Com-plete

Configuring the SMARTS Environment

Version 6.8.2

July 2023

ADABAS & NATURAL

Document ID: APS-CONF-682-20230717

Table of Contents

Configuring the SMARTS Environment v

1 About this Documentation 1

  Document Conventions 2

  Online Information and Support 2

  Data Protection 3

2 Overview of Configuration Parameters 5

3 SMARTS Configuration Sources 9

4 Sysparm Format 11

5 SMARTS POSIX Layer Configuration 13

  SMARTS POSIX Log and Trace Parameters 14

  SMARTS POSIX Tracing Parameters 16

  SMARTS POSIX Recovery Parameters 24

  SMARTS POSIX Statistics Collection Parameters 24

  SMARTS POSIX Miscellaneous Parameters 26

  Standard CDI Definitions 33

6 Configuration of the SMARTS Server Environment 39

  SMARTS Server Configuration Parameters 41

7 SMARTS Global Environment Variables 59

  File Requirements 60

  File Processing 60

  Examples 61

8 Configuring Resources for SMARTS 63

Configuring the SMARTS Environment

This documentation provides configuration information for SMARTS environments. It also describes global environment variables that can be set for the whole SMARTS address space.

The configuration parameters are described under the following headings:

Overview of Configuration Parameters

SMARTS Configuration Sources

Sysparm Format

SMARTS POSIX Layer Configuration

SMARTS Server Environment Configuration

SMARTS Global Environment Variables

Configuring Resources

Configurable Tables

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Overview of Configuration Parameters

SMARTS POSIX Log and Trace Parameters

LOG_DATA_COLL_ELEMENT_SIZE
LOG_DATA_COLL_BLOCK_SIZE
LOG_DATA_COLL_BLOCK_COUNT
TRACE_DATA_COLL_ELEMENT_SIZE
TRACE_DATA_COLL_BLOCK_SIZE
TRACE_DATA_COLL_BLOCK_COUNT

SMARTS POSIX Tracing Parameters

SYSTEM_TRACE_LEVEL
TRACE_SYSTEM_INCLUDE
TRACE_SYSTEM_EXCLUDE
TRACE_FUNCTION_INCLUDE
TRACE_FUNCTION_EXCLUDE
TRACE_GROUP_INCLUDE
TRACE_GROUP_EXCLUDE
TRACE_OUTPUT_START_AFTER
TRACE_OUTPUT_STOP_AFTER
TRACE_CFUNC_PARMS
TRACE_CFUNC_PLIST

SMARTS POSIX Recovery Parameters

ABEND_RECOVERY
THREAD_ABEND_RECOVERY

SMARTS POSIX Statistics Collection Parameters

STATISTICS_INCLUDE
STATISTICS_EXCLUDE

STATISTICS_OPTION

SMARTS POSIX Miscellaneous Parameters

ASCII
CDI_DRIVER
ENVIRONMENT_VARIABLES
HOSTS_FILE
NETWORKS_FILE
PROTOCOLS_FILE
SERVICES_FILE
FLOATING_POINT
LOAD_DLL
LOG
MESSAGE_CASE
MOUNT_FS
PROCESS_HEAP_SIZE
SECURITY_INTERFACE
SYSTEM_ID
UNSUPPORTED_FUNCTION_LIST
VSE_PRINTER_SYSNO
ZAP_LIST

Standard CDI Definitions

Support for Console Processing (All Environments)
Support for IBM z/OS File Subsystem
Support for IBM VSE File Subsystem
Support for the Portable File System (z/OS)
Support for IBM OE TCP/IP Stack (z/OS)
Support for Connectivity Systems TCP/IP Stack (VSE)
Support for Inter Process Communications Pipes (All Environments)

SMARTS Server Configuration Parameters

ADABAS-BP
ADACALLS
ADADBID
ADALIMIT
ADAROLL
ADASVC
APPLYMOD
BUFFERPOOL
DUMPDSN
EOJ-VER
GLOBAL-MAXENQS

INIT-PGM
INSTALLATION
MAXENQS
MAXTASKS
MESSAGE-ID
PATCHAR
PROGRAMISD
RESIDENTPAGE
ROLL-BUFFERPOOL
SAVEPOOL
SAVEPOOL-ANY
SECSYS
SECSYS-APPL
SERVER
STARTUPPGM
TASK-GROUP
THREAD-GROUP
THSIZEABOVE
TIBTAB
TRACECLASS
TRACEOPTION
TRACETABLE
WORKLOAD-AVERAGE
WORKLOAD-MAXIMUM

3 SMARTS Configuration Sources

The configuration information for SMARTS consists of two parts:

- the SMARTS POSIX layer parameters, which apply to all environments where SMARTS runs, including the SMARTS server environment; and
- the SMARTS server environment configuration parameters, which apply only to the server environment. With Com-plete, Com-plete is the SMARTS server environment and the Com-plete sysparms are the SMARTS server environment configuration parameters.

All parameters must be stored in one or more text members, which must be defined in the startup JCL under DD name SYSPARM (VSE: DLBL name SYSIPT).

The parameters defined in the SYSPARM/SYSIPT dataset are read and processed during initialization.

POSIX parameters are valid for SMARTS client and server environments. SMARTS server parameters are *not* valid in client environments and will cause the system to display the warning message

```
Unknown keyword = xxxxxxxx
```

where *xxxxxxxx* is the keyword that was not recognized.

Which parameters need to be specified is product and installation dependent. A minimum POSIX layer configuration might look like this:

```
SYSTEM ID=Posix1
ASCII=NO
SYSTEM TRACE LEVEL=3
CDI DRIVER=('tcpip,PAAOSOCK')
CDI DRIVER=('file,PAAMFSIO,PAD=SPACE')
```

A minimum SMARTS server environment configuration might look like this:

```
WORKLOAD-AVERAGE=100
WORKLOAD-MAXIMUM=1000
DUMPDSN=<prefix>.&SYSNAME..&JOBNAME..D&YYMMDD..T&HHMMSS.
SERVER=(OPERATOR,TLINOPER)       operator command reader
SERVER=(POSIX,PAENKERN)               POSIX server definition
```

4 Sysparm Format

Sysparms must be entered according to established keyword coding conventions.

When read from SYSPARM/SYSIPT, each statement must begin in column one. A maximum of 80 characters per statement is allowed.

More than one sysparm is allowed per statement, but successive sysparms must be separated by a comma, and the statement itself must be terminated by a blank. For example:

```
KEYWORD1=value1,KEYWORD2=value2...,KEYWORD9=value9
```

Continuation statements are allowed: a statement in parentheses may be wrapped after a comma. For example:

```
KEYWORD=(value1,         comment: this statement is continued on the next line
      value2)
```

Multiple statements for the same keyword are permissible. Depending on the keyword, specifying the same keyword again may

- override a previous specification (example: SYSTEM_TRACE_LEVEL or PATCHAR); or
- add another member to a list (example: CDI_DRIVER or RESIDENTPAGE).

When entered as PARM parameters in z/OS or VSE, standard PARM entry conventions apply. Each keyword must be entered in its entirety in any given statement in the format:

```
KEYWORD=value
```

All keyword values may be either fully spelled out or abbreviated. Software AG recommends that you always use the full spelling. Abbreviations must consist of the minimum number of characters required to uniquely differentiate a given keyword from any other acceptable keyword.

If a keyword option is omitted, the default value takes effect. If column one of any statement contains an asterisk, that statement is treated as a comment.

For most keywords, the default value should normally be used.

5 SMARTS POSIX Layer Configuration

- SMARTS POSIX Log and Trace Parameters 14
- SMARTS POSIX Tracing Parameters 16
- SMARTS POSIX Recovery Parameters 24
- SMARTS POSIX Statistics Collection Parameters 24
- SMARTS POSIX Miscellaneous Parameters 26
- Standard CDI Definitions 33

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

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
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,

Group	Functions
	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
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
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,

Group	Functions
	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
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,

Group	Functions
	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	See table	none

Parameter	Use	Possible Values	Default
	function names are unique within those 25 bytes, to avoid ambiguous trace output.		

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,tttttttt) | (NO,tttttttt)	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).

tttttttt

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.

Caution: 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.

TCPIP

Parameter	Use	Possible Values	Default
TCPIP	Whether the TCP/IP interface is required.	YES | NO	YES

On VSE the TCP/IP interface is always initialized by default. Specify TCPIP=NO in order to disable it.

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

Parameter	Use	Possible Values	Default
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

Parameter	Use	Possible Values	Default
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 TCP/IP Stacks on z/VSE

APS 3.3.1 including fixpack 15 is the minimum level required for TCP/IP support

Which TCP/IP stack shall be used must be specified in the Smarts JCL:

```
// OPTION SYSPARM='NN' /* NN = stack ID
```

In order to use the Barnard Software (BSI) stack, LIBDEF PHASE must include the BSI sublib. (The CSI phases are located in PRD1.BASE which must be present anyway.)

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.

6 Configuration of the SMARTS Server Environment

- SMARTS Server Configuration Parameters 41

Start-up parameters are available to customize the execution of the SMARTS server environment. The start-up options, whether specified as PARM parameters or entered as statements read from SYSPARM (z/OS) or SYSIPT (VSE), are specified as keyword parameters (so-called "sysparms") and must be entered according to established keyword coding conventions. See **Sysparm Format**.

The sysparms are interpreted and processed by the PARM-processor module of the SMARTS server environment when the server environment is initialized.

Note: For z/OS and VSE systems, SMARTS server sysparms may be overridden during the initialization of the environment without updating the member in the partitioned dataset. For more information about specifying or overriding sysparm data, see Installation on z/OS and Installation on VSE.

For a standard z/OS installation, Software AG recommends that you define the size of the installation by setting the region size and one or both of the parameters

```
WORKLOAD-AVERAGE
WORKLOAD-MAXIMUM
```

and omit the configuration of the following parameters to SMARTS:

ADABAS-BP
BUFFERPOOL
ROLL-BUFFERPOOL
SAVEPOOL
SAVEPOOL-ANY
TASK-GROUP
TIBTAB
THREAD-GROUP

If one or more of these sysparms are set, these settings will be used.

The following parameters may be specified in a SMARTS server environment only; otherwise, the following warning message will be issued:

```
Unknown keyword = xxxxxxxx
```

where *xxxxxxxx* is the keyword that was not recognized.

SMARTS Server Configuration Parameters

ADABAS-BP

Sysparm	Use	Possible Values	Default
ADABAS-BP	Used to define the ADABAS buffer pool.	see text	The number of subpools to allocate will be decided internally. The size of the elements depends on the definitions made to create ADALCO .

This buffer pool is used for ADABAS interface work areas, which are acquired outside of the thread but in the key of the thread. This parameter enables users to determine the key(s) for which buffer subpools are built and the number of buffers in each subpool.

The format for the value is as follows:

```
ADABAS-BP=((no,key),(no,key) . . . (no,key))
```

- where

no	is the number of elements to allocate in the buffer subpool for this key. This must be greater than 1 and less than or equal to 8192.
key	is the storage protect key in which the buffer subpool will be allocated. This may be any number between 1 and 15. For z/OS, FACOM, and Hitachi systems, only keys 8 to 15 should be specified here.

By default, a subpool is built for keys 8 to 15. 8192 bytes are allocated for each subpool and the number of areas that can exist in each subpool is dependent on the size of the various ADALNK areas required.

Notes:

1. If an error is encountered in an ADABAS-BP system parameter, the whole line of code is ignored. Therefore, if there is no following ADABAS-BP specification in the system parameters, the defaults are in effect.
2. A subsequent specification of the ADABAS-BP system parameter totally overwrites a previous ADABAS-BP specification. Therefore, if the second specification is incorrect, the defaults again apply even if the first ADABAS-BP specification is correct.

Example:

```
ADABAS-BP=((20,9),(50,12),(100,8))
```

An ADABAS buffer pool is built with three subpools:

- the first is built in key 9 and has 20 elements;
- the second is built in key C(12) and has 50 elements; and
- the third is built in key 8 and has 100 elements.

ADACALLS

Sysparm	Use	Possible Values	Default
ADACALLS	The maximum number of incomplete ADABAS calls from an application before the SMARTS/ ADABAS interface rolls out the application.	see text	10

Note: This parameter is ignored if ADAROLL=NO is specified.

The format for the value is as follows:

```
ADACALLS={ n | (dbid,n) }
```

- where

n	is an integer between 1 and 32767.
dbid	is an ADABAS database ID. If 'dbid' is specified, ADACALLS applies only to calls directed to the specified database.

ADADBID

Sysparm	Use	Possible Values	Default	Required
ADADBID	The default database ID for ADABAS.	1-255	none	no

The value specified for ADADBID is used if the application program does not supply a specific database ID in the ADABAS control block. Refer to the *ADABAS Operations Manual* for a description of the use of the database ID.

ADALIMIT

Value	n|(dbid,n)
Default	32768 (=32*1024)

The maximum value that can be specified for n is 1,048,576 (=1024*1024).

If ADALIMIT=0 is specified, this forces no limit, overriding the default value.

ADAROLL

Sysparm	Use	Possible Values	Default
ADAROLL	The number of seconds the SMARTS server environment will wait for ADABAS calls before it rolls out the program making the call.	see text	see text

The format for the value is as follows:

```
ADAROLL={ n | (dbid,n) | ALWAYS | (dbid,ALWAYS) | NO | (dbid,NO) }
```

- where

n	is an integer representing the number of seconds that the SMARTS server environment will wait.
dbid	is an ADABAS database ID. If 'dbid' is specified, ADAROLL applies only to calls directed to the specified database.
ALWAYS	indicates that the program is always eligible for rollout.
NO	indicates that the program is never eligible for rollout.

By default, the SMARTS server environment dynamically calculates the optimum value for each database based on the statistics for the database. The starting value is ALWAYS; i.e., at the first ADABAS call, the program is always eligible for rollout. Then ADAROLL is calculated based on the average response time (A) using the following rule:

A < 0.05 sec	ADAROLL=0.1
0.05 sec < A < 0.5 sec	ADAROLL=2*A
A > 0.5 sec	ADAROLL=ALWAYS

Software AG recommends that you allow this parameter to default.

ADASVC

Sysparm	Use	Possible Values	Default
ADASVC	The decimal SVC number to be used when communicating with ADABAS.	see text	13

The format for the value is as follows:

```
ADASVC={ n | (dbid,n) }
```

- where

n	is an integer 201 to 255 for z/OS, and 1 to 110 for VSE.
dbid	is an ADABAS database ID. If 'dbid' is specified, ADASVC applies only to calls directed to this database.

By default (ADASVC=13), the interface to ADABAS version 5 or above is disabled. Programs issuing a call to ADABAS version 5 or above are terminated with ABEND code U0004.

APPLYMOD

Sysparm	Use	Possible Values	Default	Required
APPLYMOD	Include or remove a system-wide modification in/from the SMARTS session.	91 or 92	none	no

This sysparm must be used for FACOM systems. Software AG does not recommend using it in other operating system environments.

The format for the value is as follows:

```
APPLYMOD={ n | (n,NO) }
```

- where

n	is the applymod number, in this case either 91 or 92.
n,NO	indicates the removal of the 'n' applymod.

Possible Applymods

When an error other than an application program error occurs in SMARTS, a dump is normally scheduled by the SMARTS recovery processing. The dump is written to the SYSUDUMP, SYSABEND, or SYSMDUMP DD statement using normal OS rules.

- **Applymod 73: force operating system dump**
  Specify Applymod=73 only at the request of your support representative to force an operating system dump prior to recovery after an abend.

  This will cause a dump to be taken according to the installation dump options set for SMARTS. Please note, this Applymod is only required to produce additional diagnostics in an error situation.

  The installation could suffer severe performance problems, and large numbers of dumps written, if this Applymod is set for any length of time and therefore it should only be set at the request of your support representative.

- **Applymod 91: Use the OS SNAP function to write a dump**
  Applymod 92: Use the IEATDUMP or SDUMP function to write a dump
  By default, applymod 92 is in effect, ensuring that unformatted dumps are written to dynamically allocated datasets according to the dataset name pattern defined by parameter DUMPDSN= .

  This is the dump format expected by Software AG support when you send in a dump for problem analysis. Also, this is by far the fastest method of writing a dump. If you prefer to get abend dumps according to your SYSMDUMP, SYSUDUMP, or SYSABEND definitions, specify APPLYMOD=(92,NO) in order to turn off applymod 92.

  If you prefer to produce dumps using the SNAP function, specify APPLYMOD=91 *in addition*.

  > **Note:** SMARTS may write dumps for certain non-abend error situations also. These dumps cannot be written to SYSMDUMP, SYSUDUMP, or SYSABEND, therefore, they are always written using either IEATDUMP / SDUMP or SNAP.

  Note to FACOM users: Software AG recommends that you also use the default (SDUMP). Software AG support may ask you to format the dump using a batch job before sending it to Software AG.

BUFFERPOOL

Sysparm	Use	Possible Values	Default
BUFFERPOOL	Defines the parameter for building the general buffer pool.	see text	see text

The format for the value is as follows:

```
BUFFERPOOL={ esize,enum [,expnum] [,loc] }
```

- where

esize	Required. Determines the size of each individual element in this buffer subpool. The value is rounded up to the next multiple of 64.
enum	Required. Determines the number of elements of the specified 'esize' that will initially be built in the buffer subpool to be defined.
expnum	Optional. Determines the number of elements by which the buffer subpool is expanded if the primary 'enum' is not sufficient. The default 'expnum' value is "enum/4" with a minimum of "1". The 'expnum' value is affected by the amount of space required for preemptive expansion of the subpool. As not all requests can expand a subpool when it becomes empty, SMARTS requires preemptive expansion of the general buffer pool. The space required for preemptive expansion is calculated internally. When the space available in the subpool reaches that specified for preemptive expansion, the subpool is expanded by one quarter of the number of subpool elements, or 10, whichever is lower. The 'expnum' value must be equal to or greater than the figure used for preemptive subpool expansion. If the specified value is lower, it is forced to this figure.

loc	Optional. Determines where the buffer subpool elements are to be allocated. Valid values are BELOW, ANY, and DS: ■ **BELOW** the default; storage is to be allocated below the 16-megabyte line. ■ **ANY** available only on 31-bit-capable systems; storage can be allocated anywhere within the primary address space and is to be allocated above the 16-megabyte line under normal circumstances. ■ **DS** available only on ESA-capable systems; storage can be allocated within a data space.

For each correctly specified BUFFERPOOL parameter, a subpool is built in the general buffer pool from which all non-specific buffer pool requests are satisfied.

If the BUFFERPOOL parameter is not specified, or is specified one or more times and all are incorrect in the sense that they are unusable, SMARTS builds a default bufferpool with standard sizes and numbers of elements based on the size of your installation.

When at least one BUFFERPOOL parameter is accepted as valid, the default is not invoked. This means that the parameters are not merged.

DUMPDSN

z/OS only.

Sysparm	Use	Possible Values	Default	Required
DUMPDSN	A data set name pattern to be used for the dump data set when SMARTS writes a dump using the z/OS IEATDUMP service	Any valid data set name, use of system symbols is permitted. See IBM documentation of the macro IEATDUMP, parameter DSNAD=	If DUMPDSN= is not specified, SMARTS uses SDUMP (if running APF-authorized) or SNAP instead of IEATDUMP, otherwise no dump will be taken.	no

Software AG recommends that you do specify this parameter for non-APF authorized installations in order to avoid SNAP dumps.

For authorized installations, this is not necessary, because SDUMP can be used. SDUMPs are written to SYS1.DUMPxx (or substitute).

Note: The data set name must be one that the userID in effect for the SMARTS address space is permitted to allocate.

Example for fixed dump data set name:

```
DUMPDSN=DUMP.DATASET.NAME
```

Example using symbols:

```
DUMPDSN=DUMP.&jobname..D&YYMMDD..T&HHMMSS.
```

EOJ-VER

Sysparm	Use	Possible Values	Default	Required
EOJ-VER	The indicated character string must be entered as part of the EOJ operator command when SMARTS terminates.	1 to 8-character string	none	no

GLOBAL-MAXENQS

Sysparm	Use	Possible Values	Default
GLOBAL-MAXENQS	The maximum number of ENQs or LOCKs that can be outstanding from user programs in the SMARTS region or partition.	100-32767	1024

INIT-PGM

Sysparm	Use	Possible Values	Default
INIT-PGM	Specifies the name(s) of programs to be loaded by SMARTS at the end of initialization.	see text	none

The format for the value is as follows:

```

INIT-PGM={ name | (name1, name2, ..., namen) }
```

The programs named are called from the nucleus during startup in the order they are specified, executed in the SMARTS address space in SMARTS's key, and deleted after execution. If a program ABENDs, SMARTS initialization optionally continues.

The programs are internal to Software AG applications that run on SMARTS and are supported by this parameter for legacy reasons.

Otherwise, it is preferable to use the SERVER statement to obtain control during startup, termination, and for operator commands, if required.

INSTALLATION

Sysparm	Use	Possible Values	Default
INSTALLATION	A 1 to 8-position character string used as an installation identification name. The name may not contain a comma.	character string	********

MAXENQS

Sysparm	Use	Possible Values	Default
MAXENQS	The maximum number of z/OS ENQs or VSE LOCKs that may be outstanding for any one application program.	1-256	15

Each outstanding ENQ/LOCK resource held occupies 24 bytes plus the length of RNAME in the general buffer pool while the resource is held (whether it is held as SHR or EXCLUSIVE).

MAXTASKS

Sysparm	Use	Possible Values	Default
MAXTASKS	The maximum number of tasks to be used within a given SMARTS run.	n	254 (z/OS, Facom, Hitachi) 27 (VSE)

- where 'n' is the maximum number of tasks that will be allocated within task groups:

- For z/OS and MSP (FACOM), the number must be greater than zero and less than or equal to 254. This is a nominal maximum of 256 less the 2 SMARTS system tasks.
- For VSE, the number must be greater than zero and less than or equal to 27; that is, VSE maximum tasks = 32 less 5 SMARTS system tasks.

This parameter should be allowed to default unless there is a valid reason for restricting the number of tasks to be attached. The only mechanisms for attaching tasks are through the start-up parameters or through the TASKS operator command.

MESSAGE-ID

Sysparm	Use	Possible Values	Default
MESSAGE-ID	Value to be used as the system ID in the SMARTS message prefix.	x (see below) | INSTALLATION	patch character

SMARTS messages have a prefix with the format

```
pppgggnnnnx
```

- where

ppp	product ID (APS)
ggg	message group ID
nnnn	message number
x	system ID

By default, the patch character is used as the system ID (see the PATCHAR sysparm).

Specify MESSAGE-ID=INSTALLATION to use the installation ID instead of the patch character as the system ID.

PATCHAR

Sysparm	Use	Possible Values	Default
PATCHAR	Except for an asterisk (*), a character that uniquely identifies the running SMARTS server environment within the system.	<char> | *	*

- where '<char>' is any valid printable character except an asterisk (*).

If another SMARTS server environment with the same patch character is active, SMARTS is terminated during initialization.

The default patch character '*' (asterisk) allows multiple SMARTS server environments with this patch character to be active at the same time.

This character is important in two areas:

1. Every message sent to the console has the patch character of the issuing SMARTS server environment following the message-identifier; for example, RTSABS0006-2. Before the sysparms are processed, the default patch character is shown in all messages.
2. Data can be added to the profile system as being specific to a certain system. When the data is read, the system searches for data relating to the patch character of the running system before taking the global information. In this way, you can customize your sessions differently in different SMARTS server environments using the same SMARTS system dataset.

PROGRAMISD

Sysparm	Use	Possible Values	Default
PROGRAMISD	The number of in-storage directory (ISD) slots to be reserved for SMARTS online programs.	n	100

- where 'n' is an integer from 1 to 16 digits in length. The minimum value is 10.

Each program ISD entry occupies 128 bytes of page-fixed storage containing the disk address of an online program that has been or is executing. For a given ISD, the entries are dynamically altered to reflect the most current program usage based upon frequency of use.

RESIDENTPAGE

Sysparm	Use	Possible Values	Default
RESIDENTPAGE	The name of a program to be loaded and made resident when SMARTS is initialized.	program-name	none

This parameter is relevant only for Com-plete. In all other environments, all modules are assumed to be reentrant, and are loaded into the address space automatically at first reference.

The program must be fully reentrant. If it is not marked reentrant, a warning message is issued on the operator's console at SMARTS initialization time.

The program must reside in the COMPLIB chain (z/OS) or the LIBDEF search chain (VSE) of the SMARTS initialization procedure.

ROLL-BUFFERPOOL

Sysparm	Use	Possible Values	Default
ROLL-BUFFERPOOL	The size of the fixed roll buffer pool	(Esize,Eno,Expno,Loc)	not allocated

The values have the same meaning as for the BUFFERPOOL parameter, except for Loc. The following values are valid for the ROLL-BUFFERPOOL Loc:

BELOW	allocate the roll buffer pool below the 16MB line only
ANY	allocate the roll buffer pool either below or above the 16MB line
DS	(the default) allocate the roll buffer pool in a data space

SAVEPOOL

Sysparm	Use	Possible Values	Default
SAVEPOOL	The number of "savepool" entries to be allocated below the 16MB line.	n>=100	calculated by SMARTS depending on the configuration

SAVEPOOL is a critical parameter as these areas are used as base level save areas and can therefore not be expanded. If they are filled, SMARTS terminates abnormally.

SAVEPOOL-ANY

Sysparm	Use	Possible Values	Default
SAVEPOOL-ANY	The number of "savepool" entries to be allocated above the 16MB line.	n>=100	calculated by SMARTS depending on the configuration

It is important to carefully review the value specified for SAVEPOOL-ANY based on the usage of the system. When these areas run out, the system can continue to run using savepool entries allocated below the line; however, this wastes a valuable resource.

SECSYS

Sysparm	Use	Possible Values	Default
SECSYS	An alternate security subsystem to validate user IDs and passwords during logon.	NO | RACF | ACF2 | TOPSECRET | COMSEC,R|A|T	NO

The specified subsystem is interrogated to determine dataset access authority during utility processing. This parameter applies to z/OS.

SECSYS-APPL

Sysparm	Use	Possible Values	Default
SECSYS-APPL	The application name to be used for uniquely identifying this SMARTS nucleus to the external security system (see SECSYS).	name	SAG#RTS

SERVER

Sysparm	Use	Possible Values	Default
SERVER	Information that identifies a server to SMARTS.	server-information	none

- where the server information has the format

```
(serv-id , init-mod , p1 , p2 .... pn)
```

serv-id	is the ID for this server (1-8 chars)
init-mod	is the name of the initialization/termination routine
p1...pn	are parameters to be passed to the initialization routines

Specifying the SERVER parameter causes SMARTS to build a server directory entry (SDE) for the specified server and pass control to the initialization routine specified to initialize the server.

STARTUPPGM

Sysparm	Use	Possible Values	Default
STARTUPPGM	Specifies the name(s) of one or more SMARTS application programs to be invoked at the end of initialization.	see text	none

The format for the value is as follows:

```
STARTUPPGM={ name | (name1, name2, ..., namen) }
```

These programs are scheduled in the order in which they are specified to execute in SMARTS server threads once the system has initialized. These application programs execute as attached tasks under SMARTS's user ID and can use all SMARTS functionality.

Sufficient batch or free TIBs must be available in SMARTS's TIBTAB to accommodate the number of programs specified.

Note: Each process running in a SMARTS server environment has a control block called TIB associated with it. The TIB contains identifying information such as a one- to five-digit terminal identification number (TID) and a one- to eight-character terminal information block name (TIBNAME). Either the TID or TIBNAME may be used to specify a single process.

TASK-GROUP

Sysparm	Use	Possible Values	Default
TASK-GROUP	A group comprising one or more tasks, available when SMARTS is started.	(grp,num,priority,maxq)	(DEFAULT,num)

- where

grp	Required. The name of the task group being defined. The default task group is DEFAULT.
num	Required. The number of tasks to be allocated in the task group. This value must be greater than 1 and less than 254 (z/OS) or 27 (VSE). The default number of tasks is calculated dynamically based on the size of the installation.
priority	the priority to be assigned to the operating system task, which is attached for z/OS and MSP (FACOM) systems only. This parameter is accepted under VSE, but has no meaning. Valid values are 0-255; the default is 248. '255' is the priority at which the task-dependent service processor task is running. Without the ADABAS high performance environment (HPE), this is '250'. While '255' is accepted, the task will in fact only be given a priority of '250'.
maxq	The maximum number of TIBs (default 16) expected on this task group's work queue at the same time. Under normal circumstances, the default should be adequate. When there are problems and it is not, a secondary Last In First Out (LIFO) queue is used so that no work is lost. The normal queue is First In First Out (FIFO), which ensures that work is done in the order in which it is received. This is why the LIFO queue is only used as a secondary backup.

Important: For SMARTS, only the TASK-GROUP DEFAULT is available. Software AG strongly recommends that you use the default definition. If other products running on SMARTS require changes to the defaults or allow the definition of their own TASK-GROUPs, that will be indicated in the relevant documentation.

Notes:

1. A maximum of 8 task groups may be defined.
2. Task-group names are converted to uppercase prior to being processed; therefore, a parameter entered in lowercase is treated as, and appears in, uppercase letters.
3. If more than one specification appears for a task group, the last valid specification is used.
4. The task group DEFAULT must always exist in the system. If it is not explicitly defined by the installation, the task group is built by the system with the default values.
5. Note that the total number of tasks to be attached must not exceed the MAXTASKS specification. This is not checked until the task groups are being built; however, exceeding the value leads to task-group allocation errors as against parameter errors.

Examples:

```
TASK-GROUP=(DEFAULT,4)
```

The DEFAULT task group is allocated with four attached tasks, the default priority, and the default maximum queue size specification.

```
TASK-GROUP=(DEFAULT,4,200)
TASK-GROUP=(TASK-GRP,4,150)
```

The DEFAULT task group is allocated with four attached tasks with a priority of 200 and the default maximum queue size specification. A second group called TASK-GRP is also allocated with three attached tasks, a priority of 150, and the default maximum queue size specification.

THREAD-GROUP

Sysparm	Use	Possible Values	Default
THREAD-GROUP	A thread group containing one or more thread subgroups and threads, to be available when SMARTS is started.	see below	see below

The format for the value is

```
(grp,(sub,size,num,cpu,real,key),...,(sub,size,num,cpu,real,key))
```

- where

grp	Required. The name of the thread group being defined.
sub	The name of the subgroup being defined. If a subgroup name is specified more than once for the same group, the last valid specification is used when parameter processing has been completed.
size	Required. The amount of storage in kilobytes to be allocated for each thread below the line. A valid value is between 8 kilobytes and 1 megabyte.
num	The number of threads to be allocated in the thread subgroup. The value must be greater than 1 and less than 4096. Generally, this subparameter is required. It can be omitted for one (and only one) thread subgroup in the address space; in this case, the number of threads to be allocated for the subgroup is calculated dynamically by SMARTS based on the size of the installation.
cpu	The CPU time in seconds (default 0.00) that a user program can use in the thread subgroup for one SMARTS transaction. This value may be entered as an integer or to a level of hundredths of seconds using the 'n.nn' format. If a 0 is provided as the CPUTIME for a thread subgroup, no CPU limit is placed on programs running in the associated threads.
real	The wait time in seconds (default 0.00) for the thread subgroup, after which a message is issued to the console if the user program has not given up control of its thread. This value may be entered as an integer or to a level of hundredths of seconds using the 'n.nn' format. If 0 is specified, elapsed time is not checked for the thread subgroup.
key	The key (default M) in which the threads within the subgroups are allocated: ■ **M** The thread keys are a mixture of user keys excluding the key in which SMARTS is running. ■ **N** No storage protection is implemented and all threads run in the same key as SMARTS. **Note:** The user may also specify a value in the range 1 to 15 inclusive to allocate a thread to that key explicitly.

The default value is

```
THREAD-GROUP=(DEFAULT,($DEFAULT,8,num))
```

- where "num" is calculated dynamically based on the size of the installation.

Important: For SMARTS, only the THREAD-GROUP DEFAULT is available. Software AG strongly recommends that you use the default definition. If other products running on SMARTS require changes to the defaults or allow the definition of their own THREAD-GROUPs, that will be indicated in the relevant documentation.

Notes:

1. A maximum of 8 thread groups may be defined.

2. A maximum of 8 subgroups can be allocated per thread group. The subgroups may be defined on one line or on different lines. When a second THREAD-GROUP statement is used, the same group name must be specified to relate the subgroup entries.
3. Thread group and subgroup names are converted to uppercase prior to being processed; therefore, a parameter entered in lowercase is treated as, and appears in, uppercase letters.
4. If more than one specification appears for a thread subgroup of a thread group, the last valid specification is used.
5. The amount of storage specified on the THSIZEABOVE sysparm is allocated above the line for each thread defined as a result of the THREAD-GROUP sysparm.
6. The thread group DEFAULT must always exist in the system. If it is not explicitly defined by the installation, the thread group is built by the system with the default values. If it is defined, the system ensures that a thread subgroup with a thread size at least as large as that required by DEFAULT is allocated. If not, the system allocates an additional subgroup for the group. If too many subgroups have been defined, the last one defined is overwritten to allow for the default specification.
7. The keyword data is processed from left to right. If more than one thread subgroup is defined on one line and the line contains an error, even if an error message is issued for the line, any subgroups processed up to the error are still accepted. That is to say, if the first subgroup is correct and the second is not, an error message is issued but the first thread subgroup is defined while the second and subsequent specifications in the same statement are ignored.

Examples

```
THREAD-GROUP=(DEFAULT,(SMALUTIL,80,3),(BIGUTIL,300,2,5,9,15))
```

This allocates the DEFAULT thread group with two subgroups:

- the first subgroup called SMALUTIL contains three threads with 84K below the line and takes the defaults for CPUTIME, REALTIME, and the protectkey to be allocated to the thread.
- the second subgroup called BIGUTIL contains two threads with 304K below the line, has a maximum CPUTIME of 5 CPU seconds, a REALTIME value of 9 seconds, and each thread has a storage protectkey of 15.

The following sets of sysparms defines exactly the same thread subgroups:

```
THREAD-GROUP=(DEFAULT,(SMALUTIL,80,3),(BIGUTIL,300,2,5,9,15))
THREAD-GROUP=(DEFAULT,(SMALUTIL,40,8),(BIGUTIL,300,2,5,9,15))
THREAD-GROUP=(DEFAULT,(SMALUTIL,80,3))
```

The following sets of sysparms defines exactly the same thread subgroups in two thread groups, one called DEFAULT and the other called EXTRAGRP:

```
THREAD-GROUP=(DEFAULT,(SMALUTIL,80,3))
THREAD-GROUP=(EXTRAGRP,(BIGUTIL,300,2,5,9,15))
THREAD-GROUP=(EXTRAGRP,(SMALUTIL,80,3))
THREAD-GROUP=(DEFAULT,(BIGUTIL,300,2,5,9,15))
```

THSIZEABOVE

Sysparm	Use	Possible Values	Default
THSIZEABOVE	The amount of storage above the 16 MB line, in multiples of 1024 bytes, to be allocated to each thread.	n	1024

TIBTAB

Sysparm	Use	Possible Values	Default
TIBTAB	The location and size of the TIB table to be built when SMARTS is initialized.	DYNnnnnn | ANYnnnnn	ANYnnnnn

- where

DYN	is the table to be built below the 16MB line
ANY	is the table to be built above the 16MB line
nnnnn	is the number of TIBs. The maximum is 32767. For the default value, "nnnnn" is calculated dynamically based on the size of the installation.

Note: Each process running in a SMARTS server environment has a control block called TIB associated with it. The TIB contains identifying information such as a one- to five-digit terminal identification number (TID) and a one- to eight-character terminal information block name (TIBNAME). Either the TID or TIBNAME may be used to specify a single process.

TRACECLASS

Sysparm	Use	Possible Values	Default
TRACECLASS	The class of trace event to be included in (or excluded from) the SMARTS trace table.	class | (class,OFF)	QTIB

- where 'class' is one of the following valid trace classes:

GENERIC	used for support purposes
QTIB	TIB queue management
OP	application program requests
FIXBPOOL	fixed-length buffer pool operations
ROLL	roll-processing events
RESOURCE	resource manager get/free
DISPATCH	dispatcher events

The option '(class,OFF)' indicates exclusion of the specified class.

TRACEOPTION

Sysparm	Use	Possible Values	Default
TRACEOPTION	The specified trace option is active for this execution of SMARTS.	option	no options active

The valid trace options are as follows:

ABEND	The trace continues to run during SMARTS abnormal termination. (Normally the trace stops recording at the first indication of termination.) Use only when required by Software AG support personnel.
EXTENDED	Trace processing uses the extended form of the trace record. Use only when requested by support personnel to find specific information as it decreases the number of trace records that can be held in a trace buffer.

TRACETABLE

Sysparm	Use	Possible Values	Default
TRACETABLE	The size of the SMARTS trace table, which is used to trace events occurring within the SMARTS system.	n | nK	8K

The TRACETABLE sysparm can be a valuable tool for problem resolution.

The minimum size of the trace table is 8K.

TRACETABLE=0 indicates that no tracing is performed.

WORKLOAD-AVERAGE
WORKLOAD-MAXIMUM

The WORKLOAD-AVERAGE parameter specifies a normal workload value, and the WORKLOAD-MAXIMUM parameter specifies a maximum workload value. SMARTS uses these values togther with the region sizes above and below the 1bMB line to configure itself.

These parameters are not required, but tuning them may improve performance.

Sysparm	Use	Possible Values	Default
WORKLOAD-AVERAGE	The average number of parallel processes expected to run in SMARTS	1-32767	WORKLOAD-MAXIMUM divided by 4
WORKLOAD-MAXIMUM	The maximum number of parallel processes expected to run in SMARTS	1-32767	50 if WORKLOAD-AVERAGE is not specified, otherwise WORKLOAD-AVERAGE times 4

Example:

```
WORKLOAD-AVERAGE=50
WORKLOAD-MAXIMUM=400
```

7 SMARTS Global Environment Variables

- File Requirements 60
- File Processing 60
- Examples 61

The SMARTS server system makes it possible for you to specify global environment variables for the SMARTS address space. These variables are returned to a program issuing the 'getenv' function for any given environment variable and enable a system-wide specification of any given variable. If the same variable has been set in a local process using the 'putenv' function, this is returned and the global version of the variable is ignored.

File Requirements

For z/OS environments, the file containing these variables

- has a record format of F, FB, or VB.
- has a valid record length. Software AG recommends a record length small enough for editors to handle. A record length of 256 is sufficient for most needs.
- is identified on the ENVIRONMENT_VARIABLES configuration parameter of SMARTS.

For VSE environments, the file containing these variables

- is a member in a VSE library.
- is identified on the ENVIRONMENT_VARIABLES configuration parameter of SMARTS.

For other environments, see the documentation for the particular Software AG application product that uses SMARTS.

File Processing

The contents of the file are processed as follows:

1. Each record in the file is read. Only one global environment variable may be specified per line.
2. The start of the variable name is the first nonblank character in the record.
3. The end of the variable name is the next '=' sign or blank found following the first nonblank.
4. If there is no '=' sign or no data follows the '=' sign, the environment variable is defined but has no value; that is, it is a null-terminated string with null as the first character.
5. Comments are allowed and are specified by an asterisk (*) in column 1.
6. To establish the value, SMARTS searches from the end of the record to find the first nonblank. The data after the equals sign to this point is treated as the variable. It is not possible to specify comments on these lines. For this reason, Software AG recommends that you not use numbered datasets such as those produced by TSO/ISPF to avoid interference with the values assigned to environment variables.

7. If the string starts with an apostrophe and ends with an apostrophe, the apostrophes are omitted in the environment variable but all data between them (including blanks, apostrophes, etc.) form part of the environment variable.
8. If the string starts with the value "X'" (that is, the character X followed by an apostrophe) and ends with an apostrophe, the data between them is treated as a hexadecimal value and must therefore be 0 to 9 or A to F. Note that 'a' to 'f' are treated as invalid hexadecimal data.
9. If the same variable name is specified more than once, the last one in the file is the active value for the variable after initialization.

Examples

Following are a number of examples of global variables:

```
MyVariable=This is my variable string
```

A request to getenv for MyVariable returns a pointer to 'This is my variable string' (without apostrophes).

```
QuotedVariable='This is my quoted variable string      '
```

A request to getenv for QuotedVariable returns a pointer to 'This is my quoted variable string ' (without the apostrophes).

```
NullVariable=
```

A request to getenv for NullVariable returns a pointer to '' (without apostrophes).

```
HexVariable=X'AABBCCDD'
```

A request to getenv for HexVariable returns a pointer to the hexadecimal value 'AABBCCDD'.

```
NotHexVariable=x'AABBCCDD'
```

A request to getenv for NotHexVariable returns a pointer to the string 'AABBCCDD' (without the apostrophes).

8 Configuring Resources for SMARTS

The primary consideration is the amount of storage made available to the POSIX server in the address space, whether in batch or in the SMARTS server environment. Basically, the larger the address space, the more requests that can be concurrently serviced using attached tasks and address space storage.

The SMARTS server automatically calculates its optimal configuration based on

- the amount of storage available in the address space; and
- the expected average and maximum workload as indicated by the WORKLOAD-AVERAGE and WORKLOAD-MAXIMUM parameters.

For VSE environments, Software AG recommends that you use the parameter rather than the WORKLOAD-AVERAGE and WORKLOAD-MAXIMUM parameters.