, setitimer, sleep, strptime, time, timer_delete, timer_getoverrun, timer_gettime, timer_settime, times, tzname, tzset, timer_create, strftime
USERCONTEXT	getcontext, makecontext, setcontext, swapcontext
WIDE_CHAR	btowc, iswalnum, iswalpha, iswcntrl, iswctype, iswdigit, iswgraph, iswlower, iswprint, iswpunct, iswspace, iswupper, iswxdigit, mblen, mbrlen, mbrtowc, mbsinit, mbsrtowcs, mbstowcs, mbtowc, putwc, putwchar, swprintf, swscanf, towctrans, towlower, towupper, wcrtomb, wctob, wctomb, wctrans, wctype, wcwidth, wmemchr, wmemcmp, wmemcpy, wmemmove, wmemset, wcscat, wcschr, wcscmp, wcscoll, wcscpy, wcscspn, wcslen, wcsncat, wcsncmp, wcsncpy, wcspbrk, wcsrchr, wcsrtombs, wcsspn, wcsftime, wcsstr, wcstod, wcstok, wcstol, wcstombs, wcstoul, wcswcs, wcswidth, wcsxfrm
XTI	t_accept, t_alloc, t_bind, t_close, t_connect, t_error, t_free, t_getinfo, t_getprotaddr, t_getstate, t_listen, t_look, t_open, t_optmgmt, t_rcv, t_rcvconnect, t_rcvdis, t_rcvrel, t_rcvreldata, t_rcvudata, t_rcvuderr, t_rcvv, t_rcvvudata, t_snd, t_snddis, t_sndrel, t_sndreldata, t_sndudata, t_sndv, t_sndvudata, t_strerror, t_sync, t_sysconf, t_unbind

TRACE_OUTPUT_START_AFTER

Parameter	Use	Possible Values	Default
TRACE_OUTPUT_START_AFTER	Start putting out trace data, after the specified number of trace records have been issued. This can be used as a mechanism for reducing the number of records output if a large and unwieldy output is anticipated.	Any numeric value	none

TRACE_OUTPUT_STOP_AFTER

Parameter	Use	Possible Values	Default
TRACE_OUTPUT_STOP_AFTER	Stop putting out trace data, after the specified number of trace records have been issued. This can be used as a mechanism for reducing the number of records output if a large and unwieldy output is anticipated.	Any numeric value	none

TRACE_CFUNC_PLIST

Parameter	Use	Possible Values	Default
TRACE_CFUNC_PLIST	Specify whether C function parameter list tracing is to be active or not.	YES ¦ NO	NO

TRACE_CFUNC_PARMS

Parameter	Use	Possible Values	Default
TRACE_CFUNC_PARMS	Specify the formats of parameters of a paticular C function to be traced. One invocation of this keyword is required for every different C function to be traced. The number of parameters to be traced is limited to 8 and only the 1st 25 bytes of the C functions name will be traced, so programmers should try to ensure that function names are unique within those 25 bytes, to avoid ambiguous trace output.	See table	none

Table of C Function Parameter Tracing Options

The values for this keyword are specified as follows:

```
TRACE_CFUNC_PARMS=('sample_function_name',p1,p2,p3,...,p8,RET=pr)
```

where p1, p2 ... etc represent the formats of the parameters passed and pr represents the format of the returned value. The possible values for these variables and their meanings are listed in the table below: Variable Meaning C Character type data. S Short Integer type data. I Integer type data. L Long Integer type data. G Long Long type data. F Floating Point type data. D Double Precision Floating Point type data. U Long Double Precision Floating Point type data. P Pointer type data. V Void Pointer type data. A Variable (unknown)

Variable	Meaning
C	Character type data.
S	Short Integer type data.
I	Integer type data.
L	Long Integer type data.
G	Long Long type data.
F	Floating Point type data.
D	Double Precision Floating Point type data
U	Long Double Precision Floating Point type data.
P	Pointer type data.
V	Void Pointer type data.
A	Variable (unknown)

So the example given above may have been coded as:

```
TRACE_CFUNC_PARMS=("function_passing_2ints_and_a_pointer",I,I,V,RET=I)
```

SMARTS POSIX Recovery Parameters

In general, the recovery parameters are always set to YES so that threads can be cancelled when SMARTS terminates. When the recovery parameters are set to NO, SMARTS does not terminate properly.

Use the NO value *only* for debugging purposes when requested to do so by your Software AG technical support representative.

ABEND_RECOVERY

Important:
Use this parameter only when requested to do so by your Software AG technical support representative.

Parameter	Use	Possible Values	Default
ABEND_ RECOVERY	Whether a recovery environment is established for a logical process in the SMARTS environment.	YES | NO	YES

NO means that SMARTS does not recover or cleanup when an ABEND occurs for a process.

THREAD_ABEND_RECOVERY

Important:
Use this parameter only when requested to do so by your Software AG technical support representative.

Parameter	Use	Possible Values	Default
THREAD_ABEND_RECOVERY	Whether a recovery environment is established for a pthread created in the SMARTS environment.	YES | NO	YES

NO means that SMARTS does not recover or cleanup when an ABEND occurs in a pthread.

SMARTS POSIX Statistics Collection Parameters

Statistics collection parameters are processed in the order in which they are entered, the last specification encountered takes precedence.

STATISTICS_INCLUDE

Parameter	Use	Possible Values	Default
STATISTICS_INCLUDE	Specifies resource to include in statistics collection.	See table	None

One resource may be specified per parameter. To activate more than one resource, the parameter must be specified multiple times:

```
STATISTICS_INCLUDE = CFUNCTION
STATISTICS_INCLUDE = CONDVAR
STATISTICS_INCLUDE = MUTEX
```

Value	Description	Default number of blocks allocated
CFUNCTION	Collect statistics for C and C++ functions.	100
PFUNCTION	Collect statistics for POSIX functions.	100
MUTEX	Collect statistics for Mutex processing.	20
CONDVAR	Collect statistics for Condition Variable processing.	20
FILE	Collect statistics about file usage.	10
SOCKET	Collect statistics about sockets usage.	10
STORAGE	Collect statistics about storage usage.	1
ALL	Collect all statistics.	N/A

A number may be included on the `STATISTICS_INCLUDE` statement to override the default number of blocks to be allocated for the given resource:

```
STATISTICS_INCLUDE = CFUNCTION(200)
STATISTICS_INCLUDE = CONDVAR(10)
```

This feature cannot be used when `STATISTICS_INCLUDE=ALL` is specified. As mentioned above, the size of the SMARTS log file block/element size is determined by the storage required by the statistics blocks, which is controlled by the number of blocks allocated. The maximum length that can be specified for the log file is 32K and data will be lost if the maximum is exceeded. Use the override facility to reduce this size.

STATISTICS_EXCLUDE

Parameter	Use	Possible Values	Default
STATISTICS_EXCLUDE	Specifies resource to exclude from statistics collection.	See tables and discussion for STATISTICS_INCLUDE	None

STATISTICS_OPTION

Parameter	Use	Possible Values	Default
STATISTICS_OPTION	Specifies how the data will be processed when flushed from the internal buffers.	See table	None

LOG	The data will be hardened using the SMARTS logging facility.
FORMAT	The data will be formatted and written to the APSSTAF dataset.

Both options may be specified, but on separate statements.

SMARTS POSIX Miscellaneous Parameters

ASCII

Parameter	Use	Possible Values	Default
ASCII	Whether ASCII runtime conversion is on and whether a translation table is to be used..	YES | NO | (YES,ttttttt) | (NO,ttttttt)	NO

SMARTS executables may be compiled as ASCII or EBCDIC executables. ASCII may be required, for example, in cases where ASCII dependencies are built into the processing algorithm(s).

ttttttt

translation table name e.g. CP1145. Applied translation tables are:

- CP1145 (Spanish)
- CP871 (Icelandic)
- CP273 (German)

The ASCII parameter value must match the way the executables were built. ASCII and EBCDIC executables may not be intermixed.

C_STACK_SIZE

Important:
Use this parameter only when requested to do so by your Software AG technical support representative.

Parameter	Use	Possible Values	Default
C_STACK_SIZE	Preallocates C stack storage for internal use.	0 - 4095M	200K

Note:
The value may be indicated in bytes, in kilobytes with a "K" modifier, or in megabytes with an "M" modifier; for example, 320,000 bytes may also be specified as 320K or 32M. The `C_STACK_SIZE` parameter is used to preallocate a storage area for internal use.

CDI_DRIVER

Parameter	Use	Possible Values	Default
CDI_DRIVER	Defines CDI protocols to SMARTS and specifies the modules whilch implement the required functionality.	see format below	none

CDI driver parameters:

```
CDI_DRIVER=('CDIparm1')
CDI_DRIVER=('CDIparm2')
CDI_DRIVER=('CDIparm3')
```

A separate CDI_DRIVER parameter is required for each CDI driver you want to use. The order of CDI drivers within the parameter specification does not matter. See the section *Standard CDI Definitions* for more information.

Each CDI protocol driver definition takes the following form:

```
protocol,module,key1=value1
```

- where

protocol	is the name of the CDI protocol being defined
module	is the name of the load module implementing this CDI protocol. This load module must be accessible to the POSIX server environment.
key1..n/value1..n	are keyword/value pairs specific to the CDI protocol driver.

For information about specifying the keyword/value pairs, refer to the implementation documentation for the relevant CDI protocol.

A default driver will always be loaded for the 'FILE' protocol. The driver loaded will be the default native file I/O driver for the relevant hardware platform

ENVIRONMENT_VARIABLES

Parameter	**Use**	**Possible Values**	**Default**
ENVIRONMENT_VARIABLES	Names the file containing global environment variable definitions for the POSIX server.	file-name (see format below)	no global environment variables

The file name uses URL-like notation as follows:

- z/OS: If the file is in the PDS A.B.C member (MEMBER), specify it as

  ```
  /a/b/c/member.ext
  ```

  (note that *.ext* is ignored)

- VSE: If the file is Library "A", Sublibrary "B", Member "C", Member Type "D", specify it as:

  ```
  /a/b/c.d
  ```

- All environments: If the file is a sequential file called X.Y.Z, specify it as

```
/x/y/z/
```

FLOATING_POINT

Parameter	Use	Possible Values	Default
FLOATING_POINT	Specify whether the SMARTS environment should use the binary floating point format internally (*IEEE*) or the hexadecimal floating point format (*HFP*)	IEEE|HFP	Depends on support available on hardware platform

If the hardware platform where SMARTS is running does not support the required instruction set for binary floating point operations (*IEEE*), the `FLOATING_POINT` parameter value will default to *HFP*. The default value should only be overridden if this is explicitly instructed in a product's installation notes.

Warning:
Mixing applications with IEEE and HFP floating point arithmetic causes unpredictable results from floating point operations.

HOSTS_FILE

Parameter	Use	Possible Values	Default
HOSTS_FILE	Names the file containing the TCP/IP host name and address table.	File name	No host name table

The file name uses the same URL-like notation as described for the parameter `ENVIRONMENT_VARIABLES`.

LOAD_DLL

Parameter	Use	Possible Values	Default
LOAD_DLL	Preloads DLL executables in the batch environment only.	1-8 character DLL name	none

The DLL executable name is available from the execution environment; for example, STEPLIB.

LOG

Parameter	Use	Possible Values	Default
LOG	Whether messages written to APSLOG are also written to the console.	LOG | OPERATOR	LOG

When OPERATOR is specified, all messages are written to both APSLOG and the operator console.

MESSAGE_CASE

Parameter	Use	Possible Values	Default
MESSAGE_CASE	Whether messages are translated to all uppercase characters before being sent to the console.	UPPER | MIXED	MIXED

Normally, SMARTS messages are written as a combination of upper- and lowercase characters.

MOUNT_FS

Parameter	Use	Possible Values	Default
MOUNT_FS	Specifies the mapping of file names (for example, on open function calls) to the underlying physical file container or file name.	see text	none

SMARTS files can be processed either directly to the underlying file system of the native operating system or to an intermediate level known as the portable file system (PFS). Access to the files within a PFS is transparent using the standard POSIX APIs.

Multiple PFS files are permitted as long as each file has a different protocol name and a different container. When using multiple PFS container files, it is necessary to indicate which physical files are to contain which logical files. The MOUNT_FS parameter is used in conjunction with the CDI_DRIVER parameter specifying the one or more PAANPFS drivers. See the section *Standard CDI Definitions* for more information.

The MOUNT_FS parameter has two subparameters: the first subparameter maps to the name of the PFS driver in the CDI_DRIVER parameter and the second subparameter maps to the logical file name as specified by the application program POSIX calls.

For example:

```
CDI_DRIVER=('PFS1,PAANPFS,CONTAINER=CIO://DD:PFS01')
CDI_DRIVER=('PFS2,PAANPFS,CONTAINER=CIO://DD:PFS02')

MOUNT_FS=('PFS1://','/usr/')
MOUNT_FS=('PFS2://','/misc/')
```

The above parameters identify two PFS file systems: /usr files map to the physical dataset specified by PFS1 and /misc files map to the physical dataset specified by PFS2.

To refer to (open) a file in PFS01, issue

```
f1=open("/usr/data",...)
```

Any other pathnames are assumed to map to the default protocol file://, which is the native operating system file system.

MOUNT_FS is not limited to PFS filesystems. If you set up the POSIX parameters as

```
CDI_DRIVER=('file,PAAMFSIO') Native z/OS File I/O
MOUNT_FS=('file://','/fs/')
```

- and then issue

```
open("/fs/saguk/kxo/reg4/", ...)
```

- you are referring to sequential dataset SAGUK.KXO.REG4 in the native filesystem.

NETWORKS_FILE

Parameter	Use	Possible Values	Default
NETWORKS_FILE	Names the file containing the TCP/IP network name table.	File name	No network name table

The file name uses the same URL-like notation as described for the parameter ENVIRONMENT_VARIABLES.

PROCESS_HEAP_SIZE

Parameter	Use	Possible Values	Default
PROCESS_HEAP_SIZE	Preallocates storage for internal use.		1008

Note:
The value may be indicated in bytes, in kilobytes with a "K" modifier, or in megabytes with an "M" modifier; for example, 320,000 bytes may also be specified as 320K or 32M.

The PROCESS_HEAP_SIZE parameter is used to preallocate a storage area for internal use.

PROTOCOLS_FILE

Parameter	Use	Possible Values	Default
PROTOCOLS_FILE	Names the file containing the TCP/IP protocol name table.	File name	No protocol name table

The file name uses the same URL-like notation as described for the parameter ENVIRONMENT_VARIABLES.

SECURITY_INTERFACE

Parameter	Use	Possible Values	Default
SECURITY_INTERFACE	Identifies the security subsystem to use.	DEFAULT | ESSG | EXIT	DEFAULT

Value	Description
DEFAULT	Default security actions are taken and no external security system is consulted. User and group database files must be provided in files "$SAG_RTS_ETC/passwd" and "$SAG_RTS_ETC/group". The files are similar to UNIX-based passwd and group files in structure.
EXIT	Set security by user exit.

SERVICES_FILE

Parameter	Use	Possible Values	Default
SERVICES_FILE	Names the file containing the TCP/IP services name table.	File name	No services name table

The file name uses the same URL-like notation as described for the parameter `ENVIRONMENT_VARIABLES`.

SYSTEM_ID

Parameter	Use	Possible Values	Default
SYSTEM_ID	A name that uniquely identifies the POSIX server instance.	1-8 character string	SysName

The specified string is included in all messages issued to the operator during the execution of the POSIX server (excluding some start-up and termination messages). It may also be used in the future by the POSIX server system to uniquely identify itself within a machine.

UNSUPPORTED_FUNCTION_LIST

Important:
Use this parameter only when requested to do so by your Software AG technical support representative.

Parameter	Use	Possible Values	Default
UNSUPPORTED_FUNCTION_LIST	Whether a list of unsupported functions is written during startup.	YES | NO	NO

VSE_PRINTER_SYSNO

Parameter	Use	Possible Values	Default
VSE_PRINTER_SYSNO	Optional. Specifies the "cuu" of the VSE printer to be assigned for SYSLST.	000-FFF	FEE

ZAP_LIST

Important:
Use this parameter only when requested to do so by your Software AG technical support representative.

Parameter	Use	Possible Values	Default
ZAP_LIST	Whether a list of applied ZAPs is written during startup.	YES | NO	NO

When YES is specified, a message is written to the log for each ZAP that has been correctly applied.

Standard CDI Definitions

SMARTS provides a number of standard definitions for communication driver interfaces (CDIs) to cover a standard set of functionality in each given environment.

Support for Console Processing (All Environments)

Support for console processing may be activated in any SMARTS environment using this CDI driver.

This driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('CONSOLE,PAANCNIO')
```

There are currently no parameters for this CDI driver.

Support for IBM z/OS File Subsystem

Support for IBM z/OS File Subsystem may be activated for z/OS only using this CDI driver.

The driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('file,PAAMFSIO,BLKSIZE=<nnnnn>,LRECL=<nnnnn>,
RECFM=<fm>,VOLSER=<vvvvvv>,PRIMARY=<nnnn>,SECONDARY=<nnnn>,
DIRECTORY=<nnnn>,PAD=<xxxxx>')
```

The following table describes the use of the configuration parameters this driver supports:

Parameter	Use	Possible Values	Default
BLKSIZE	Optional. Specifies the default block size to be used for a dataset created by this driver, if it is otherwise unspecified.	user-configurable	4096
LRECL	Optional. Specifies the default logical record length to be used for a dataset created by this driver, if it is otherwise unspecified.	user-configurable	4092
RECFM	Optional. Specifies the default record format to be used for a dataset created by this driver, if it is otherwise unspecified.	F, FB, FBA, U, V, VB, VBA	VB
VOLSER	Optional. Specifies the volume serial number of the default disk pack on which to place a dataset created by this driver, if it is otherwise unspecified.	user-configurable	Site specific
PRIMARY	Optional. Specifies the default primary quantity value, in cylinders, to be allocated for a dataset created by this driver, if it is otherwise unspecified. Refer to IBM documentation on the DD statement SPACE parameter for more details on the primary quantity value.	user-configurable	1
SECONDARY	Optional. Specifies the default secondary quantity value, in cylinders, to be used for a dataset created by this driver, if it is otherwise unspecified. Refer to IBM documentation on the DD statement SPACE parameter for more details on the secondary quantity value.	user-configurable	1
DIRECTORY	Optional. Specifies the default number of directory blocks to be allocated for a PDS created by this driver, if it is otherwise unspecified. Refer to IBM documentation on the DD statement SPACE parameter for more details on the directory value.	user-configurable	10
PAD	Optional. In the case of writing to a dataset of Fixed record length, it may be necessary to pad out records with a padding character. This parameter may be used to specify the padding character.	SPACE, NULL	NULL

Support for IBM VSE File Subsystem

Support for the IBM VSE file subsystem may be activated for VSE only using this CDI driver.

The driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('FILE,PAVVFSIO', BLKSIZE=<nnnnn>,LRECL=<nnnnn>,
RECFM=<fm>,PAD=<xxxxx>)
```

The following table describes the use of the single configuration parameter this driver supports:

Parameter	Use	Possible Values	Default
BLKSIZE	Optional. Specifies the default block size to be used for a dataset created by this driver, if it is otherwise unspecified.	user-configurable	3200
LRECL	Optional. Specifies the default logical record length to be used for a dataset created by this driver, if it is otherwise unspecified	user-configurable	80
RECFM	Optional. Specifies the default record format to be used for a dataset created by this driver, if it is otherwise unspecified.	F, FB, FBA, U, V, VB, VBA	FB
PAD	Optional. In the case of writing to a dataset of Fixed record length, it may be necessary to pad out records with a padding character. This parameter may be used to specify the padding character.	SPACE, NULL	NULL

Support for FSC BS2000 File Subsystem

Support for Fujitsu Siemens Computers BS2000 File Subsystem may be activated for BS2000 using one of two CDI drivers.

For the main file subsystem use:

```
CDI_DRIVER=('file,PA2MFSIO')
```

For the shared file subsystem use:

```
CDI_DRIVER=('file,PA2SFIO,SIOTSK=<xxxxx>')
```

SIOTSK is a mandatory field for the shared file IO task and needs to be assigned a user-configurable name.

Support for the Portable File System (z/OS)

Access to the files within a portable file system (PFS) is transparent using the standard POSIX APIs after it has been properly implemented.

Define the CIO CDI driver to support PFS:

```
CDI_DRIVER=('CIO,PAANCIO')
```

Multiple PFS files are permitted as long as each file has a different protocol name and a different container.

Allocate a container to store each PFS:

```
LRECL=BLOCKSIZE=4096
```

Completely initialize the container to contain x'00's.

Reference each container by a DDNAME in the JCL.

Establish a CDI driver for each container/PFS. For example:

```
CDI_DRIVER=('PFS1,PAANPFS,CACHESIZE=2000,LRECL=4096,CONTAINER=CIO://DD:PFS01')
CDI_DRIVER=('PFS2,PAANPFS,CACHESIZE=4000,LRECL=32768,CONTAINER=CIO://DD:PFS02')
```

Note that both drivers in the example specify the same module (PAANPFS) but different protocol names. The protocol name (PFSnn in the example) is a user-defined name up to 8 characters in length.

CACHESIZE

The CACHESIZE subparameter is used to specify the number of records (4k blocks) that will be cached on platforms whose underlying IO is based on the Container IO model. (All supported platforms except BS2000). There is an overhead associated with specifying this parameter and so it should only be used on containers for which a large volume of IO is anticipated. Generally the bigger the cachesize, the greater the reduction in physical IOs, but as this value is increased, the law of diminishing returns will apply, so users will have to experiment to determine the appropriate value for specific containers in a particular application environment.

LRECL

The LRECL subparameter is used to define the logical record size the application will see when using this container. It is possible to greatly reduce the number of IOs to a device by specifying this value to be equal to the size at which the application typically does its IO at. e.g. if an application routinely issues 8k IOs to a particular container, the number of physical IOs can be almost halved by specifying an LRECL subparameter of 8192 for the CDI driver defining the protocol for that container. The LRECL subparameter is actioned when the container is first initialized, so modifying the value after that point will have no effect. The value specified must be an integral multiple of the physical container LRECL (4k).

Map each container/PFS to a 'file system'. That is, identify the mapping files, directories, and subdirectories to the containers/PFSs. For example:

```
MOUNT_FS=('PFS1://','/registry/')
MOUNT_FS=('PFS2://','/tamino/')
```

In the above example, all pathnames beginning in /registry/ are mapped to the container/PFS defined by the protocol PFS1 and all pathnames beginning in /tamino/ are mapped to the container/PFS defined by the protocol PFS2. All other pathnames are mapped to the default protocol, which is

```
file://
```

- that is, the standard z/OS file I/O.

Support for the Portable File System (VSE)

The PFS options for VSE are the same as for z/OS. Please also document that PFS uses a default LRECL etc. of 32K specifically for TAMINO. Other applications should reduce the LRECL to 4096 or 8192 as this saves dead space in the PFS when short blocks are being written and ALSO save storage at run-time.

Support for IBM OE TCP/IP Stack (z/OS)

Support for IBM OpenEdition TCP/IP may be activated for z/OS only using this CDI driver.

The driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('TCPIP,PAAOSOCK)
```

Notes:

1. The userid where the job is running must have an OE segment defined.

Support for Connectivity Systems TCP/IP Stack (VSE)

Support for the Connectivity Systems TCP/IP stack may be activated for VSE only using this CDI driver.

The driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('TCPIP,PAACSOCK,BUFSZ=n,MINQ=n,MAXQ=n')
```

The following table describes the configuration parameters this driver supports:

Parameter	Use	Possible Values	Default
BUFSZ	Optional. Specifies the length of the send and receive buffers.	629 to 65535	1492
MINQ	Optional. Specifies the minimum queue length (or backlog) that may be specified by an application in a listen() call.	3 to 32767	5
MAXQ	Optional. Specifies the maximum queue length (or backlog) that may be specified by an application in a listen() call.	3 to 32767	30

Note:
If the application specifies a value outside the above range an error does not occur. Instead, the appropriate minimum or maximum value is used.

The minimum value for BUFSZ is derived from the length of the status information received from the CSI stack and may change with future releases. The above value relates to Version 1 Release 4.0C.

Send and receive buffers will be allocated for each potential connection in a listening socket queue. Using the MAXQ could reduce the storage usage.

Support for Inter Process Communications Pipes (All Environments)

Support for pipes and named pipes may be activated for all environments using this CDI driver.

The driver may be activated using the following CDI driver definition:

```
CDI_DRIVER=('pipe,PAANPIO,BLKSIZE=<nnnn>,NBLKS=<bbbb>')
```

The following table describes the configuration parameters this driver supports:

Parameter	Use	Possible Values	Default
BLKSIZE	Optional. Defines the internal block size used by the driver in bytes. This is the size of each storage buffer allocated by the driver for storing pipe data in. The buffers are chained up to the maximum of <bbbb> entries.	User-configurable	4096
LIMIT	Optional. Defines the maximum number of storage buffers to be allocated for any open pipe descriptor.	User-configurable	128

Under normal circumstances, no configuration parameter should be required. Use these only when directed to do so by Software AG’s support personnel.

Notes:

1. Named pipes can only be supported when associated with a mounted PFS container. They are not supported on native file systems.
2. Normal (unnamed) pipes are not dependent on any file system.