

# **System V Init Quick Installation and Operation Guide 1.0**

Document Number VM01-0001-01

January 18, 2006

Sine Nomine Associates

43596 Blacksmith Square  
Ashburn, VA 20147

**Release 1.0**

**First edition, first release, May 2005**

This edition applies to the 1.0 release of SysVInit.

© Copyright Sine Nomine Associates 2005

# Contents

---

|   |    |
|---|----|
| <b>Basic Concepts</b> .....             | 1  |
| <b>SysVInit Overview</b> .....          | 2  |
| Communication with SysVInit .....       | 2  |
| The RUNLEVEL concept .....              | 2  |
| What is a service? .....                | 3  |
| <hr/>                                   |    |
| <b>Installation of SysVInit</b> .....   | 5  |
| <b>Requirements</b> .....               | 6  |
| <b>Installation Procedure</b> .....     | 7  |
| <hr/>                                   |    |
| <b>SysVInit Operation Guide</b> .....   | 13 |
| <b>SysVInit Variables</b> .....         | 14 |
| Variable Names .....                    | 14 |
| Variable Values .....                   | 14 |
| Predefined Variables .....              | 14 |
| Predefined Global Variables .....       | 14 |
| RUNLEVEL .....                          | 14 |
| User-modifiable global variables. ....  | 14 |
| Internal .....                          | 15 |
| Service Variables .....                 | 15 |
| STATUS .....                            | 15 |
| Generic Action Variables .....          | 16 |
| SYSPERF-specific Action Variables ..... | 16 |
| LINUX-specific Action Variables .....   | 16 |
| <b>SysVInit commands</b> .....          | 18 |
| RUNLEVEL .....                          | 18 |
| Syntax .....                            | 18 |
| Use .....                               | 18 |
| LIST RUNLEVELS .....                    | 18 |
| Syntax .....                            | 18 |
| Use .....                               | 18 |
| SHOW RUNLEVEL .....                     | 18 |
| Syntax .....                            | 18 |
| Use .....                               | 18 |
| SET GLOBALVAR .....                     | 19 |
| Syntax .....                            | 19 |
| Use .....                               | 19 |
| GET GLOBALVAR .....                     | 19 |
| Syntax .....                            | 19 |
| Use .....                               | 19 |
| LIST GLOBALVARS .....                   | 19 |
| Syntax .....                            | 19 |

|                       |           |
|-----------------------|-----------|
| Use                   | 19        |
| REFRESH               | 19        |
| Syntax                | 19        |
| Use                   | 20        |
| SERVICE ADD           | 20        |
| Syntax                | 20        |
| Use                   | 20        |
| SERVICE REMOVE        | 20        |
| Syntax                | 20        |
| Use                   | 20        |
| SERVICE               | 20        |
| Syntax                | 20        |
| Use                   | 20        |
| SET SERVICEVAR        | 21        |
| Syntax                | 21        |
| GET SERVICEVAR        | 21        |
| Syntax                | 21        |
| Use                   | 21        |
| LIST SERVICEVARS      | 21        |
| Syntax                | 21        |
| Use                   | 21        |
| LOG                   | 21        |
| Syntax                | 21        |
| Use                   | 21        |
| LEVELMAINT            | 22        |
| Syntax                | 22        |
| Use                   | 22        |
| CMD                   | 22        |
| Syntax                | 22        |
| Use                   | 23        |
| STOP                  | 23        |
| Syntax                | 23        |
| Use                   | 23        |
| SHUTDOWN              | 23        |
| Syntax                | 23        |
| Use                   | 23        |
| SHUTDOWN REIPL        | 24        |
| Syntax                | 24        |
| Use                   | 24        |
| <b>Example of Use</b> | <b>25</b> |

---

|                     |           |
|---------------------|-----------|
| <b>Future Plans</b> | <b>27</b> |
|---------------------|-----------|

---

|                          |           |
|--------------------------|-----------|
| <b>Technical Support</b> | <b>29</b> |
|--------------------------|-----------|

---

|                   |           |
|-------------------|-----------|
| <b>Appendices</b> | <b>31</b> |
|-------------------|-----------|

|   |           |
|---|-----------|
| <b>Appendix A. Files used by SysVinit</b> | <b>32</b> |
| SYSVINIT SRTABLE                          | 32        |
| DEFAULT NAMES                             | 32        |

|                           |    |
|---------------------------|----|
| DISPATCH EXEC . . . . .   | 32 |
| PROFILE S5IEXEC . . . . . | 32 |
| RUNLEVEL EXEC . . . . .   | 32 |
| SVM EXEC . . . . .        | 32 |
| Syntax . . . . .          | 32 |
| Use . . . . .             | 33 |
| Use . . . . .             | 34 |
| START . . . . .           | 34 |
| STOP . . . . .            | 34 |
| RESTART . . . . .         | 34 |
| STATUS . . . . .          | 34 |
| PROBE . . . . .           | 34 |
| CONDRESTART . . . . .     | 34 |
| RELOAD . . . . .          | 35 |
| FORCERELOAD . . . . .     | 35 |
| DUMMY EXEC . . . . .      | 35 |
| SYSPERF EXEC . . . . .    | 35 |
| LINUX EXEC . . . . .      | 35 |
| SNAUME TEXT . . . . .     | 35 |
| * S5IHELP . . . . .       | 35 |
| * HELPMSG . . . . .       | 35 |

# Figures

|     |  |    |
|-----|--|----|
| 1.  | Command asking SysVInit to list global variables         | 2  |
| 2.  | Example directory modifications for AUTOLOG1             | 7  |
| 3.  | AUTOLOG1 Directory entry                                 | 7  |
| 4.  | Logging off AUTOLOG1                                     | 8  |
| 5.  | Formatting and accessing AUTOLOG1 192                    | 8  |
| 6.  | VMARC command to unpack distribution files               | 8  |
| 7.  | Access AUTOLOG1 191                                      | 8  |
| 8.  | Saving a copy of PROFILE EXEC                            | 8  |
| 9.  | Replacing the old PROFILE EXEC with the SysVInit version | 9  |
| 10. | Copying SYSVINIT RTABLE into place                       | 9  |
| 11. | Detach AUTOLOG1 191 and 192                              | 9  |
| 12. | Logging on AUTOLOG1                                      | 9  |
| 13. | SysVInit startup message                                 | 9  |
| 14. | Copying SVM EXEC for each service                        | 10 |
| 15. | Using LEVELMAINT to configure the default runlevel       | 10 |
| 16. | Using PROBE to set the status of running services        | 10 |
| 17. | Using variable lists and tuples                          | 14 |
| 18. | Example of Linux-specific action variables               | 17 |
| 19. | Sending a SHOW RUNLEVEL command to SysVInit              | 18 |
| 20. | Example of use   | 25 |

---

# Basic Concepts

---

# SysVInit Overview

SysVInit is a product that brings Unix-like startup processing to z/VM and VM/ESA. It has many features of System-V-style init processing; for instance, it uses named runlevels and provides ordered startup and shutdown of services determined on a per-runlevel basis. However, in many ways it is considerably more flexible and powerful: runlevel names need not be numeric, but can be any legal 8-character identifier, and, unlike classical init, SysVInit for z/VM and VM/ESA does dependency tracking within a runlevel, and will refuse to start a service (and provide notification) if that service's prerequisites cannot be met.

---

## Communication with SysVInit

SysVInit uses the Programmable Operator (PROP) facility to accept messages sent by administrative users and to manipulate runlevel definitions and the state of various service virtual machines.

An administrative user communicates with SysVInit using the TELL command. In the default configuration, MAINT and OPERATOR are defined as administrative users. How to add and remove users is explained later in this document. The following is an example of what should be typed by the MAINT user to list the global variables in use by SysVInit:

---

```
TELL AUTOLOG1 LIST GLOBALVARS
```

---

Figure 1. Command asking SysVInit to list global variables

---

## The RUNLEVEL concept

A *runlevel* is simply a collection of services specified in a NAMES file. It can be any legal filename, whether that be a Unix-style numeric runlevel, or DEFAULT, or any other identifier.

Each *runlevel* NAMES file contains a list of services that start in that runlevel and any start or stop dependencies they might have.

The runlevel NAMES files should not be edited directly. Instead, use the LEVELMAINT command to edit them.

Any cosmetic changes you have made to a runlevel NAMES file, such as comments, case modification, or indentation, will be lost the first time LEVELMAINT is used to manipulate that runlevel.

---

## What is a service?

In this document, "service" and "service virtual machine" are often used interchangeably. This is a historical development of the way VM and CMS have traditionally worked, in that it is usual for each service virtual machine to provide exactly one service. However, this is not necessarily the case: it is sometimes reasonable for a single machine to provide multiple services. Conversely, some services (such as SYSPERF) do not correspond to any virtual machines at all.

A service is just some piece of system functionality made available to the user. Usually a single machine provides a single service, and that is the assumption made in the SVM EXEC script shipped as a template for particular service-controlling scripts. If you have services that are either not bound to any particular virtual machine or machines that provide multiple services, the service control script will require modification. The DUMMY EXEC script is a good template for services that do not map to a virtual machine, and LINUX EXEC provides a template for Linux-based services.



---

## Installation of SysVInit

This is a quick installation guide for SysVInit. It provides the necessary steps to install and start SysVInit such that it delivers functionality equivalent to your current AUTOLOG1 PROFILE EXEC.

---

# Requirements

You will require the following:

- VM/ESA 2.4, or any release of z/VM. Any earlier release of VM/ESA should work if the CMS Utilities feature is installed (to provide WAKEUP processing). However, such configurations are unsupported. VM/ESA 2.4 is the oldest tested release.
- 5 cylinders of 3390 DASD.
- Some method of transferring files available via http to your VM system.
- VMARC (the VMARC package is available from the IBM download library, <http://www.vm.ibm.com/download>)

## Installation Procedure

1. Log in to CMS as a user with sufficient privileges to modify directory entries and link other users' disks.
2. Download the S5I VMARC file. It can be found at <http://www.sinenomine.net/vm/s5i>. You will need a userid at Sine Nomine in order to download files. If you do not have one, instructions for creating a userid are available on the web site.
3. After downloading, reblock the file to F 80 if necessary. For purposes of this document, we will assume that the file resides on the minidisk or SFS directory accessed at D, and that minidisk or directory has sufficient space both to hold the file and its unpacked contents. Five cylinders (4M) is sufficient.
4. Using your local directory maintenance procedures, create a 5-cylinder 192-disk for AUTOLOG1.

---

```
* 0192 created for SysVInit
MDISK 0192 3390 0001 0005 VOL001
```

---

Figure 2. Example directory modifications for AUTOLOG1

If you have no AUTOLOG1 user

Very old releases of VM/ESA and P/390 preconfigured systems do not have an AUTOLOG1 user. If you intend to use SysVInit on such a system, you must first create an AUTOLOG1 user. If you wish to proceed, create an AUTOLOG1 user with a directory entry similar to the following:

---

```
USER AUTOLOG1 AUTOLOG1 32M 32M ABCDEG
ACCOUNT 9 SYSTEM
IPL 190
MACH XA
XAUTOLOG OP1 MAINT
CONSOLE 0009 3215
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
MDISK 0191 3390 0001 0001 VOL000
* 0192 created for SysVInit
MDISK 0192 3390 0001 0005 VOL001
```

---

Figure 3. AUTOLOG1 Directory entry

To migrate P/390 services it will be necessary to replicate the existing startup sequence from AUTOLG EXEC on OPERATOR's 191-disk. Also comment out the line which executes AUTOLG1 EXEC in OPERATOR's PROFILE EXEC.

End of If you have no AUTOLOG1 user

---

Do not use the A-Disk or SFS

---

Although it is not necessary to install SysVInit in AUTOLOG1's 192, SysVInit currently must reside on a minidisk. It also must not reside on the A-disk of the user as which it runs: both PROP and GLOBALV management, which SysVInit use, require a writeable A-disk for their own use. Since SysVInit tries to keep only a write link to its files whenever possible, the product will not run correctly if SysVInit's files are resident on the SysVInit user's A-disk.

---

End of Do not use the A-Disk or SFS

---

5. Install the product. The product installation is a manual and fairly labor-intensive process at this point. We hope to substantially automate it in future releases. At present, it consists of the following steps.
  - a. Log off the AUTOLOG1 user

---

```
FORCE AUTOLOG1
```

---

Figure 4. Logging off AUTOLOG1

- b. Format and access the AUTOLOG1 192 disk.

---

```
LINK AUTOLOG1 192 1192 W
FORMAT 1192 E
```

---

Figure 5. Formatting and accessing AUTOLOG1 192

- c. Ensure that VMARC MODULE is on an accessed disk.
    - d. Unpack the file:

---

```
VMARC UNPACK S5I VMARC D = = E
```

---

Figure 6. VMARC command to unpack distribution files

- e. Access AUTOLOG1 191

---

```
LINK AUTOLOG1 191 1191 W
ACCESS 1191 F
```

---

Figure 7. Access AUTOLOG1 191

- f. Save a copy of the original PROFILE EXEC.

---

```
COPYFILE PROFILE EXEC F = ORIG =
```

---

Figure 8. Saving a copy of PROFILE EXEC

- g. Copy PROFILE S5IEXEC to AUTOLOG1's PROFILE EXEC

---

```
COPYFILE PROFILE S5IEXEC E = EXEC F (REPLACE OLDDATE
```

---

Figure 9. Replacing the old PROFILE EXEC with the SysVInit version

- h. (Optional) Copy the help files to MAINT 19D and rebuild the HELP saved segment. Detailed information for this step can be found in the *CP Planning and Administration* manual.
- i. Copy SYSVINIT SRTABLE to SYSVINIT RTABLE

---

```
COPYFILE SYSVINIT SRTABLE E = RTABLE =
```

---

Figure 10. Copying SYSVINIT RTABLE into place

- 6. Modify SYSVINIT RTABLE to suit your installation. Edit the SYSVINIT RTABLE file and replace HOSTNODE with your nodename, and add any other additional privileged users you wish to configure.

SYSVINIT RTABLE is column-sensitive

When replacing HOSTNODE, if your node name is shorter than eight characters, be sure to pad it with spaces until it is exactly eight characters. Otherwise the column alignments in the RTABLE file will be damaged (DMSPOS, DMSPOR, and DISPATCH must begin in column 56) and SysVInit will not run correctly.

End of SYSVINIT RTABLE is column-sensitive

- 7. Detach AUTOLOG1's disks.

---

```
DETACH 1191
DETACH 1192
```

---

Figure 11. Detach AUTOLOG1 191 and 192

- 8. Start the SysVInit service by logging on the AUTOLOG1 user.

---

```
XAUTOLOG AUTOLOG1
```

---

Figure 12. Logging on AUTOLOG1

- 9. Wait for SysVInit to start. Wait for 30 seconds. If you have the OPERATOR console active, you will see there a response similar to:

---

```
MSG FROM AUTOLOG1: PROP running with routing table SYSVINIT RTABLE Z2
```

---

Figure 13. SysVInit startup message

- 10. Create the runlevel scripts for each service you want to start.

From an administrative user (MAINT or OPERATOR initially), tell AUTOLOG1 to run the SERVICE ADD to create the init script for each service. Refer to the printed-out original PROFILE EXEC for the list of services.

For the list shown above, the following commands would be needed:

---

```
TELL AUTOLOG1 SERVICE VMSERVS  ADD
TELL AUTOLOG1 SERVICE VMSERVU  ADD
TELL AUTOLOG1 SERVICE VMSERVR  ADD
TELL AUTOLOG1 SERVICE TCPIP    ADD
TELL AUTOLOG1 SERVICE DIRMAINT  ADD
TELL AUTOLOG1 SERVICE DATAMOVE  ADD
TELL AUTOLOG1 SERVICE GCS      ADD
```

---

Figure 14. Copying SVM EXEC for each service

11. Add the correct virtual machines to the default runlevel. From the administrative user tell AUTOLOG1 to run the LEVELMAINT command to add each user to the default runlevel. After this task is complete, remove the dummy service. Refer to the printed-out original PROFILE EXEC for the list of services.

For the list shown above, the following commands would be needed:

---

```
TELL AUTOLOG1 LEVELMAINT DEFAULT VMSERVS
TELL AUTOLOG1 LEVELMAINT DEFAULT VMSERVU
TELL AUTOLOG1 LEVELMAINT DEFAULT VMSERVR
TELL AUTOLOG1 LEVELMAINT DEFAULT TCPIP
TELL AUTOLOG1 LEVELMAINT DEFAULT DIRMAINT
TELL AUTOLOG1 LEVELMAINT DEFAULT DATAMOVE
TELL AUTOLOG1 LEVELMAINT DEFAULT GCS
```

---

Figure 15. Using LEVELMAINT to configure the default runlevel

At this point, the default runlevel behaves identically to the old AUTOLOG1 PROFILE EXEC: all the services will be started, with no regard to dependencies. Dependency verification is described below.

12. Update the service status variables Now we wish to tell SysVinit to update its status for each of the machines in its runlevel. As they were all started with the old AUTOLOG1 PROFILE EXEC, they should all be running, and thus when we issue PROBE commands, their status should be set to AVAIL. The following sequence of commands will execute the PROBE and set the appropriate status variables:

---

```
TELL AUTOLOG1 SERVICE VMSERVS  PROBE
TELL AUTOLOG1 SERVICE VMSERVU  PROBE
TELL AUTOLOG1 SERVICE VMSERVR  PROBE
TELL AUTOLOG1 SERVICE TCPIP    PROBE
TELL AUTOLOG1 SERVICE DIRMAINT  PROBE
TELL AUTOLOG1 SERVICE DATAMOVE  PROBE
TELL AUTOLOG1 SERVICE GCS      PROBE
```

---

Figure 16. Using PROBE to set the status of running services

Now you have completed the migration to SysVinit.

\_\_\_\_\_ If you encounter difficulties \_\_\_\_\_

If you have difficulty with this procedure, please send mail to **beta-s5i@sinomine.net**.

\_\_\_\_\_ End of If you encounter difficulties \_\_\_\_\_



---

# SysVInit Operation Guide

This section is intended to document the functions available when the SysVInit user is running.

---

# SysVInit Variables

---

## Variable Names

Variable names may be any legal Rexx variable name. However, it is unwise to use the "at sign" ("@"), as variables containing this character are used internally to represent list variables.

---

## Variable Values

All variables in SysVInit can contain as their values all characters excepting the colon and the percent sign. Specifically, blanks are legal characters within variable values.

Whenever it is necessary to specify a list of values for a service or global variable, separate the different values with colons. When a variable value is actually a tuple, separate the parts of the value with percent signs.

For instance, the following command would set the list of allowed admins to MAINT and OPERATOR at the local node, along with MAINT at NODE2.

---

```
TELL AUTOLOG1 SET GLOBALVAR ADMIN MAINT:OPERATOR:MAINT%NODE2
```

---

Figure 17. Using variable lists and tuples

Internally this will set ADMIN to "MAINT", ADMIN@1 to "OPERATOR", and ADMIN@2 to "MAINT%NODE2" in AUTOLOG1's LASTING GLOBALV file.

---

## Predefined Variables

### Predefined Global Variables

SysVInit defines the following global variables. Additional global variables may be defined by the user and referenced in modified startup scripts, but it is generally preferred to use service-specific variables, perhaps in template scripts if they are widely applicable, rather than global variables.

#### RUNLEVEL

The RUNLEVEL global variable tracks the runlevel at which the system is currently operating. It should never be set directly, but only via the RUNLEVEL command.

#### User-modifiable global variables.

The following global variables are intended for modification by the user.

**ADMIN** ADMIN contains the list of allowed administrator IDs for SysVInit. For administrative userids at remote nodes, use the syntax *userid%nodeid*: The default value of ADMIN is OPERATOR:MAINT.

**STARTTIMEOUT** STARTTIMEOUT specifies the amount of time SysVInit should wait for a service to start. The global STARTTIMEOUT value can be overridden by service-specific values. The value is an integer representing the number of seconds to wait.: The default value of STARTTIMEOUT is 30.

**STOPTIMEOUT** STOPTIMEOUT specifies the amount of time SysVInit should wait for a service to shut down. The global STOPTIMEOUT value can be overridden by service-specific values. The value is an integer representing the number of seconds to wait.: The default value of STOPTIMEOUT is 30.

**DEFAULTRUNLEVEL** DEFAULTRUNLEVEL specifies the runlevel SysVInit should enter at startup.: The default value is DEFAULT.

## Internal

The following global variables are for SysVInit's internal use, and should not generally be modified by the user.

**DISKOWNER** DISKOWNER contains the userid of the virtual machine that owns the SysVInit code. This should generally be the same as the virtual machine that runs SysVInit.: The default value of DISKOWNER is AUTOLOG1.

**DISKADDR** DISKADDR specifies the virtual address of the disk on which the SysVInit code resides.: The default value of DISKADDR is 192.

**DISKLABEL** DISKLABEL specifies the volume label of the disk on which the SysVInit files are installed.: The default value of DISKLABEL is AUT192.

**LOGRETENTION** LOGRETENTION is an integer that specifies for how many days PROP log files are kept on the 191-disk.: The default value of LOGRETENTION is 45.

## Service Variables

All service variables are user-modifiable. The following variables are predefined, but users are encouraged to create their own variables referenced in modified startup scripts to create their own effects.

When a service variable is used to trigger an action within a startup script, this variable is referred to as an *action variable*

\_\_\_\_\_ Action Variables do not take effect immediately \_\_\_\_\_

Setting an action variable does not, by itself, execute the action associated with the variable: a restart (or reload, if implemented) is necessary to actually enact the changes.

\_\_\_\_\_ End of Action Variables do not take effect immediately \_\_\_\_\_

## STATUS

Each service keeps track of its current status in the service variable STATUS. This variable should never be set directly, but only by startup scripts.

## Generic Action Variables

All standard templates (LINUX, DUMMY, and SVM) support the following variables.

**STARTTIMEOUT** STARTTIMEOUT specifies the amount of time SysVInit should wait for a service to start. When specified as a service variable, this overrides the global STARTTIMEOUT value. The value is an integer representing the number of seconds to wait.

**STOPTIMEOUT** STOPTIMEOUT specifies the amount of time SysVInit should wait for a service to shut down. When specified as a service variable, this overrides the global STOPTIMEOUT value. The value is an integer representing the number of seconds to wait.

## SYSPERF-specific Action Variables

SYSPERF is a predefined dummy service in the DEFAULT runlevel. It is intended to set system-wide performance characteristics. It contains the following action variables.

**DSPBUF** DSPBUF is an interface to the CP SET SRM DSPBUF command. Its argument is simply the argument for CP SET SRM DSPBUF. For further information, see the *CP Command and Utility Reference*.

**LDUBUF** LDUBUF is an interface to the CP SET SRM LDUBUF command. Its argument is simply the argument for CP SET SRM LDUBUF. For further information, see the *CP Command and Utility Reference*.

**STORBUF** STORBUF is an interface to the CP SET SRM STORBUF command. Its argument is simply the argument for CP SET SRM STORBUF. For further information, see the *CP Command and Utility Reference*.

## LINUX-specific Action Variables

LINUX EXEC is a template for the definition of Linux-based service machines. It contains action variables allowing the guest to couple to restricted Guest LAN or VSWITCH segments.

VSWITCH and GLAN limitations

Please note that the VSWITCH and GLAN action variables only expose a subset of the functionality available in the corresponding CP SET commands. If you require advanced features such as trunking on your VSWITCH, you will need to modify the appropriate guest's service script.

End of VSWITCH and GLAN limitations

**GLAN** The value of GLAN is a colon-delimited list of Guest LANs the guest is allowed to couple to. The percent sign may be used within a LAN definition to specify the LAN owner: the LAN name is specified as *ownerid%lanname*, and SYSTEM is the implicit LAN ownerid if none is specified.

**VSWITCH** The value of VSWITCH is a colon-delimited list of VSWITCHes the guest is allowed to couple to. The percent sign may be used within a VSWITCH definition to append a VLAN identifier to the VSWITCH name. For VSWITCHes, the VLAN number can be appended to the VSWITCH name, as *switchname%vlan*.

**Example of Use** The following example would allow SVM1 to couple to the VSWITCHes SW1 and SW2, and restrict SW2 to VLAN 0003. It would then allow SVM1 to couple to SYSTEM-owned Guest LAN GLAN1, and to GLAN2 owned by userid MIKE. The restart ensures that the changes are made.

---

```
TELL SYSVINIT SET SERVICEVAR SVM1 VSWITCH SW1:SW2%0003
TELL SYSVINIT SET SERVICEVAR SVM1 GLAN GLAN1:MIKE%GLAN2
TELL SYSVINIT SERVICE SVM1 RESTART
```

---

Figure 18. Example of Linux-specific action variables

---

## SysVInit commands

All of these commands are sent to the AUTOLOG1 machine by privileged users or service machines. Thus, all of them are preceded by an implicit TELL AUTOLOG1 command. For example, the following would be used to send a SHOW RUNLEVEL command to SysVInit running in AUTOLOG1:

---

```
TELL AUTOLOG1 SHOW RUNLEVEL
```

---

Figure 19. Sending a SHOW RUNLEVEL command to SysVInit

---

## RUNLEVEL

### Syntax

```
▶▶—RUNLEVEL—runlevel—————▶▶
```

### Use

RUNLEVEL instructs SysVInit to switch to the runlevel defined in the *runlevel* NAMES file.

---

## LIST RUNLEVELS

### Syntax

```
▶▶—LIST RUNLEVELS—————▶▶
```

### Use

LIST RUNLEVELS searches the configuration disk (AUTOLOG1 192 by default) for NAMES files. Each NAMES file found defines a runlevel to the SysVInit service.

---

## SHOW RUNLEVEL

### Syntax

```
▶▶—SHOW RUNLEVEL——runlevel——————▶▶
```

### Use

If *runlevel* is not specified, SHOW RUNLEVEL uses the current runlevel as its argument. The services invoked at this runlevel are displayed with their start and stop dependencies.

---

## SET GLOBALVAR

### Syntax

```
▶▶ SET GLOBALVAR -name- -value- ▶▶
```

### Use

SET GLOBALVAR sets the global variable *name* to *value*. This change is persistent across sessions.

---

## GET GLOBALVAR

### Syntax

```
▶▶ GET GLOBALVAR -name- ▶▶
```

### Use

GET GLOBALVAR prints the value of the global variable *name*

---

## LIST GLOBALVARS

### Syntax

```
▶▶ LIST GLOBALVARS ▶▶
```

### Use

LIST GLOBALVARS prints all of SysVInit's global variables not tied to a particular service. Note that this will display the internal representation of list variables. Thus, if you see:

```
SNADIS0227I Service GLOBAL variable ADMIN = MAINT.  
SNADIS0227I Service GLOBAL variable ADMIN@1 = OPERATOR.  
SNADIS0227I Service GLOBAL variable ADMIN@2 = MAINT%NODE2.
```

This means that ADMIN was previously set to the list MAINT:OPERATOR:MAINT%NODE2.

---

## REFRESH

### Syntax

```
▶▶ REFRESH ▶▶
```



by the virtual machine whose userid is *service*. This allows service machines to report and manipulate their own state within SysVInit.

---

## SET SERVICEVAR

### Syntax

```
▶▶ SET SERVICEVAR -service- -name- -value- ▶▶
```

This command sets the value of the variable *name* to *value* for the service *service*.

---

## GET SERVICEVAR

### Syntax

```
▶▶ GET SERVICEVAR -service- -name- ▶▶
```

### Use

GET SERVICEVAR prints the value of the variable *name* bound to the service *service*.

---

## LIST SERVICEVARS

### Syntax

```
▶▶ LIST SERVICEVARS -service- ▶▶
```

### Use

LIST SERVICEVARS prints all of SysVInit's global variables tied to service *service*.

---

## LOG

### Syntax

```
▶▶ LOG text ▶▶
```

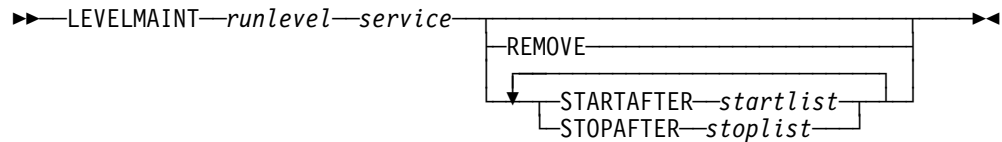
### Use

LOG prints out whatever is in the free-form string *text* with spaces replaced by underscores. It is primarily useful for annotation of console logs to highlight unexpected behavior.

---

## LEVELMAINT

### Syntax



### Use

LEVELMAINT is the command used to actually manipulate runlevels. LEVELMAINT will first attempt to acquire a write link to the disk containing the *runlevel* NAMES file. If this does not succeed, the command fails.

LEVELMAINT will then take action depending on the command.

If REMOVE is specified, LEVELMAINT will remove *service* from the specified runlevel. If this causes the specified level to be devoid of any virtual machines, that runlevel and its associated NAMES file will be deleted. STARTAFTER and STOPAFTER are meaningless and ignored in this case.

Otherwise, *service* will be added to *runlevel* NAMES. If *runlevel* did not previously exist, it is now created.

If either STARTAFTER or STOPAFTER is specified (either, both, or neither may be) then the next argument must be a colon-separated list of service machines. *service* cannot START in runlevel *runlevel* until all machines listed in STARTAFTER are in state AVAIL. Likewise, it cannot STOP in runlevel *runlevel* until all machines are in state UNAVAIL or OFFLINE.

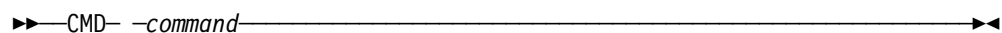
Note that this only applies to RUNLEVEL processing; *service* may be called to attempt to START or STOP the service at any time, regardless of the state of its dependencies. LEVELMAINT itself does not change the state of *service*. Use the SERVICE or the RUNLEVEL command for this.

LEVELMAINT is intended to provide a simple interface for specifying new service machine startups within a specific runlevel without the need to manually modify any of AUTOLOG1's control files. Thus it is well-suited to use by automated installation tools.

---

## CMD

### Syntax



## Use

CMD executes the command *command* in AUTOLOG1's context.

This command does not go through the SysVInit authorization processing: rather, because it allows the user to execute privileged commands, the authorized user must be specified in SYSVINIT RTABLE.

By default, MAINT and OPERATOR at the local node are the only users privileged to execute this command.

---

## STOP

### Syntax

▶▶—STOP—◀◀

### Use

STOP halts the PROP processor but leaves all started virtual machines running. It is not a useful command during normal operation but may be helpful when debugging SysVInit dispatch errors.

This command does not go through the SysVInit authorization processing: rather, because it kills the SysVInit processor, the authorized user must be specified in SYSVINIT RTABLE.

By default, MAINT and OPERATOR at the local node are the only users privileged to execute this command.

---

## SHUTDOWN

### Syntax

▶▶—SHUTDOWN—◀◀

### Use

SHUTDOWN switches to runlevel SHUTDOWN (which stops all machines in the current runlevel and starts no new ones) and if that is successful, shuts down the VM system with the CP SHUTDOWN command.

This command stops the entire machine. Because of its extraordinary danger, the authorized user must be specified both in SYSVINIT RTABLE *and* as an ADMIN user in SysVInit authorization processing.

By default, MAINT and OPERATOR at the local node are the only users privileged to execute this command.

---

# SHUTDOWN REIPL

## Syntax

▶▶—SHUTDOWN REIPL—————▶▶

## Use

SHUTDOWN REIPL switches to runlevel SHUTDOWN (which stops all machines in the current runlevel and starts no new ones) and if that is successful, shuts down and restarts the VM system with the CP SHUTDOWN REIPL command.

This command stops and restarts the entire machine. Because of its extraordinary danger, the authorized user must be specified both in SYSVINIT RTABLE *and* as an ADMIN user in SysVInit authorization processing.

By default, MAINT and OPERATOR at the local node are the only users privileged to execute this command.

## Example of Use

We will presume that DEFAULT NAMES has been set up as seen earlier, and that SysVInit is running in the AUTOLOG1 virtual machine. Now we wish to make the following changes.

1. TCPIP needs a start timeout of 45 seconds. Perhaps it has an OSA card that takes a long time to settle.
2. DIRMAINT should start in runlevel DEFAULT, but only after the SFS server machines (VMSERVS, VMSERVR, VMSERVU) are up.
3. DATAMOVE should behave like DIRMAINT.
4. LINUX01 should start automatically, but only after TCPIP is available.
5. Conversely, TCPIP should not shut down until after LINUX01 is down.
6. LINUX01 is a Linux guest that should inherit the LINUX template.
7. LINUX01 should be permitted to couple to VSWITCH VSW1.
8. After all this configuration is complete, LINUX01 should be started

From a userid privileged to execute the CMD command (and therefore specified in SYSVINIT RTABLE: MAINT or OPERATOR, by default) the following commands would be needed (the two indented lines should each be typed as a single long line, and the comments should be omitted).

---

```

TELL AUTOLOG1 SET SERVICEVAR TCPIP STARTTIMEOUT 45          /* Req. 1 */
TELL AUTOLOG1 LEVELMAINT DEFAULT DIRMAINT STARTAFTER        /* Req. 2 */
    VMSERVS:VMSERVU:VMSERVR
TELL AUTOLOG1 LEVELMAINT DEFAULT DATAMOVE STARTAFTER        /* Req. 3 */
    VMSERVS:VMSERVU:VMSERVR
TELL AUTOLOG1 LEVELMAINT DEFAULT LINUX01 STARTAFTER TCPIP   /* Req. 4 */
TELL AUTOLOG1 LEVELMAINT DEFAULT TCPIP STOPAFTER LINUX01   /* Req. 5 */
TELL AUTOLOG1 SERVICE LINUX01 ADD LIKE LINUX                /* Req. 6 */
TELL AUTOLOG1 SET SERVICEVAR LINUX01 VSWITCH VSW1          /* Req. 7 */
TELL AUTOLOG1 SERVICE LINUX01 START                          /* Req. 8 */

```

---

Figure 20. Example of use



---

## Future Plans

The following features are planned in the near future.

- The ability to run from SFS as well as from minidisks.
- A command similar to Linux's *chkconfig* to allow updating of many runlevels simultaneously and facilitate service insertion by installation scripts.
- A dependency checker to make sure there are no dependency loops or unsatisfiable dependencies in a runlevel.
- An automated installation tool.



---

## Technical Support

If you need help, please send e-mail to [beta-s5i@sinenomine.net](mailto:beta-s5i@sinenomine.net). Support will be provided on a best-effort basis.



---

# Appendices

---

## Appendix A. Files used by SysVinit

---

---

### SYSVINIT SRTABLE

SYSVINIT SRTABLE is the PROP table that defines the actions taken when a particular command is received. This file should be copied to SYSVINIT RTABLE, the node id replaced with the local node name, and any further authorizations you want to add added to it.

---

### DEFAULT NAMES

DEFAULT NAMES is the initial DEFAULT runlevel. It initially contains only SYSPERF EXEC, but should have its contents replaced with contents derived from PROFILE ORIG. It should also, generally speaking, be the default runlevel invoked at SysVinit startup, and should represent normal operation of the VM system.

---

### DISPATCH EXEC

DISPATCH EXEC is the executable called by PROP in the main event loop of SysVinit. It should not require customer modification.

---

### PROFILE S5IEXEC

PROFILE S5IEXEC is the replacement for AUTOLOG1's PROFILE EXEC on its 191-disk. It only needs modification if AUTOLOG1 192 is not the disk containing the SysVinit package.

---

### RUNLEVEL EXEC

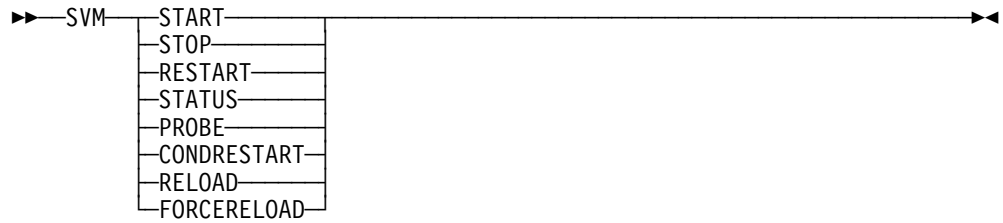
RUNLEVEL EXEC is invoked by DISPATCH EXEC in response to a RUNLEVEL command sent to the AUTOLOG1 user. It manages the transition between the current runlevel (if any) and the runlevel specified on its command line. It should not be invoked directly by the user.

---

### SVM EXEC

SVM EXEC is a template for the script invoked to start and stop service machines. For many machines it will be sufficient to copy SVM EXEC to *userid* EXEC using SERVICE *userid* ADD.

### Syntax



## Use

In most cases, the SVM EXEC default behavior will not require any changes. A description of what each command is intended to do follows.

### START

Starts the service. Issues a warning, but returns successfully, if the service is already running. On successful conclusion, puts the service into state INIT.

### STOP

Stops the service. Issues a warning, but returns successfully, if the service is already stopped. On successful conclusion, puts the service into state UNAVAIL.

### RESTART

Stops the service. If the stop is successful, starts the service.

### STATUS

Queries the service's STATUS GLOBALV. Does not perform any tests to determine whether the actual state of the service matches the stored status.

### PROBE

Attempts to determine the actual state of the service by querying the service virtual machine, and stores that information in the STATUS global variable. It makes this determination in the following way:

**AVAIL** Service is running disconnected. This is the default active state.

**UNAVAIL** Service is logged off or the service virtual machine does not exist.

**OFFLINE** Service is logged on to a logical device. This usually indicates that the service is offline for administrative reasons.

**UNKNOWN** Any other response from the QUERY results in the service being assigned state UNKNOWN.

### CONDRESTART

Issues a PROBE to determine the state of a service machine. If the machine is in state AVAIL, then it is restarted. Otherwise no action is taken.

### RELOAD

Requests that the service reload its configuration files. The service may ignore this request if it believes that its configuration is current.

### FORCERELOAD

Demands that the service reload its configuration files. The service may not ignore this request.

## Use

The user is expected to copy *SVM* *exec* to *service* *EXEC* for each service created. The resulting executable may then need to be modified. The following sections describe the desired behavior of each command.

## START

START attempts to start the service. For service machines in which there is only one function per machine (i.e., most CMS machines), XAUTOLOGging the machine will usually suffice. However, for machines which provide multiple services (e.g. a Linux guest providing DNS and file service) it may be necessary to replace XAUTOLOG with a more sophisticated startup procedure.

START puts the service into state INIT. Generally, after *starttimeout* seconds have elapsed, a PROBE will be run to test whether the service appears to be available. RUNLEVEL handles that on the user's behalf by default.

## STOP

STOP attempts to shut down the service. For single-function guests, this usually means logging the machine off. STOP first tries sending a signal and waiting up to *stoptimeout* seconds (defaults to 30 seconds, which may be overridden by a global timeout, which may in turn be overridden by a service-specific timeout).

If the guest is not enabled for SIGNAL SHUTDOWN, or if the signal timeout expires, then STOP attempts to force the guest. It checks periodically, and if the guest is still logged on at the expiration of *stoptimeout*, then the service is marked UNKNOWN. A successful logging off of the guest puts it into state UNAVAIL.

## RESTART

RESTART does a STOP and then a START of the service and fails if either of those actions fails.

## STATUS

STATUS simply reads and reports the status variable set for *service*. It does not actually attempt to verify whether the service is available; that job is reserved for PROBE.

## PROBE

It is PROBE's job to determine whether a service is actually available and update a service's status depending on its determination. The default implementation assumes a single-service machine, and that a service running disconnected should be in state AVAIL, a service logged on at a terminal should be in state OFFLINE (that is, offline for administrative action), and that a service that is not logged on is UNAVAIL.

Obviously, this rather crude test fails for multiservice machines, and in general is not helpful for determining the actual availability of a service. PROBE is a likely candidate for replacement with a more sophisticated test that actually attempts to exercise the provided service.

## CONDRESTART

CONDRESTART issues a PROBE; only if the service is in state AVAIL will it then issue a RESTART. It must not restart a service that was not already running.

## RELOAD

RELOAD has no default implementation. Its purpose is to request that the service reload its configuration; for a DNS server, this might be rereading its zone files, for instance. The service is free to ignore a RELOAD request if it believes its configuration is current. This allows a mechanism for conditional reloading of configuration if that reloading is an expensive action (e.g. reloading service tables in a database).

## FORCERELOAD

FORCERELOAD also lacks a default implementation. It does the same thing as RELOAD, except that the application may not ignore a FORCERELOAD command, even if it believes that its configuration has not changed. FORCERELOAD should therefore be used sparingly, as it may trigger expensive actions.

---

## DUMMY EXEC

DUMMY EXEC responds to the same commands as SVM EXEC, and sets its status variable, but does not actually start a service. It is useful as a template for performing performance adjustments, or as a placeholder service for use in conjunction with STARTAFTER and STOPAFTER for service grouping.

---

## SYSPERF EXEC

SYSPERF EXEC is derived from DUMMY EXEC and is present in the shipped DEFAULT runlevel. It includes the action variables LDUBUF, STORBUF, and DSPBUF to control system-wide performance settings.

---

## LINUX EXEC

LINUX EXEC is functionally equivalent to SVM EXEC, but includes Action Variables GLAN and VSWITCH to control guest coupling to restricted Guest LANs or VSWITCHes. It is intended as a template on which to base Linux guest startup scripts.

---

## SNAUME TEXT

SNAUME TEXT is the message repository for Sine Nomine Associates applications.

---

## \* S5IHELP

The S5IHELP files are the command help for the SysVInit commands.

---

## \* HELPMSG

The HELPMSG files are the message help text for messages emitted by SysVInit.

# **System V Init Quick Installation and Operation Guide 1.0**

Document Number VM01-0001-01

January 18, 2006

Sine Nomine Associates

43596 Blacksmith Square  
Ashburn, VA 20147

**Release 1.0**