Adabas SOA Gateway

SMARTS Installation and Operation

Version 2.6.1

November 2016

ADABAS & NATURAL

This document applies to Adabas SOA Gateway Version 2.6.1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: APS-IO-261-20161123

Table of Contents

Preface

This documentation provides information on how to install and operate Software AG's Multiple Architecture Runtime System (SMARTS).

The information on installing and operating SMARTS is structured as follows:

Introduction	Overview of the SMARTS runtime environment its system requirements.
Installation on OS/390	Describes the SMARTS installation procedure on OS/390 systems
Installation on VSE/ESA	Describes the SMARTS installation procedure on VSE/ESA systems
Installation on VM/CMS	Describes the SMARTS installation procedure on VM/CMS systems
Initialization and Termination	Initialization and termination of the Posix and SMARTS server environment
Parameter Configuration	Explanation of SMARTS parameters and possible values.
Operator Command Processing	Communication Driver Interface (CDI) & SMARTS server environment operator commands
Support and Maintenance	Reporting problems / problem resolution / maintenance
ConfigurationTables	Translation & reference tables

1

SMARTS Environments, Features and Requirements

This chapter contains a brief overview of the Software AG Multiple Architecture Runtime System (SMARTS) and the requirements to run it.

The SMARTS Runtime Environment

SMARTS supports multiple platforms and environments on each platform. It implements a standard runtime environment and supports standard C library functions such as standard I/O, sockets, pthreads, and semaphores.

Although primarily a C runtime environment, SMARTS extensions provide access as well to the facilities from other programming languages including Assembler, COBOL, and PL/1.

SMARTS operates in two distinct environments: server and client.

The SMARTS Server Environment

SMARTS supports a high performance environment for running server components on the different supported platforms: OS/390, VSE/ESA, VM/CMS, MSP (FACOM), and BS2000.

The SMARTS server environment provides a multitasking architecture that uses blocks of storage called 'threads' in which SMARTS application programs can run. By ensuring that all application storage is in contiguous blocks of the address space, the server environment is in a position to move dormant applications out of the address space to make room for more active users. If or when such a dormant application becomes active again, it is simply moved back into the address space and dispatched again.

The SMARTS server environment

- uses its multitasking structure to make full use of operating system subtasks to drive the system CPUs.
- uses contiguous storage threads to give it more control over the applications' storage areas, thus enabling it to handle more applications running in the one address space.
- shares the underlying operating system subtasks between users and is thus not subject to restrictions on the number of processes that can be concurrently supported.
- uses a state-of-the-art buffer pool management technique that ensures consistent path lengths, no fragmentation of storage areas, and expansion and contraction of the storage areas within the address space.
- uses a state-of-the-art resource manager that ensures the shortest path length possible for the serialization of resources. The technique uses only machine instructions unless it is necessary to wait for a resource.

- is ready now for 24-by-7 operation as the nucleus does not need to be cycled for any internal reasons and the buffer manager and resource manager can handle higher than expected loads subject to the resources being available in the address space.

The SMARTS Client Environment

On the supported platforms, SMARTS-based applications are supported in a number of so-called client environments: batch, OS/Transaction Server (CICS), Com-plete (version 6.1 or above), and UTM. In all cases, the support enables client applications to use the SMARTS SDK library functions:

Server Environment	Supported Client Environments
OS/390	Batch, TSO, OS/TS, IMS/DC, IMS/Batch, Com-plete
VSE/ESA	Batch, OS/TS, Com-plete
MSP (FACOM)	Batch, Com-plete
VM/CMS	CMS Command line programs
BS2000	Batch/TIAM, UTM

In general, the client environments support everything that the server environment supports; however, Software AG recommends that you avoid running heavy duty applications in client environments. For example, a 'pthreaded' application should not be run in an OS/TS environment, even though it is functionally possible.

Supported TCP/IP Stacks

- IBM OpenEdition TCP/IP available in supported OS/390 and z/OS releases.
- VSE/ESA version 2.5 or above using the Connectivity Systems TCP/IP 4 VSE Service Pack E.

  Note: This requires Adabas SVC 7.4.1 . The prerequisites described in *VSE/ESA Prerequisites* in the Release Notes of EntireX for VSE/ESA must be fulfilled. If you are using Entire Net-Work, version 5.8.1 or above is required.

- MSP/EX from Fujitsu using the TISP TCP/IP stack.
- VM/ESA using the standard IBM stack available on VM.

2 Installation of SMARTS Client on CICS

Software AG recommends that you keep unmodified copies of all materials distributed or created as part of the installation process. This may assist with problem diagnosis later.

The installation procedure is described under the following headings:

- The Installation Tape
- Generic Installation Procedure
- Installing the SMARTS Server Environment
- Installing the CICS Client Environment
- Installation Verification
- Where Next ?

The configuration parameters are described in *Configuring the SMARTS Environment*.

The Installation Tape

Note: While you are free to rename the datasets, the dataset names used in this section are used consistently throughout the product documentation to ensure clarity.

- Tape Contents
- Copying Contents of the Tape to Disk

Tape Contents

The following table lists the product datasets and what the dataset contains:

Dataset	Contains ...
APSvrs.LOAD	all load modules required by SMARTS
APSvrs.SRCE	all JCL, sample source members, and SMARTS macros.

Datasets Created during the Installation Process

Dataset	Contains ..
APSvrs.USERSRCE	source members
APSvrs.USERLOAD	load modules for all SMARTS environments
APSvrs.CICSLOAD	CICS client environment

Sample Members Copied to the APSvrs.USERSRCE Dataset

The following table lists the sample members copied to the user source dataset during the installation process. These must be modified before being used:

Member	Contains ..
RJANPARM	sample server environment parameters.
RJBNINS1	sample JCL to copy TLINOS to an authorized dataset.
RJBNPROC	sample procedure to run the SMARTS server environment.

Copying Contents of the Tape to Disk

The cartridge is delivered with a standard Software AG Tape Creation Report indicating the datasets, formats, and sizes of each dataset on the cartridge.

To create the mainframe datasets, load the datasets using either of the following:

- the IEBCOPY utility for PDS datasets
- the IEBGENER utility for sequential datasets.

Generic Installation Procedure

The following steps are required for OS/390 environments.

Step 1: Allocate and Initialize User PDS Datasets

- To create and initialize the datasets required for SMARTS, modify the sample job in member RJBNINS2 on the APSvrs.SRCE dataset to suit your installation's environment, and run it to create the appropriate datasets.

  This job also copies all modifiable members from the APSvrs.SRCE dataset to the newly created APSvrs.USERSRCE dataset in order to retain all APSvrs.SRCE members as delivered.

Step 2: Specify the TCP/IP Configuration

a **Modify the CDI_DRIVER Parameter**

In the SMARTS environment, the `CDI_DRIVER` parameter is used to specify CDI (communication driver interface) protocol definitions. If TCP/IP communications are required, a definition for protocol 'tcpip' is used.

Modify the `CDI_DRIVER` parameter of the sample configuration member PXANCONF .

The following section provides a sample of the `CDI_DRIVER` parameter specification for an IBM OE TCP/IP stack. Refer to the chapter *Configuring the SMARTS Environments* for information about the parameters that can be specified.

For an IBM OE TCP/IP Stack

```
CDI_DRIVER=('tcpip,PAAOSOCK') ↩
```

Note: The userid used for the job running the SMARTS environment with the IBM OE TCP/IP Stack driver configured must have an OE segment defined in the RACF (or compatible) profile.

b **Customize the SMARTS TCP/IP host name and address table**

If your TCP/IP stack on OS/390 does not support host name/host address lookup (DNS), SMARTS uses a local address table that mimics the DNS functionality.

Use the sample host name parameter member PXANHOST in APSvrs.SRCE and customize to suit your needs. When customizing the local table, define:

- any host names and addresses that will be accessed from within the SMARTS server partition and
- the host where the local SMARTS server is executing.

For example, for a local host with name LOCAL and IP address 127.0.0.1 and a remote host with name REMOTE and IP address 157.189.160.95, you should specify:

```
127.0.0.1 LOCAL AF_INET 157.189.160.95 REMOTE AF_INET ↩
```

Verify and if necessary add or modify the following parameter in the members RJBNPROC and PXANCONF to point to the PXANHOST member:

```
HOSTS_FILE=file://APSvrs/SRCE/PXANHOST ↩
```

c **Review the Input and Output Datasets**

In all environments, SMARTS requires output DD statements for STDOUT, and STDERR and an input DD statement for STDIN.

- If the output DD statements are not provided, they default to a SYSOUT dataset;
- If the input DD statement is not provided, it defaults to an empty default dataset.

Software AG recommends that you allow these DD statements to default unless the output needs to be directed in some way, or input is required on STDIN for the application in question.

Installing the SMARTS Server Environment

This section explains the additional steps required to run the SMARTS server environment.

Step 1: Authorize the SMARTS Server Environment (if necessary)

SMARTS itself does not require to be run as an authorized user, however, some SMARTS based products may have such a requirement generally or if you want to use certain features. To run SMARTS as an authorized user, proceed as follows:

a To achieve this, place module TLINOS in an authorized dataset. Use one of the following alternatives:

Alternative 1:

Copy the module to a library that is already authorized and use the library as the STEPLIB for SMARTS. Sample JCL to copy the module is contained in member RJBNINS1.

Note: If you do not use RJBNINS1, you must ensure that whatever method you use to copy the module retains the correct AMODE/RMODE attributes and authorization code for the module.

Alternative 2:

Authorize the distributed SMARTS load library permanently by adding an entry for the dataset in member IEAAPFnn of library SYS1.PARMLIB, where "nn" is the suffix used at your site.

Note: This requires an IPL of the system. You can avoid having to perform an IPL for the initial installation if you have a product installed that allows automatic authorization of the dataset.

b Modify the SMARTS Server Procedure: Specify only the authorized dataset as STEPLIB, and add a COMPLIB load library concatenation containing all other load libraries, including the SMARTS load library:

```
//STEPLIB DD DISP=SHR,DSN=APF.AUTHORIZED.LIBRARY
//*
//COMPLIB DD DISP=SHR,DSN=APSvrs.USERLOAD
//APSULIB DD DISP=SHR,DSN=APSvrs.LOAD
```

Step 2: Customize the SMARTS Server Procedure

The SMARTS server environment is initialized or started by invoking a procedure or running a job with the appropriate JCL. The example JCL member RJBNPROC illustrates a typical SMARTS procedure that might be used for an OS/390 installation.

a Add the procedure to the appropriate library

When first installing the SMARTS server environment, add the procedure RJBNPROC to the installation's system procedure library SYS1.PROCLIB or to a user-defined procedure library.

The example member contains the following JCL:

```
 //RJBNPROC PROC APSPARM=RJANPARM,
//           OPARM=,
//           REG=32M
//*********************************************************************
//*                                                                   *
//* SMARTS Server Environment Startup Procedure for OS/390            *
//*                                                                   *
//*********************************************************************
//IEFPROC EXEC PGM=TLINOS,
//         PARM='&OPARM;',
//         REGION=&REG;,TIME=1440,DPRTY=(14,14)
//*
//STEPLIB DD DISP=SHR,DSN=APSvrs.USERLOAD
//APSULIB DD DISP=SHR,DSN=APSvrs.LOAD
//*
//SYSPARM DD DISP=SHR,DSN=APSvrs.USERSRCE(&APSPARM;)
//SYSPRINT DD SYSOUT=X ↩
```

b Prepare the procedure

During the installation process, either

- alter the supplied RJBNPROC procedure to suit your requirements and copy the resulting procedure to an installation procedure library.

c Invoke the procedure

Invoke the SMARTS procedure either from the operator's console using

- an OS/390 START command

```
 S RJBNPROC,... ↩
```

- or an OS/390 batch job

```
 //SMARTS JOB ............
//IEFPROC EXEC RJBNPROC,... ↩
```

In either case, you need to understand the DD statement functions and the use of available start-up options to implement the features of the SMARTS server environment.

The following sections use the above sample JCL as a basis for defining the SMARTS server environment initialization procedure for OS/390.

d **Check required and optional DD statements**

The required and optional DD statements in the above procedure are described in more detail below:

STEPLIB	Required . Identifies the authorized load library in which the SMARTS server environment OS/390 start-up module TLINOS resides. No other SMARTS server environment modules need to be available in this library. This dataset is only referenced once during initialization and therefore its placement is not an issue.
COMPLIB	Required . Identifies the PDS library concatenation that effectively becomes the STEPLIB for the duration of the run. This means that all modules loaded during the execution of the SMARTS server environment are loaded from this concatenation.
SYSMDUMP	Optional . Identifies where the OS/390 control program should write a formatted dump if an ABEND occurs or the SMARTS server environment requests such a dump. If not specified, no support can be provided for any problem as no diagnostic information is available. Software AG recommends that you specify SYSMDUMP instead of the SYSABEND or SYSUDUMP DD statements, as more information is available for diagnosis if a problem should occur. Note as well that a SYSMDUMP usually completes in less time. Estimates for the size of the dataset specified by this DD statement must be made according to the IBM documentation.
SYSPRINT	Optional . Identifies where statistics are printed at termination or using the STATS operator command. If not specified, this DD is allocated using dynamic allocation as a sysout dataset.
SYSPARM	Required . Identifies the file or library member in which the desired SMARTS server environment system parameters are to be found. If the SMARTS server environment is to be periodically stopped and started in order to test various start-up options, the symbolic parameter &RTSPARM can be used to identify the member containing the desired options. However, you can use the PARM option at start-up time as a short-term test of a specific option. The available system parameters are described in Configuring the SMARTS Environments.

e **Check the symbolic parameters**

You can modify the SMARTS procedure, including the format and use of the symbolic parameters. However, the symbolic parameters indicated below are generally sufficient to meet the needs of most installations:

&OPARM	Specifies a character string that is passed to the SMARTS server environment control program using the PARM sysparm (see the chapter Configuring the SMARTS Environments for use of this feature).
&RTSPARM	Specifies the member name in the library identified by the DD name SYSPARM that contains the control statements specifying the start-up parameters (sysparms). You can create multiple members to allow tailoring of SMARTS server environment initialization to meet the specific conditions defined by the control statements.

Step 3: Add SMARTS Server Configuration Statements

- Add the appropriate SMARTS server configuration parameters prior to starting the SMARTS server environment.

  In particular, select a dataset name pattern for DUMPDSN= that will allow the user in effect for the SMARTS address space to allocate names.

  The configuration parameters are described in *Configuring the SMARTS Environment*.

  For parameters relevant to your application, refer also to the documentation for the software that runs on SMARTS.

Installing the CICS Client Environment

Step 1: Modify the CICS Procedure

a **Add the LOAD datasets to the DFHRPL concatenation**

Add the APSvrs.LD00 dataset to the DFHRPL concatenation in the procedure.

b **Add the SYSPARM dataset**

Add a "`//SYSPARM DD`" statement for the SMARTS POSIX configuration.

This dataset is required to define the runtime characteristics of your POSIX environment.

For more information, see *SMARTS Configuration Sources*.

c **Add a dataset for environment variables**

Add a "`//CONFIG DD`" statement for the dataset containing the environment variables required by your POSIX applications within CICS.

This DD name is specified by the `ENVIRONMENT_VARIABLES` parameter in `SYSPARM`, which defaults to “CONFIG”.

For more information, see the section *SMARTS POSIX Miscellaneous Parameters*, ENVIRONMENT_VARIABLES.

d **Provide a DFHZNEP node error program**

If the installation already has a DFHZNEP node error program in use, modify it to invoke the SMNE transaction under the conditions detailed in the model assembler program PACNZNEP, supplied in the *APSvrs.SRCE* Tape file. .

If the installation does not have a DFHZNEP node error program in use, use the supplied model program PACNZNEP to create one.

Step 2: Define Transactions to CICS

- Four standard transactions are required:
  - TDSP, which is used internally by SMARTS;
  - SMGO, which is used to initialize the POSIX server;
  - SMEX, which may be used to terminate the POSIX server;
  - SMNE, which is invoked by the CICS Node Error Program to provide SMARTS cleanup functionality.

  The standard transactions must be defined to run the following Assembler programs:

  - TDSP to run PACNSTRT. The name TDSP is mandatory as it is used internally by SMARTS. Define this Transaction with TASKDATALOC(BELOW) and TASKDATAKEY(USER)..
  - SMGO to run PACNKERN. The name SMGO is mandatory as it is used internally by SMARTS. Define this Transaction with TASKDATALOC(BELOW) and TASKDATAKEY(CICS)..
  - SMEX to run PACNKERX. SMEX is a suggested transaction name. Define this Transaction with TASKDATALOC(BELOW) and TASKDATAKEY(CICS).
  - o SMNE to run PACNNEP. The name SMNE is mandatory, as it is used internally by SMARTS. Define this Transaction with TASKDATALOC(BELOW) and TASKDATAKEY(CICS)..

Step 3: Define Transactions to CICS

a Autoinstall should also be activated. Autoinstall can be activated by setting the SIT parm :

```
PGAIPGM=ACTIVE
```

b The following Assembler language programs must be defined to CICS.

- PACNABEX with EXECKEY(CICS)
- PACNKERN with EXECKEY(CICS)
- PACNKERX with EXECKEY(CICS)
- PACNNEP with EXECKEY(CICS)
- PACNPCEX with EXECKEY(USER)
- PACNSMGO with EXECKEY(CICS)
- PACNSTRT with EXECKEY(CICS)

The following Assembler Programs must be defined to CICS as follows:

- RAANPARM, define with "RELOAD=YES", and EXECKEY(USER)

Step 4: Define Programs to the Program List Tables

a **POSIX Initialization Table**

The program list table PLTPI is used to automatically initialize SMARTS POSIX at CICS startup:

In the DFHPLTxx for the PLTPI, insert PACNSMGO as a second phase PLT program.

b **POSIX Shutdown Table**

The program list table PLTSD is used to automatically terminate SMARTS POSIX at CICS shutdown:

In the DFHPLTxx for the PLTSD, insert PACNKERX as a first phase PLT program.

Installation Verification

1) Initialization Messages of SMARTS

A successfull initialization of SMARTS can be verified in the CICS protocol output and looks similar to this messages:

```
+APSPSX0015-* POSIX V271 Build 030212 Patch level=2 Initialization in prog
+APSPSX0004-* Module 'L$CPRSU' Loaded
+APSPSX0050-SysName CDI FILE protocol initialized
+APSPSX0066-SysName Trace level = 1
+APSPSX0068-SysName No System Tracing enabled
+APSPSX0069-SysName No functions are being traced
+APSPSX0036-SysName Global environment variables processed successfully
+APSPSX0026-SysName Sockets Initialization successful
+APSPSX0050-SysName CDI TCPIP protocol initialized
+APSPSX0064-SysName Trace DataSpace Initialised, ESIZE=0:BSIZE=0:NBLKS=0
+APSPSX0065-SysName Log DataSpace Initialised, ESIZE=0:BSIZE=0:NBLKS=0
+APSPSX0008-SysName SMARTS SERVER V271 System initialized, nucleus size 56
+PACNKERN - POSIX INTERFACE INITIALISED.
```

2) Installation Verification

Verification of communcation between CICS using Natural and EntireX Communicator can be performed by logging on the Natural library SYSETB and proceeding the Natural Tutorial .

Example – CICS:

```
N411  RCA=(BROKER),RCALIAS=(BROKER,EXAAPSC)
```

3) CICS Shutdown verification

A successful shutdown of CICS using the CEMT P,SHUT command will present `PACNKERX - POSIX INTERFACE TERMINATED` on the terminal that issued the shutdown command. This verifies that control has been given to the SMARTS termination routine.

Where Next ?

You have now installed the SMARTS software. You can continue now with the installation of the application that is to run on SMARTS.

Note that the configuration procedure of the application that runs on SMARTS may instruct you to modify some of the configuration parameters of SMARTS.

3

Installation on VSE/ESA

This document describes procedures for installing SMARTS under VSE/ESA.

Software AG recommends that you keep unmodified copies of all materials distributed or created as part of the installation process. This may assist with problem diagnosis later.

- The Installation Tape
- Installing the SMARTS Server Environment
- Installing the VSE CICS Client Environment
- Where Next ?

The Installation Tape

The installation tape is described under the following headings:

- Tape Contents
- Copying Contents of the Tape to Disk

Tape Contents

The installation tape contains the following files:

Dataset	Contains . . .	
APSvrs.LIBR	Library	SAGLIB
	Sublibrary	APSvrs SMARTS components: phases, objects, JCL, sample source members, and macros.

Sample JCL and Source Members

The following table lists the sample source and job members in the APSvrs sublibrary. These must be modified before being used:

Member	Contains ..
APSSIP.J	Sample job to initialize the SMARTS system adapter.
PXANCONF.P	The POSIX server configuration parameters.
PXANHOST.P	Sample parameter file to customize the TCP/IP host name and host address table.
RJANPARM.P	Sample server environment parameters.
RJBNINS1.J	Sample job to restore the SMARTS libraries.
RJBNINS2.J	Sample job to allocate the SMARTS VSAM Dump file.
RJBNINS3.J	Sample job to allocate and restore the MSHP History file.
RJBNINS4.J	Sample job to allocate the SMARTS VSAM Trace file.
RJBNPROC.J	Sample procedure to run the SMARTS Server Environment.

Copying Contents of the Tape to Disk

Step 1: Restore the SMARTS library

- Use the following JCL, supplied in the APSvrs sublibrary as member RJBNINS1.J, to restore the SMARTS library:

```
* $$ JOB JNM=APSREST,CLASS=c,DISP=d,LDEST=(,uid)
* $$ LST CLASS=c,DISP=d
// JOB APSREST  --- Restore APS Library ---
/*
/* ================================================================= *
/* Restore APS Library                                               *
/* ================================================================= *
/*
// PAUSE
// ASSGN  SYS006,cuu
/*
// DLBL   SAGLIB,'saglib.library',0,SD
// EXTENT ,vvvvvv,1,0,ssss,ttt
/*
// MTC    REW,SYS006
// MTC    FSF,SYS006,nn
/*
// EXEC   LIBR
   RESTOR SUB=SAGLIB.APSvrs   : SAGLIB.APSvrs   -
          R=Y TAPE=SYS006
/*
/&
$$ EOJ
```

Installing the SMARTS Server Environment

Step 1: Installing the SMARTS Server Environment

- Use the sample JCL member APSSIP.J in the APSvrs sublibrary to initialize the SMARTS system adapter. Customize the various parameters to suit your needs.

  You must execute this JCL before you execute the SMARTS server to avoid initialization errors.

  Software AG recommends that you add this JCL to the $ASIPROC so that the SMARTS system adapter is initialized automatically at IPL time.

Step 2: Allocate the SMARTS VSAM Dump file

- Use the sample JCL member RJBNINS2.J in the APSvrs sublibrary to allocate and restore the SMARTS VSAM Dump file. Customize the various parameters to suit your needs.

  The file allocated in this step will be assigned in the SMARTS server **start-up JCL**.

Step 3: Allocate the SMARTS Trace file

- Allocate either an SD or VSAM/ESDS file for the SMARTS trace file. The APSvrs sublibrary contains a sample JCL to allocate the SMARTS Trace file as a VSAM/ESDS file (member RJBNINS4.J. Customize the various parameters to suit your needs.

  The file allocated in this step will be assigned in the SMARTS server **start-up JCL**.

Step 4: Allocate the SMARTS History file

- Use the sample member RJBNINS3.J in the APSvrs sublibrary to allocate and restore the MSHP History file. Customize the various parameters to suit your needs.

  This file will be required when applying maintenance to SMARTS.

Step 5: Customize the SMARTS TCP/IP host name and address table

a Because the current TCP/IP stack on VSE/ESA does not support host name/host address lookup (DNS), SMARTS uses a local address table that mimics the DNS functionality.

Use the sample host name parameter member PXANHOST.P in the APSvrs sublibrary and customize to suit your needs. When customizing the local table, define:

- any host names and addresses that will be accessed from within the SMARTS server partition and
- the host where the local SMARTS server is executing.

For example, for a local host with name LOCAL and IP address 127.0.0.1 and a remote host with name REMOTE and IP address 255.89.65.90:

```
127.0.0.1      LOCAL  AF_INET
255.89.65.90 REMOTE AF_INET
```

b Verify and if necessary add or modify the following parameter in the members RJBNPROC.J and PXANCONF.P to point to the PXANHOST.P member:

```
HOSTS_FILE=/SAGLIB/APSvrs/PXANHOST.P
```

Step 6: Edit the SMARTS Server start-up JCL

- Modify the sample SMARTS server start-up JCL member RJBNPROC.J in the APSvrs sublibrary to suit your installation naming conventions.

  The example SMARTS start-up JCL below is typical for a VSE/ESA environment and serves as the basis for the various descriptions and explanations that follow:

```
* $$ JOB JNM=RJBNPROC,CLASS=c,DISP=d,LDEST=(,uid)
* $$ LST CLASS=c,DISP=d
// JOB RJBNPROC  --- SMARTS Startup ---
/*
/* ================================================================= *
/* SMARTS Startup                                                    *
/* ================================================================= *
/*
// OPTION PARTDUMP,NOSYSDMP,LOG
/*
/* Dump file for APS -------------------------------------
/*
// DLBL   COMDMP,'aps.vsam.dumpfile',,VSAM,CAT=ccccccc  Step 2
/*
/* Tracing and logging -----------------------------------
/*
// ASSGN  SYSnnn,DISK,VOL=vvvvvv,SHR
// DLBL   APSTRCE,'aps.trace.file',0,SD                 Step 3
// EXTENT SYSnnn,vvvvvv,1,0,ssss,ttt
/*
/* Libdefs -----------------------------------------------
/*
// LIBDEF PHASE,SEARCH=(SAGLIB.APSvrs,                             +
               SAGLIB.WALvrs)
/*
// UPSI   00000000
// EXEC   TLINSP,SIZE=AUTO
*
* -------------------------------------------------------
* Example SYSPARMS for the SMARTS SERVER Environment (SSE) (RJANPARM.P)
* -------------------------------------------------------
*
 INSTALLATION=SMARTS                   Installation ID
 THREAD-GROUP=(DEFAULT,($DEFAULT,20,2,0,0,N))
 WORKLOAD-MAXIMUM=050
*
 SERVER=(OPERATOR,TLINOPER,TLSPOPER)   Operator Communications Server
 SERVER=(POSIX,PAENKERN)               POSIX                  Server
*
* -------------------------------------------------------
* Example SYSPARMS for the SMARTS POSIX  Environment (PSX) (PXANCONF.P)
* -------------------------------------------------------
```

```
*
 ENVIRONMENT_VARIABLES=/SAGLIB/APSvrs/ENVVARS.P
 HOSTS_FILE=/SAGLIB/APSvrs/PXANHOST.P                      Step 5
 LOG=OPER                              Messages to Operator Console
 SYSTEM_ID=SMARTS                      System ID
*
/*
// EXEC   LISTLOG
/*
/&
* $$ EOJ
```

For a description of the SMARTS SYSPARMS, see *Configuration Parameters*. For parameters relevant to your application, refer to the configuration documentation for the software that runs on SMARTS.

Installing the VSE CICS Client Environment

Step 1: Modify the CICS Procedure

a Add the `APSvrs` sublibrary to the LIBDEF search chain. This must be placed before any other Software AG product sublibrary.

b Add `DLBL` and `EXTENT` statements for STDOUT and STDERR if using.

c Add `DLBL` and `EXTENT` statements for APSTRCE.

d Declare the location of the `SYSPARM` dataset via the `PARM=` option of the `EXEC` statement. For example, if the system parameters are to be found in a sublibrary member the statement could look like this:

```
// EXEC DFHSIP,PARM='SYSPARM(/PROD/CICS/SYSPARM.P)',......
```

e Add `// OPTION SYSPARM='nn'` statement where nn is the id of the *TCP/IP* for VSE job to be used with CICS. The default is `00` if omitted.

Step 2: Define Transactions to CICS

Four transactions are required: `TDSP`, `SMGO`, `SMEX`, `SMNE`. These transactions must be defined as follows:

a `SMGO` to run `PACNKERN`. The name `SMGO` is mandatory as it is used internally by SMARTS.

b `SMEX` to run `PACNKERX`. `SMEX` is a suggested transaction name.

c `TDSP` to run `PACNSTRT`. The name `TDSP` is mandatory as it is used internally by SMARTS.

d `SMNE` to run `PACNNEP`. The name `SMNE` is mandatory, as it is used internally by SMARTS.

A sample job is provided to add these transactions to the CSD.

Step 3: Define Programs to CICS

- Many programs are used by SMARTS so autoinstall should be activated. Autoinstall can be activated by setting the `SIT` parameter `PGAIPGM=ACTIVE`

  The following Assembler language programs must be defined to CICS.

  - `PACNKERN`, define with `EXECKEY(CICS)`
  - `PACNKERX`, define with `EXECKEY(CICS)`
  - `PACNSTRT`, define with `EXECKEY(CICS)`
  - `PACNNEP`, define with `EXECKEY(CICS)`
  - `RAANPARM`, define with `"RELOAD=YES"`

  A sample job is provided to add these programs to the CSD.

Step 4: Define Programs to the Program List Tables

a To automatically initialise SMARTS POSIX at CICS startup, in the `DFHPLTxx` for the `PLTPI`, insert `PACNSMGO` as a second phase PLT program.

b To automatically terminate SMARTS POSIX at CICS shutdown, in the `DFHPLTxx` for the `PLTSD`, insert `PACNKERX` as a first phase PLT program.

Step 5: Provide a DFHZNEP node error program

- If the installation already has a `DFHZNEP` node error program in use, modify it to invoke the `SMNE` transaction under the conditions detailed in the model assembler program `PACNZNEP`, supplied in the APSvrs sublibrary. If the installation does not have a `DFHZNEP` node error program in use, use the supplied model program `PACNZNEP` to create one.

Step 6: Define transaction security

- Each of the four SMARTS transactions may be defined with only basic security to the security manager installed. `SMGO` and `SMEX` are the only transactions that may be entered at a terminal and may be protected as required.

Where Next ?

You have now installed the SMARTS software. You can continue now with the installation of the application that is to run on SMARTS.

Note that the configuration procedure of the application that runs of SMARTS may instruct you to modify some of SMARTS's configuration parameters.

4 Installation on VM/CMS

This document describes procedures for installing SMARTS under VM/CMS.

Software AG recommends that you keep unmodified copies of all materials distributed or created as part of the installation process. This may assist with problem diagnosis later.

- The Installation Tape
- Installing the SMARTS Server Environment
- Where Next ?

The Installation Tape

The installation tape is described under the following headings:

- Tape Contents
- Copying Contents of the Tape to Disk

Tape Contents

The installation tape contains the following files:

Dataset	Contains . . .	
APSvrs.LIBR	Library	SAGLIB
	Sublibrary	APSvrs SMARTS components: phases, objects, JCL, sample source members, and macros.

Sample JCL and Source Members

The following table lists the sample sources and execs loaded to the SMARTS mini disk during installation. These must be modified before being used:

Member	Contains ..
PXANCONF RTS A	The POSIX server configuration parameters.
PXANHOST RTS A	Sample parameter file to customize the TCP/IP host name and host address table.
SMARTS CONFIG A	Sample server environment parameters.
RJSNLNKD EXEC A	Sample exec to link a DLL or Shared Library for SMARTS.
RJSNLNKM EXEC A	Sample exec to link a main for SMARTS
RJSNRUN EXEC A	Sample exec to run a command line utility
RJSNSSE EXEC A	Sample exec to run the SMARTS Server environment

Copying Contents of the Tape to Disk

Step 1: Restore the SMARTS mini disks

- Use the following commands to restore the SMARTS library:

```
INPUT FROM DICK REQUIRED HERE
```

Installing the SMARTS Server Environment

Step 1: Edit the SMARTS Server start-up EXEC

- Modify the sample SMARTS server start-up exec member RJBSSSE EXEC on the APSvrs disk to suit your installation naming conventions.

  The example SMARTS start-up EXEC below is typical for a VM/CMS environment and serves as the basis for the various descriptions and explanations that follow:

```
/* */
parse arg
'sp prt *'
'nucxload pcayrini'
'nucxload playbsd '
'nucxload playbfp '
'nucxload playcfn '
'nucxload pleybsd '
'nucxload pleybfp '
'nucxload pleycfn '
'nucxload pcaytrdb'
'nucxload pcaytrdh'
'nucxload fee     '
'nucxload fea     '
'nucxload softle  '
'nucxload softla  '
tlinvm
'nucxdrop pcayrini'
'nucxdrop playbsd '
'nucxdrop playbfp '
'nucxdrop playcfn '
'nucxdrop pleybsd '
'nucxdrop pleybfp '
'nucxdrop pleycfn '
'nucxdrop pcaytrdb'
'nucxdrop pcaytrdh'
'nucxdrop fee     '
```

```
'nucxdrop fea      '
'nucxdrop softle   '
'nucxdrop softla   '
'sp prt close'
exit
```

For a description of the SMARTS SYSPARMS, see *Configuration Parameters*. For parameters relevant to your application, refer to the configuration documentation for the software that runs on SMARTS.

Where Next ?

You have now installed the SMARTS software. You can continue now with the installation of the application that is to run on SMARTS.

Note that the configuration procedure of the application that runs on SMARTS may instruct you to modify some of SMARTS's configuration parameters.

5 Initialization and Termination

This document explains the initialization and termination procedures.

SMARTS Server Environment Initialization

The SMARTS server environment can be started by invoking the SMARTS procedure in one of two ways:

1. By an operator START command (OS/390), or by a POWER R command (VSE);
2. By a batch job stream.
3. By a CMS REXX EXEC.

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

OS/390

The SMARTS procedure can be invoked from the operator's console using the OS/390 START command with the standard command format

```
S SMARTS, . . .
```

- where "SMARTS" is the name of a procedure that resides on an OS/390 procedure library.

Alternatively, the SMARTS procedure can be invoked by an OS/390 batch job

```
//SMARTS JOB .............
//IEFPROC EXEC SMARTS,...
```

VSE/ESA

The SMARTS procedure can be invoked from the operator's console using the VSE POWER R command with the standard command format

```
R RDR,SMARTS
```

- where "SMARTS" is the name of the job residing in the VSE POWER Reader Queue.

CM/CMS

The SMARTS server environment may be started from the CMS machine where it is installed by running the SMARTS REXX EXEC

SMARTS Server Environment Termination

This section provides information for terminating the SMARTS server environment under OS/390 and VSE/ESA. For SMARTS server environment termination in other environments, see the documentation for the particular Software AG application product that uses SMARTS.

OS/390

The SMARTS server environment may be terminated with the command EOJ:

```
F SMARTS,EOJ
```

- or with the OS/390 STOP (P) command:

```
P SMARTS
```

The operating system CANCEL command can also be used to terminate the SMARTS server environment, but is not recommended because it does not cause a logical shutdown.

VSE/ESA

The SMARTS server environment may be terminated with the command EOJ:

```
nn EOJ
```

- where nn is the partition reply ID.

This command immediately terminates outstanding terminal I/O requests and performs a logical shutdown of the SMARTS server system.

The operating system POWER FLUSH command can also be used to terminate the SMARTS server environment, but Software AG does not recommend it because it does not cause a logical shutdown.

VM/CMS

The SMARTS server environment may be terminated by entering the command EOJ at the console of the VM/CMS machine where SMARTS is running:

```
EOJ
```

6

Operator Command Processing

By definition, the only SMARTS environment capable of receiving and acting on operator commands is the SMARTS server environment, the commands and format for which are documented in this chapter.

SMARTS also provides applications with the ability to accept operator commands when the console communication driver interface (CDI) module is specified to initiate operator communication.

- **Communication Driver Interface (CDI) Commands**
- **SMARTS Server Environment Operator Commands**

Communication Driver Interface (CDI) Commands

When the CDI is active and the application running on SMARTS has opened the console, commands may be issued as follows in the various environments.

The following sections provide information about issuing commands under OS/390. For information about other platforms, see the documentation for the particular Software AG application product that uses SMARTS.

Commands Issued to an OS/390 Batch Job

```
F jobname,command
```

- where

jobname	is the name of the OS/390 job
command	is the command string to be passed to the application

Commands Issued to a SMARTS Server Environment OS/390 Job

```
F smarts,SERV,server-name,command
```

- where

smarts	is the name of the SMARTS server environment OS/390 job
server-name	is the name of the application specified on the SERVER configuration statement
command	is the command string to be passed to the application

SMARTS Server Environment Operator Commands

Once the SMARTS server environment has been initialized, the computer operator can control the various SMARTS server environment facilities and ascertain the status of the SMARTS server environment system by entering one or more of the SMARTS server environment operator commands at the computer operator console.

Communicating with the SMARTS POSIX Server

Apart from the SMARTS commands to initialize and terminate the POSIX server, the following operator commands may be issued to the POSIX server by issuing the SMARTS operator command

```
Posix,command
```

- where

Posix	is the name of the POSIX server at startup.
command	is one of the commands QUIESCE or FORCE.

QUIESCE

Use this command to terminate the POSIX server. Existing users are allowed to run until termination; however, any new request to use the services of the POSIX server is rejected.

Any POSIX server QUIESCE commands after the first issue a message indicating how many users are still using the system. When no users are using the POSIX server system, the command terminates the POSIX server.

FORCE

Use this command to forcibly terminate the POSIX server and bypass all integrity checks during termination processing. Because command results are unpredictable, Software AG recommends that the entire SMARTS address space be cycled whenever FORCE is used to terminate the server.

Platform Requirements for Entering Operator Commands

The following sections provide information about entering operator commands under OS/390. For information about other platforms, see the documentation for the particular Software AG application product that uses SMARTS.

Note: The MODIFY (F) command and proper ID (OS/390), or the REPLID (VSE) are assumed, and are not shown in command syntax in this chapter.

OS/390

For OS/390 systems, operator commands are entered via the OS/390 MODIFY (F) and STOP (P) commands. These commands are directed toward the job name.

The general format for entering the OS/390 MODIFY command is:

```
F id,command,argument(s)
```

- where "id" is the job or started task name.

VSE/ESA

For VSE/ESA systems, every SMARTS server environment has an outstanding reply on the console with the following message:

```
RTSOPC0085-* SMARTS READY FOR COMMUNICATIONS
```

The general format for entering a VSE SMARTS server environment operator command is:

```
nn command,argument(s)
```

- where "nn" is the VSE/ESA outstanding reply number assigned by the system. The message RTSOPC0085-* is outstanding until the EOJ operator command is entered, at which time operator communications are halted.

In the case of a SMARTS abnormal termination, the operator must respond to the outstanding reply with an "EOB" (end-of-block) operator command.

VM/CMS

For VM/CMS systems, the VM/CMS machine where SMARTS is started becomes the SMARTS console so operator commands must simply be entered at the VM/CMS console prompt and these will be processed by SMARTS.

The general format for entering a VM/CMS SMARTS server environment operator command is:

```
command,argument(s)
```

Command Format Requirements

With the exception of the EOJ command, each command may be entered in full or may be abbreviated. The minimum abbreviation required is the number of characters necessary to uniquely identify the command. The characters needed to identify the command are underlined in the description of each command in the sections that follow.

Command Overview

All SMARTS operator commands are described in the next section. The following table summarizes these commands:

Command	Purpose
DUMP	VSE/ESA: switches destination device for SMARTS server environment dumps from SYSLST to COMDMP and vice versa.
EOJ	Causes a logical shutdown of the SMARTS server environment.
PLIST	Displays a list of the current tasks defined in the requested task group and the status of each.
SERV	Enables a server to be stopped or started.
STATS	Writes the current SMARTS server environment statistics to the sysout dataset.
TLIST	Displays a list of the current threads defined in the requested thread group and the status of each.

Command Descriptions

DUMP

The DUMP command is only supported under VSE/ESA.

The computer operator uses the DUMP command to switch the device to which SMARTS ABEND dumps are written from SYSLST to COMDMP and vice versa.

Format	Description
DUMP	write snap dump to currently selected device
DUMP,DISK	write dump to VSAM dataset COMDMP
DUMP,NODISK	write dump to SYSLST

EOJ

The EOJ command causes a logical shutdown of the SMARTS server system.

Note: If the system programmer specified an EOJ verification password using the EOJ,VER system parameter, that password must also be entered with the EOJ command.

Format	Example
EOJ	EOJ
EOJ,VER=password	EOJ,VER=STOPCOMP

Note that a logical shutdown of SMARTS can also be performed with the OS/390 STOP (P) command.

If SMARTS does not come down after you have entered EOJ, try

```
EOJ,FORCE
```

PLIST

This command provides a list of the current tasks defined in the requested task group and the status of each. If no task group is supplied as a parameter to this command, all tasks of all task groups are displayed. The following is a sample output resulting from this command.

```
RTSOPC0099-T COMMAND RECEIVED AT 9:33:15 FROM CONSOLE - 00 WAS PLIST
RTSOPC0067-T -> GrpName Status Use Wait LastOp Time Program  Tid.. L
RTSOPC0067-T -> OC       A-Run
RTSOPC0067-T -> TAM      A-Wait                     USTACK       4 0 STC06160
RTSOPC0067-T -> MSGPO    A-Wait
RTSOPC0067-T -> PAGING   A-Wait
RTSOPC0067-T -> FIO      A-Wait
RTSOPC0067-T -> DEFAULT  A-Wait   0    2 Wrtm       USTACK       4 0
RTSOPC0067-T -> DEFAULT  A-Wait   0    2 Coexit     UPDS         4 4
RTSOPC0067-T -> DEFAULT  A-Wait   0    2
RTSOPC0001-T PList command COMPLETED.
```

GrpName

This is the name of the task group of which the task in question is a member. In the case of system tasks, this is the name of the system task.

Status

This reflects the current status of the task. The status is a combination of two state indicators separated by a dash ('-'). The primary state indicator is the letter preceding the dash indicates whether the task is Active, Quiescing or Dormant by the letters A, Q and D respectively. Active in this sense indicates that the task is available to do work. When it is quiescing, it will remain active long enough to finish any work which has been started by the task while dormant tasks cannot be used and will have no secondary state associated with them. The secondary states that may occur are as follows:

Status	The task is . . .
Wait	waiting. In this state, the task is waiting on new work or on events requested by programs running in threads associated with it.
HrdW	in a 'hard wait' status caused by the program currently running on it. The task is not available to service other programs that might be waiting for it.
Run	currently running a user program.
Disp	going through its dispatching cycle either finishing off old work or looking for new work.

Use

This is the current use count for the task. The use count includes the current user of the task, any users for whom a wait was issued on the task and any users with an affinity for this task.

Wait

This is the current wait count for the task. This reflects the number events upon which the task is waiting and includes two standard events those being that work has been queued to the task group work queues or to the task's own work queue.

LastOp

This is the last SMARTS server environment operator command that was issued under control of the task.

Time

When the task has a secondary status of 'Run', this will reflect the time in seconds that this user has spent under control of the task.

Program

This is the name of the program currently active under control of the task, or the last program to be active under control of the task if it has a secondary status of 'wait'. If the task has never been used, this will be blank, however, once it has been used, this will always contain a value.

Tid..

This is the tid of the current TIB active under control of the task, or the last TIB to be active under control of the task if it has a secondary status of 'wait'. If the task has never been used, this will be blank, however, once it has been used, this will always contain a value.

L

This is the level number on which one of the following users is running:

- the user currently active under control of the task; or
- the last user that was active under control of the task if it has a secondary status of 'wait'.

If the task has never been used, this is blank; once it has been used, this always contains a value.

SERV

The SERV command enables the computer operator to pass commands to a server. Servers can be started and terminated using this command, and requests can be sent to servers. These servers must be specified in the SMARTS server environment sysparm SERVER. See the chapter *Configuring SMARTS Environments*.

Examples

```
SERV TERM,server-id
```

- to terminate the specified server.

```
SERV INIT,server-id,server-parameters
```

- to initialize the specified server. The string "*server-id,server-parameters*" is exactly the same as it would be specified in a SYSPARM SERVER=(server-id,server-parameters).

```
SERV server-id,server-command
```

- to send a command to the specified server. As long as the server-id does not conflict with any valid operator command name, the term "SERV" can be omitted in this notation.

STATS

The STATS command writes the SMARTS server environment EOJ statistics to the dataset specified by the SYSPRINT DDNAME in the SMARTS server environment start-up procedure.

Format	Example
STATS	STAT

TLIST

The TLIST command lists the current threads defined in the requested thread group and the status of each. If no thread group name is provided as a parameter to the request, all threads of all thread groups are displayed.

Subgrp

The name of the thread subgroup to which the thread in question belongs.

Status

The current status of the thread. The status is a combination of two state indicators separated by a dash ('-'). The primary state indicator is the letter preceding the dash and indicates whether the thread is active, quiescing, or dormant by the letters A, Q, and D, respectively. "Active" indicates that the thread is available to do work. When it is quiescing, it remains active long enough to finish any work that was started in the thread, while a dormant thread cannot be used and has no secondary state associated with it. The secondary states that may occur are as follows:

Status	Description
Free	Indicates that the thread is free to run other work. If there was a previous user of the thread, this state indicates that this user's program ended or was rolled out.
Occ	The 'occupied' status indicates that the thread is available to do work; however, the user program currently occupying the thread must be rolled out before any new work can be started in the thread.
Disp	Indicates that the thread is reserved and the dispatcher is currently in the process of either starting a new user program or rolling in a user program that was previously rolled out.
Run	Indicates that the user program in the thread is currently running.
Susp	Indicates that the user program has been temporarily suspended as a wait was issued either directly by the user program or indirectly by a function used by the program. In this state, the user program may not be rolled out. Internally, it indicates that the operating system task associated with the work is active elsewhere. Once the condition for the wait is satisfied, the task continues processing this work.

Use

The current use count for the thread. The use count includes the current user of the thread plus any other non-relocatible users previously rolled out from this thread.

Wait

The current wait count for the thread. This reflects the number of users waiting to run in the thread at the present time.

LastOp

The last SMARTS server environment operator command that was issued in the thread.

Time

When the thread has a secondary status of 'Susp' or 'Run', this reflects the time in seconds that this user spent in the thread.

Program

The name of the program currently active in the thread, or the last program to be active in the thread if the thread has a status of 'free' or 'occ'. If the thread has never been used, this is blank; however, once the thread has been used, this always contains a value.

Tid..

The TID of the current TIB active in the thread, or the last TIB to be active in the thread if the thread has a status of 'free' or 'occ'. If the thread has never been used, this is blank; however, once the thread has been used, this always contains a value.

Active L

The level number on which the one of the following users is running:

- the user currently active in the thread; or
- the last user to be active in the thread if the thread has a status of 'free' or 'occ'.

If the thread has never been used, this is blank; however, once the thread has been used, this always contains a value.

7

Support and Maintenance for SMARTS

The SMARTS system nucleus is written mostly in IBM 390 Assembler but also in open source C and is therefore supported using cumulative fix packs as a means to provide corrections to customers.

- Reporting Problems
- Problem Resolution

Reporting Problems

Problems should be reported to your local technical support center. You will be asked to provide whatever information is required to solve the problem. Generally, you should have the following available when reporting a problem:

1. Version, revision, and SM level of the SMARTS software where the problem occurred.
2. Type and level of operating system where SMARTS was running.
3. Version, revision, and SM level of other products associated with the problem (for example, Natural, Adabas).
4. Message numbers where applicable.
5. System log for a period of time before the event.
6. Sequence of actions used to cause the problem, if reproducible.
7. Name and offset of the module where the problem occurred. Where an ABEND occurs within a SMARTS module, generally RC will point to the start of the module where you will find a constant identifying the module. The PSW address should be subtracted from the address in RC to provide the offset into the module.
8. The register contents at the time of the ABEND.

With this information, it may be possible to identify a previous occurrence of the problem and a correction. If this is not the case, the following additional information is required:

1. The operating system online dump or SMARTS address space dump, as appropriate.
2. Output from the job where the failure occurred.
3. Other information that support personnel feel is relevant.

Problem Resolution

A number of tools are available to diagnose problems as follows.

Batch Dumps

When running in batch, a standard dump is taken for the POSIX server address space, as would be taken for a normal batch task. Standard diagnosis techniques may be applied to this dump.

Trace Facilities

Where problems are encountered with the operation of the POSIX server interface, the trace functions may be useful in determining the nature of the problem. POSIX server tracing may be activated using the POSIX server TRACE configuration parameter.

8 Configuration Tables

A number of translation or reference tables are supplied with the POSIX server in source format. Software AG recommends that you use these tables without modification. If errors are found in these tables, Software AG will make the correction generally available.

PAANAETT
The table PAANAETT is used to translate ASCII data to EBCDIC data.

PAANEATT
The table PAANEATT is used to translate EBCDIC data to ASCII data.

PAANEULE
The table PAANEULE is used to translate uppercase EBCDIC characters to lowercase. It is used primarily by the 'tolower' and '_tolower' functions.

PAANEUTT
The table PAANEUTT is used to translate EBCDIC lowercase characters to EBCDIC uppercase characters. It is used primarily by the 'toupper' and '_toupper' functions.

PAANSPCE
The table PAANSPCE is used to identify 'white space characters' in an EBCDIC data stream. It is used primarily for the 'isspace' function.