



natural

natural

Version 4.1.2 for
Mainframes

SMARTS Installation and
Operation

 **SOFTWARE AG**

This document applies to Natural Version 4.1.2 for Mainframes and to all subsequent releases.

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

© Copyright Software AG 2003
All rights reserved.

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

About this Documentation	1
About this Documentation	1
SMARTS Environments, Features and Requirements	2
SMARTS Environments, Features and Requirements	2
The SMARTS Runtime Environment	2
The SMARTS Server Environment	2
The SMARTS Client Environment	3
Supported TCP/IP Stacks	3
Installation of SMARTS Client on CICS for EntireX	4
Installation of SMARTS Client on CICS for EntireX	4
The Installation Tape	4
Tape Contents	4
Copying Contents of the Tape to Disk	5
Installing the SMARTS CICS Client Environment	5
Installation Verification	9
1) Initialization Messages of SMARTS	9
2) Installation Verification	10
3) CICS Shutdown verification	10
Where Next ?	10
Installation on VSE/ESA	11
Installation on VSE/ESA	11
The Installation Tape	11
Tape Contents	11
Copying Contents of the Tape to Disk	12
Installing the SMARTS Server Environment	12
Installing the VSE CICS Client Environment	15
Where Next ?	16
Initialization and Termination	17
Initialization and Termination	17
SMARTS Server Environment Initialization	17
OS/390	17
VSE/ESA	17
SMARTS Server Environment Termination	17
OS/390	18
VSE/ESA	18
Operator Command Processing	19
Operator Command Processing	19
Communication Driver Interface (CDI) Commands	19
Commands Issued to an OS/390 Batch Job	19
Commands Issued to a SMARTS Server Environment OS/390 Job	19
SMARTS Server Environment Operator Commands	19
Communicating with the SMARTS POSIX Server	20
Platform Requirements for Entering Operator Commands	20
Command Format Requirements	21
Command Overview	21
Command Descriptions	21
Support and Maintenance for SMARTS	27
Support and Maintenance for SMARTS	27

Reporting Problems	27
Problem Resolution	28
Batch Dumps	28
Trace Facilities	28
Configuration Tables	29
Configuration Tables	29

About this Documentation

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
● 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

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.

This chapter covers the following topics:

- The SMARTS Runtime Environment
-

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/GCS, 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/GCS	Batch (command line applications only)
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 'pthreads' 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.
- MSP/EX from Fujitsu using the TISP TCP/IP stack.
- VM/ESA using the standard IBM stack available on VM.

Installation of SMARTS Client on CICS for EntireX

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
- Installing the SMARTS CICS Client Environment
- Installation Verification
- Where Next ?

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

The Installation Tape

This section describes the contents of the installation tape for usage in connection with EntireX:

- Tape Contents
- Copying Contents of the Tape to Disk

Tape Contents

Datasets Created during the Installation Process

Dataset	Contains ..
APSVrs.USERSRCE	Source members needed during installation

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 ..
EXXCONF	Sample member with environment parameters.
EXXSYSYP	Sample member with SMARTS configuration parms.
EXXDIR	Sample configuration member for Location Transparency.
PXANHOST	Sample member with HOSTS definitions for TCP/IP.
PXANSRVT	Sample member with SERVICES definitions for TCP/IP.
DFHCSDUP	Job to define programs and transactions to CICS
DFHINPUT	Input file for DFHCSDUP job.
PACNZNEP	Sample assembler source of DFHZNEP node error program.

Copying Contents of the Tape to Disk

If not yet done, copying the tape contents to disk as described under *Copying Tape Contents to Disk* in the EntireX documentation. A description of the tape contents can be found under *Contents of Mainframe Tape*.

Installing the SMARTS CICS Client Environment

Important:

This section describes how to install the CICS Client Environment for SMARTS specifically for usage in connection with EntireX.

Step 1: Allocate and Initialize User PDS Dataset

- To create and initialize the datasets required for SMARTS, modify the sample job in member EXXINS2 on the APS271.S001 dataset to suit your installation's environment, and run it to create the appropriate dataset.

This job copies all needed modifiable members from the APS271.S001 dataset to the newly created APSvrs.USERSRCE dataset in order to retain all APS271.S001 members as delivered.

Step 2: Modify the CICS Procedure

1. Add the following datasets to the DFHRPL concatenation:

```
DFHRPL DD DISP=SHR,DSN=...
        DD DISP=SHR,DSN=APSVRS.LDnn
        DD DISP=SHR,DSN=APSVRS.LD00
        DD DISP=SHR,DSN=XTEVRS.LDnn
        DD DISP=SHR,DSN=BTEVRS.LSnn
        DD DISP=SHR,DSN=BTEVRS.LDnn
        DD DISP=SHR,DSN=EXXVRS.LOAD
        DD DISP=SHR,DSN=...
```

2. Add the following DD statements to your CICS procedure:

```

/** -----
/** Definition of environment variables and SMARTS configuration
/** -----
//SYSPARM DD DISP=SHR,DSN=APSVrs.USERSRCE(EXXSYSP)
//CONFIG DD DISP=SHR,DSN=APSVrs.USERSRCE(EXXCONF)
/**
/** -----
/** Optional: Only used with Location Transparency
/** Datasets and code conversion tables for Location Transparency
/** -----
//XDSINI DD DISP=SHR,DSN=APSVrs.USERSRCE(EXXDIR)
//ECSOBJ DD DISP=SHR,DSN=BTEvrs.ECnn
/**
/** -----
/** TCP data member
/** (contact your TCP/IP administrator)
/** -----
//SYSTCPD DD DISP=SHR,DSN=SYS1.PARMLIB(TCPDATA)
/**
/** -----
/** Optional: HOSTS and services definitions
/** -----
//HOSTS DD DISP=SHR.DSN=APSVrs.USERSRCE(PXANHOST)
//SERVICES DD DISP=SHR.DSN=APSVrs.USERSRCE(PXANSRVT)
/**
/** -----
/** Optional: Only used with SSL
/** Certificate file for SSL calls
/** -----
//CACERT DD DISP=SHR,DSN=EXXvrs.CERT(CACERT)
//RANDFILE DD DISP=SHR,DSN=EXXvrs.SRCE(RND)
/**
/** -----
/** Trace and Output datasets
/** -----
//APSLOG DD SYSOUT=X
//XTSLOG DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//STDOUT DD SYSOUT=X
//STDERR DD SYSOUT=X
//XDSOUT DD SYSOUT=X

```

Step 3: Modify the EXXSYSP member (SMARTS configuration)

- This dataset is required to define the runtime characteristics of your SMARTS environment. Refer to member called EXXSYSP in APSVRS.USERSRCE which has been created in step 1. This file contains the following:

EXXSYSP:

```

* -----
* Support for IBM OS/390 File Subsystem.
* -----
CDI_DRIVER=('FILE,PAAMFSIO')
*
* -----
* TCP/IP configuration for SMARTS (OE TCP/IP STACK).
* -----
CDI_DRIVER=('TCPIP,PAAOSOCK')
*
* -----

```

```

* DDNAME assignments for ENV dataset HOSTS and SERVICES file
* -----
ENVIRONMENT_VARIABLES=DD:CONFIG
HOSTS_FILE=DD:HOSTS
SERVICES_FILE=DD:SERVICES
*

```

Notes:

1. SMARTS only supports IBM OpenEdition TCP/IP stack for z/OS and OS/390.
2. Every TCP/IP communication (OE stack) needs an USS segment (formerly called OE segment), therefore you need to define this for your CICS addressspace.

Contact your TCP/IP administrator for more information or see section *SMARTS POSIX Layer Configuration* and *Sysparm Format*.

Step 4: Modify the EXXCONF member (Environment variables)

- Here you can set environment variables for applications that are using SMARTS. This dataset is specified by the ENVIRONMENT_VARIABLES parameter in SYSPARM, which defaults to "CONFIG". Refer to member called EXXCONF in APSvrs.USERSRCE which has been created in step 1:

EXXCONF:

```

* -----
* BROKER VARIABLES
* -----
ETB_TRANSPORT=TCP-SSL-NET
ETB_STUBLOG=3
*
* -----
* SET DDNAMES FOR SSL AND LOCAL TRANSPARENCY (Optional)
* -----
RANDFILE=DD:RANDFILE
ECSOBJDIR=DD:ECSOBJ/
*
* -----
* SET DDNAME AND TRACE LEVEL FOR XTS
* -----
XTSLOG=DD:XTSLOG
* XTSTRACE=65534

```

For more information, see the section *Environment variables in Entire X* of the EntireX documentation.

Step 5: Modify the PXANHOST and PXANSRVT member (optional)**1. Customize the SMARTS TCP/IP host file (PXANHOST)**

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.USERSRCE 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 address space.

Sample:

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
157.189.160.95 ETB001
```

2. Customize the SMARTS TCP/IP service file (PXANSRVT)

This table is used to set the relationship between services names and protocol numbers for any TCP/IP stack implementation that does not provide this facility.

Sample:

Format: <service_name> <protocol num>/<protocol name> (#COMMENT)

```
ETB001      3800/tcp      # Broker ETB001
```

Note:

The need of the services file depends on the application using the Broker stub (i.e. Natural RPC may require it).

Step 6: Provide a DFHZNEP node error program (optional)

- 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 .USERSRCE dataset.

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

The node error program gets control on the following CICS events:

```
DFH2410 - NODE UNRECOVERABLE VTAM LOSTERM ERROR CODE XX
DFH3462 - SESSION TERMINATED
```

If the node error program gets control, it will clean up the SMARTS resources within CICS.

Note:

For additional information on CICS node error programs, please refer to the appropriate IBM CICS documentation.

Step 7: Customize your CICS**1. Define Programs and Transactions to CICS using DFHCSDUP**

Please use sample job DFHCSDUP in APSVRS .USERSRCE to apply all necessary definitions to CICS . The member DFHINPUT is the input for job DFHCSDUP and contains all necessary definitions to CICS.

Important:

Autoinstall must be activated since there are many other programs will be loaded dynamically by SMARTS. AUTOINSTALL feature in CICS can be activated by setting the SIT parm :

```
PGAIPGM=ACTIVE
```

2. Define Programs to the Program List Tables

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.

Sample PLT entry:

```
DFHPLT TYPE=ENTRY , PROGRAM=PACNSMGO
```

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.

Sample PLT entry:

```
DFHPLT TYPE=ENTRY , PROGRAM=PACNKERX
```



Warning:

If CICS will be shut down with the `Immediate` command (i.e. "CEMT P,SHUT,I"), control is not being given to the DFHPLTSD at shutdown, and SMARTS cannot terminate normally. Software AG strongly recommends not to use "CEMT P SHUT I".

Installation Verification

1) Initialization Messages of SMARTS

A successful 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 communication 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 client under CICS which can be used by EntireX. You can continue now with the installation of the applications that has to run on SMARTS. This can be any application using the delivered SMARTS stub EXAAPSC and running under CICS, for instance EntireX RPC Servers, Natural RPC client/servers or other applications (e.g. COBOL) that use the SMARTS stub to access EntireX Broker.

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

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.

This document covers the following topics:

- 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 Sublibrary	SAGLIB 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 FFS,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 RJBINS2.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 RJBINS4.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 RJBINS3.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

1. 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
```

2. Verify and if necessary add or modify the following parameter in the members RJBINPROC.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 RJBNSPROC.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:

```
* $$ JOB JNM=RJBNSPROC,CLASS=c,DISP=d,LDEST=(,uid)
* $$ LST CLASS=c,DISP=d
// JOB RJBNSPROC --- SMARTS Startup ---
/*
/* ===== *
/* SMARTS Startup *
/* ===== *
/*
// OPTION PARTDUMP,NOSYSDMP,LOG
/*
/* Dump file for APS -----
/*
// DLBL COMDMP,'aps.vsam.dumpfile',,VSAM,CAT=cccccc 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) (RJANPARAM.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

1. Add the APSVRS sublibrary to the LIBDEF search chain. This must be placed before any other Software AG product sublibrary.
2. Add DLBL and EXTENT statements for STDOUT and STDERR if using.
3. Add DLBL and EXTENT statements for APSTRCE.
4. 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)',.....
```

5. 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:

1. SMGO to run PACNKERN. The name SMGO is mandatory as it is used internally by SMARTS.
2. SMEX to run PACNKERX. SMEX is a suggested transaction name.
3. TDSP to run PACNSTRT. The name TDSP is mandatory as it is used internally by SMARTS.
4. 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 PGAIIPGM=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

1. To automatically initialise SMARTS POSIX at CICS startup, in the DFHPLTxx for the PLTPI, insert PACNSMGO as a second phase PLT program.
2. 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.

Initialization and Termination

This chapter covers the following topics:

- SMARTS Server Environment Initialization
 - SMARTS Server Environment Termination
-

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.

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.

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.

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.

This chapter covers the following topics:

- 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

<i>jobname</i>	is the name of the OS/390 job
<i>command</i>	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

<i>smarts</i>	is the name of the SMARTS server environment OS/390 job
<i>server-name</i>	is the name of the application specified on the SERVER configuration statement
<i>command</i>	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

<i>Posix</i>	is the name of the POSIX server at startup.
<i>command</i>	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.

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-relocatable 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.

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 patch levels as a means to provide corrections to customers.

This chapter covers the following topics:

- 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.

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.