

SAS Technical Support Document TS-678

Installing the SAS 9.0 Object Spawner for SAS 8.2 Servers on OS/390 and z/OS Platforms

Lindy Mayfield
October 2003
SAS/Europe Customer Support
Lindy.mayfield@eur.sas.com

Table of Contents

Installing and Running the SAS 9.0 Object Spawner	3
Introduction.....	3
Installing the SAS/ IT 9.0 Spawner for use with SAS 8.2 IOM.....	4
Overview.....	4
Installing Version 9 Object Spawner Load Modules from Hotfix	5
Installing New SAS 8.2 SASWQS Module from Hotfix.....	5
Security Considerations	5
Setting up the Object Spawner Started Task.....	6
Setting up the Object Spawner to run under USS.....	7
Creating and configuring the object spawner configuration file.....	9
Creating and configuring the object spawner shell script	10
Creating the SAS IOM Server CLIST	12
Starting the Object Spawner	13
Testing the operator connection to the Object Spawner	14
Testing the connection via the EG Administrator	14

Installing and Running the SAS 9.0 Object Spawner

Introduction

In order to facilitate the move away from APPC and towards the technology used by SAS 9, a hotfix has been created for SAS 8.2 to allow the SAS 8.2 IOM Server to be started by the SAS 9.0 Object Spawner.

In order to run the SAS 9.0 Object Spawner, a full SAS 9.0 installation is not necessary. A hotfix has been created that contains a small subset of the SAS 9.0 software which contains only the modules required to run the object spawner.

A bit of a technical description of how things work may be in order. Here is a brief comparison between the old and the new setup. The old way worked very well: The object spawner is started and listens to a port. The client connects to that port and supplies credentials. Once validated, the spawner uses APPC services to launch SAS in an APPC address space. The spawner gives the client socket to the IOM server and the IOM server takes it. The client and server are now connected with the server running in an APPC address space.

The new way works like this: The object spawner is started and listens to a port, just like always. The client connects to that port and is validated. Next, the spawner spawns the command supplied in the configuration file, which is a shell script that will launch SAS with the proper parameters. Since the object spawner uses the spawn service, it can easily pass the socket from the client to the server.

Installing the SAS/ IT 9.0 Spawner for use with SAS 8.2 IOM

Overview

The Version 9 Object Spawner can launch either an SAS 8.2 or SAS 9 IOM server. Implementing the Version 9 Object Spawner with the SAS 8.2 IOM Server will consist of the following actions, and each will be covered in detail.

- Install Version 9 Object Spawner Load Modules from Hotfix
- Install New SAS 8.2 SASWQS Module from Hotfix
- Security considerations
- Setting up the Object Spawner Started Task
- Setting up the Object Spawner to run under USS
- Creating and configuring the object spawner configuration file
- Creating and configuring the object spawner shell script
- Creating the SAS IOM Server CLIST
- Starting the Object Spawner
- Testing the operator connection to the object spawner
- Testing the connection via the EG Administrator

Two things need to be done before starting the configuration of the object spawner:

1. The customer must obtain the new V9 Object Spawner modules from hotfix 82IH13. These modules must be installed into a separate load library and must be kept separate from the original SAS 8.2 install.
2. A new SAS 8.2 SASWQS module is needed for the SAS 8.2 IOM Server. This new module exists in the latest base SAS 8.2 hotfix bundle or can be obtained from hotfix 82BB10. This is the only change to the 8.2 installation.

For links to the hotfixes, see SAS Note SN-008715

<http://support.sas.com/techsup/unotes/SN/008/008715.html>

Installing Version 9 Object Spawner Load Modules from Hotfix

Obtain hotfix 82IH13 which contains a small subset of SAS 9 load modules needed for executing the SAS 9.0 Object Spawner. The 82IH13 hotfix load library will only be used with the object spawner and not with the SAS 8.2 installation. **The 82IH13 hotfix library must be kept separate from the SAS 8.2 library.** Also provided in the 82IH13 hotfix CNTL data set are several examples to aid in this setup.

Note that an object spawner is a stand-alone SAS program that takes care of starting SAS IOM servers. The object spawner can be a different release than the IOM server. The same object spawner can also launch IOM servers of different SAS levels (currently SAS 8.2, SAS 9.0, and SAS 9.1).

Installing New SAS 8.2 SASWQS Module from Hotfix

Obtain the new SASWQS module from hotfix 82BB10 or the latest base SAS 8.2 hotfix bundle. This new module will work with both APPC and UNIX System Services. The SASWQS module belongs with the SAS 8.2 installation and should reside in either this hotfix load library or the SAS 8.2 prefix.LIBRARY. **The SASWQS module should be in a separate load library than the 82IH13 hotfix which contains supporting SAS 9.0 modules.** Take care not to mix these libraries. The new SASWQS module must be in the search path of the CLIST that will start up the SAS IOM Server.

Security Considerations

IBM provides two levels of daemon security. The first is normal UNIX security whereby a daemon that switches the UID must run as superuser with a UID of 0, or the daemon must make a successful `__passwd()` call before a `spawn()` that will switch to that UID. In the case of the object spawner running under normal UNIX security, it is not necessary to run it with a superuser UID.

If RACF Program control is enabled and the RACF facility class BPX.DAEMON is defined, then any program that makes a `__passwd()` call must run from a RACF program controlled library.

For more information, please refer to the UNIX System Services Planning guide, especially the section entitled "Setting up for Daemons."

To begin the setup configuration, it is assumed that you have done the following:

- You have a load library with the new SAS 9.0 spawner modules in it from hotfix 82IH13. For purposes of this install document, I will call this library OBJSPAWN.V9.LOAD.
- You have modified your 8.2 installation to use the new SASWQS module from hotfix 82BB10 (or latest hotfix bundle).

Setting up the Object Spawner Started Task

There are three possible ways to run the Object Spawner:

1. The Object Spawner can be run in an MVS address space as a STC, exactly as the 8.2 Object Spawner. This is the preferred method.
2. The SAS 9 Object Spawner can also run from a USS shell. There may be an advantage to this method in that a SAS administrator with no special MVS STC privileges can start, stop, and maintain the object spawner. This is a good method for testing or for preparing to run the object spawner from BPXBATCH.
3. A variation of the USS shell method above is to run using BPXBATCH. This allows the object spawner to be started as an STC and at the same time run in a BPX address space. This method is used to overcome the PARM string's 200 character limitation when running directly as a STC.

Normally, you will choose to run the object spawner directly as a STC. This method is fine for using clients such as Enterprise Guide, Appdev Studio, or the Java interfaces. The Object Spawner reads its configuration file from a local text file in LDIF format.

If you wish for the spawner to connect to an LDAP server in order to read the configuration information, then the problem of the object spawner parm string being longer than will fit into the MVS JCL Parm string of 100 characters must be dealt with. In 9.0 the only workaround for this is to run the object spawner under the shell. At this point the object spawner can be started from the shell, for example running it in the background, or it can be run as a started task using BPXBATCH. The BPXBATCH method would be more suitable in a production environment.

Therefore the recommendation would be to run the object spawner directly as an STC without LDAP, and as a STC using BPXBATCH if LDAP is used. Running the object spawner from the USS shell may be helpful for testing, for example in preparing for BPXBATCH.

There are two restrictions when running in an MVS address space. The first is that SYSPRINT must be routed to an HFS file in order to ensure that any CLIST or other errors

from the launch of the SAS IOM server are captured. The second restriction is that if the PARM string is longer than 100 characters, then the object spawner must be run in a USS shell. For this reason, if you are using LDAP then you cannot fit the parameters in the JCL PARM string, so you will have to run from the shell.

If you are using an LDIF file and running the Object Spawner as a STC, a sample STC is provided in the 82IH13 hotfix CNTL data set called OBJSTC. If you are connecting to an LDAP server for the configuration information, please read the next section "Setting up the Object Spawner to run under USS" otherwise skip to the section "Creating and Configuring the Object Spawner Configuration File".

Setting up the Object Spawner to run under USS

This step is optional and only necessary if you are using LDAP or you wish to run from the shell for testing purposes.

First decide where you want to install the SAS Object Spawner files on USS. There is no need to install these in a public directory unless it is the intent that they be started by normal users. For purposes of this installation, I have chosen the arbitrary directory `/objspawn9` and the name `startobj.sh` for the shell script that will invoke the object spawner.

Create an external link to MVS.

```
$ ln -e OBJSPAWN objspawn
```

Next you need to let the system know the location of the MVS object spawner load modules. For purposes of demonstration, I have chosen the name `OBJSPAWN.V9.LOAD`. The load library is stored in the UNIX environment variable `STEPLIB`. The external link will search first the dataset named in `STEPLIB` and then go through the "normal" MVS search order, LPA, etc.

```
$ export STEPLIB='OBJSPAWN.V9.LOAD'
```

If your SAS/C transient library is not in the linklist, then you must also create an environment variable for this.

```
$ export ddn_CTRANS=SAS.SASC.TRANSLIB
```

Do not attempt to concatenate the SAS/C transient library in the `STEPLIB`. This will cause the object spawner to fail.

Next put all of the following into a shell script, in our case called `startobj.sh`.

```
#!/bin/sh
export STEPLIB=OBJSPAWN.V9.LOAD
exec /objspawn9/objspawn \
    -cf /objspawn9/objspawn.conf \
    -sv -slf /objspawn9/objspawn.log \
    > /objspawn9/bpxbat.log
```

Verify that this shell script works properly. At this point you can either run the object spawner in the background or as a system STC using BPXBATCH.

If this is to be run as an STC, then create the JCL to run BPXBATCH which will run the object spawner. The below JCL is provided in the 82IH13 hotfix CNTL dataset member USSSTC.

```
//OBJSPAWN EXEC PGM=BPXBATCH,REGION=20M,
//  PARM='SH '
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//STDIN DD PATH='/objspawn9/startobj.sh'
//STDOUT DD PATH='/tmp/objspawn9/bpxbat.stdout',
//      PATHMODE=(SIRUSR,SIWUSR,
//      SIRGRP,SIWGRP,SIROTH,SIWOTH),
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC)
//STDERR DD PATH='/tmp/objspawn9/bpxbat.stderr',
//      PATHMODE=(SIRUSR,SIWUSR,
//      SIRGRP,SIWGRP,SIROTH,SIWOTH),
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC)
```

Notes:

- The STEPLIB environment variable in the shells script points to the Object Spawner V9 modules.
- The environment variable ddn CTRANS points to the SAS/C transient library.
- When you use this method to start the spawner, there is no longer the 100 character limitation in the PARM statement.

Creating and configuring the object spawner configuration file

Creating the object spawner configuration file is similar to the V8 version of this file. For purposes of demonstration, I have chosen to call this file `objspawn.conf`.

```
dn: sasSpawnercn=MySpawner
objectClass: sasSpawner
sasSpawnercn: MySpawner
sasMachineDNSName: localhost
sasOperatorPort: 9902
sasOperatorPassword: sas
sasVerbose: true
sasLogfile: /objspawn9/objspawn.log

# This Server runs 8.2
dn: sasServercn=MyServer82
objectClass: sasServer
sasPort: 9903
sasProtocol: bridge
sasCommand: /objspawn9/sas82.sh nosasuser --
sasMachineDNSName: localhost
sasServercn: MyServer82
```

The PDS member OBJCFG in the 82IH13 hotfix CNTL data set is an example of this configuration file. In the sample configuration file you will need to change the ports, the location of the log file, and the location of the shell script (`sasCommand`).

Notes:

- The first block of text defines the object spawner.
- The `sasOperatorPort` defines the port that you can telnet into to display a status of the spawner and shut it down.
- The password will be used when you telnet to the spawner. Make it simple for testing, and then change it.
- The remaining blocks of text, (here only one), define the servers that will be started when connecting to that particular port.
- The `sasCommand` points to the shell script to launch SAS. This will be discussed in the next section.
- Note the double dashes "--" after the `sasCommand`. The following shell script uses this marker to separate the static from the dynamic parms passed to the IOM server.
- Make sure the permission bits are set correctly for this file to be read. As an example, `chmod 644 objspawn.conf`.

Creating and configuring the object spawner shell script

A shell script is used to launch SAS in a UNIX System Services (BPX) address space. This shell script need only be modified to point to the CLIST to launch SAS as an IOM server. The IOM Server CLIST will be discussed next section. In the 82IH13 hotfix CNTL data set a sample shell script is provided called SAS82SH.

```
#!/bin/sh
#
#-----
# The following should be modified or verified
#-----
#
# The following command should point to your SAS CLIST
#
cmd="/bin/tso -t EX '&sasprefix.CLIST(SAS82IOM)'"
#
#
# The shell log contains the ouput from the /bin/tso command.
#
shlog="/tmp/inttech/shell.log"
#
# Use the account data to place SAS in the correct service class
#
# export _BPX_ACCT_DATA=A123
#
# Number of seconds before JWT timeout due to inactivity
#
# export TMOU=30
#
# In case the SAS CLIST needs to call execs
#
# export SYSPROC="PROD.MVS.CLIST:USER.MVS.CLIST"
#
#-----
# No user modifications beyond this point
#-----
#
#
echo `date` "Starting an IOM Server. \n" >> $shlog
#
# foundDashDash is a boolean. When TRUE, we found the string
# "--" in our arguments.
#
foundDashDash=0
#
# Construct our arguments
#
args=''
for arg in "$@" ; do
  if [ "$arg" != "--" ]; then
    tmp="$arg ";
  else
    tmp="SRVOPTS('";
    foundDashDash=1;
  fi
  args="$args$tmp"
done
#
# If we found a "--", we need to close the SRVOPTS option
#
```

```

if [[ $foundDashDash -ne 0 ]]; then
  args="$args '"
fi
#
# Construct our command line...
#
cmd="$cmd '$args'"
#
echo `date` "Starting SAS with command:" $cmd "\n" >> $shlog
#
# Crank up SAS
#
exec $cmd >> $shlog

```

Notes:

- Make sure that the permission bits are set correctly for this file. As an example, `chmod 755 sas82.sh`.
- Take special note of the `_BPX_ACCT_DATA` environment variable. This may prove extremely valuable in tuning using WLM.
- Note that the shell script's output is directed to a file. This is valuable in debugging, and is necessary if this is run from MVS and BPXBATCH.
- The line that reads `cmd="/bin/tso -t EX '&sasprefix.CLIST(SAS82IOM) '"` will need to point to the CLIST defined in the next section.

Creating the SAS IOM Server CLIST

A sample IOM Server CLIST is provided in the 82IH13 hotfix CNTL data set member called SAS82IOM. This CLIST differs from the CLIST provided with base SAS or other SAS products. The IOM Server CLIST contains the NOSASUSER logic in it along with new CLIST parameters.

Here are a few differences in the CLIST highlighted :

```
STAI      NOSTAI      /* Trap subtask abends?      */ +
STAX      NOSTAX      /* Trap attentions?          */ +
STACK     NOSTACK     /* Create new i/p stack?     */ +
SHARE     NOSHARE     /* Share subpool 78          */ +
SASCLIENTPORT() /* Client socket from spawner*/ +
SASDAEMONPORT() /* Port to connect to spawner*/ +
SRVOPTS() /* for SAS IOM server      */ +
NETENCRA LG() /* Encryption algorithm      */ +
INHERIT(0) /* Get socket from spawner   */ +
NOSASUSER /* Suppress SASUSER allocate */ +
TCPDEBUG(0) /* TCP/IP Debugging          */ +
TRACE     /* CLIST debugging           */ +
```

The bottom lines of the CLIST should include the following logic:

```
/*
/* put system sort link library in STEPLIB if needed
/*
IF &STR(&SORTLINK) NE THEN +
  SET SORTLDSN=&STR()
/*
/*
/* For IOM Server
/*
IF &STR(&SRVOPTS) NE THEN +
  SET OPTIONS=&STR(&OPTIONS &SRVOPTS)
/*
/* Invoke SAS
/*
/*
SET TASKLIB = &STR(&LOAD &SASLOAD &SORTLDSN)
SET PARM=&STR(T(&TASKLIB) E(&ENTRY) O('&OPTIONS') &FLUSH &NOFLUSH +
  &STAE &NOSTAE &STAI &NOSTAI &STAX &NOSTAX +
  &STACK &NOSTACK &SHARE &NOSHARE)
SASCP &PARM
SET RC=&LASTCC
CONTROL NOMSG
IF &AUTOEXEC NE THEN +
  FREE F(&DDAUTOEX)
EXIT CODE(&RC)
```

Starting the Object Spawner

Starting the object spawner assumes that you have done the following:

- You have installed the 2 necessary hotfixes
- You have made the appropriate security changes necessary.
- You have setup a method to start the Object Spawner (STC or shell)
- You have created an object spawner configuration file.
- You have created and customized the `sas82.sh` shell script to launch the SAS 8.2 IOM Server.
- You have verified, perhaps using PROC TCPTTEST, that your TCP/IP environment is configured properly.

Start the object spawner STC normally, or from the shell by launching the shell script `startobj.sh`.

If the object spawner fails to start, check all log files for errors or other diagnostic information.

Testing the operator connection to the Object Spawner

This is just a quick reality check to make sure that the object spawner is working ok. Telnet into the operator port of the object spawner; then type in the operator password. Enter the `list` command to see the available servers defined. Enter the `bye` command to shut down the spawner.

Please note that the spawner does not echo characters typed back to the telnet session, and by default echo is not turned on with the telnet client that Microsoft provides.

Testing the connection via the EG Administrator

The very first tests should be with the EG Administrator. Begin by creating a new server and supplying the hostname and port of the IOM 8.2 server.

When debugging, refer to all logs including the MVS syslog.

