Automated Startup and Sequencing of the CS-Studio Phoebus Alarm System Core Programs

Peter Bonneau, Mary Ann Antonioli, Aaron Brown, Pablo Campero, Brian Eng, George Jacobs, Mindy Leffel, Tyler Lemon, Marc McMullen, and Amrit Yegneswaran
Physics Division, Thomas Jefferson National Accelerator Facility, Newport News, VA 23606
December 15, 2022

The CS-Studio Phoebus alarm system requires core programs to be started and initialized in a sequence before they be used. An automated startup and sequencing of the programs, which simplifies the startup and makes the system more reliable, has been developed, tested, and implemented.

The Phoebus alarm system's core programs must be running at all times. All other alarm system programs use the core programs for communication between the alarm system programs, the Phoebus user interface, and for communication to external devices and programs.

During development of the Phoebus alarm test system [1] [2], it was discovered that the core programs must be started and initialized in a sequence before the alarm system can be used. The order of the startup program sequence—Apache Kafka Zookeeper, Apache Kafka server, procServ, and Phoebus alarm server—can be done manually.

To simplify and make the alarm system startup more reliable, an automated startup and sequencing has been developed, Fig. 1 and Table I, that is done when the Linux alarm system computer is booted.

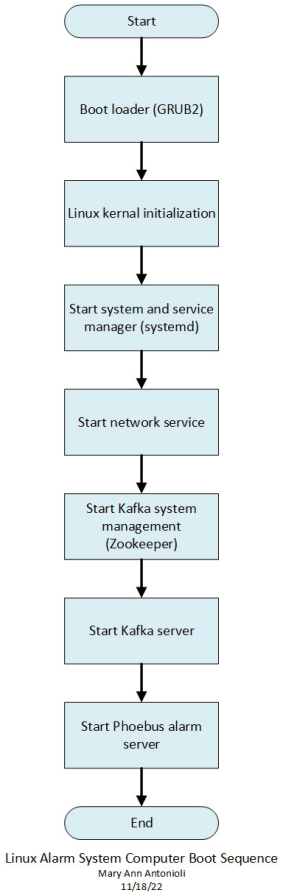

FIG.1. Phoebus Alarm computer boot sequence.

Alarm system program name	Automatic boot sequence #	Program function
Kafka Cluster Management (Zookeeper. service)	1	Kafka management for alarm system inter-program communication; as a prerequisite, the computer network must be initialized and working
Kafka server (Kafka.service)	2	Hosts the three alarm system message communication streams; upon initialization, the alarm state, alarm command, and alarm configuration streams are active and ready to service any of the alarm sub-programs
procServ (alarm_server. service)	3	Interactive command wrapper that allows remote access via telnet to the alarm system command console while the system is in operation; launches Phoebus Alarm Server program
Alarm server	4	Monitors EPICS process variables for alarm conditions via channel access; stores alarm configuration settings for each process variable

TABLE I. Alarm program automatic startup sequencing during Linux boot.

Following the execution of the Linux boot loader and the initialization of the Linux kernel, the system and service manager for Linux is started. Referred to as *systemd*, the Linux service manager bootstraps user space and manages user processes, and provides device management, login management, network connection management, and event logging. The alarm system uses systemd for the automated startup and sequencing of the core programs.

A configuration and command file with the extension *.service* is used to encode information and commands for programs controlled and supervised by systemd. For each of the core programs, an individual .service configuration command file was developed, tested, and implemented.

The zookeeper.service configuration command file is run first in the alarm system boot sequence. As a prerequisite, the computer networking must be initialized. Kafka Zookeeper is used for cluster management of the alarm system's inter-program communication.

The Kafka.service systemd configuration command file runs second in the sequence and starts and initializes the Kafka server, which hosts the alarm system's three message communication streams—alarm state, alarm command, and alarm configuration streams.

The alarm_server.service command file starts the programs procServ and the Phoebus alarm server. The interactive command wrapper program procServ allows users remote access via telnet to the alarm server command console while the system is in operation and launches the alarm server program, Fig. 2. The Phoebus alarm server monitors EPICS process variables for alarm conditions via channel access. The server also stores alarm configuration settings for each process variables.

In conclusion, an automated startup and sequencing of the CS-Studio Phoebus alarm system core programs has been developed, tested, and implemented. The automated startup sequence simplifies the alarm system startup and makes the system more reliable.

[1] P. Bonneau, et al., *Proposal to Implement Alarm System in Control System Studio Phoebus for the Hall C Neutral Particle Spectrometer*, DSG Note 2021-37, 2021.

[2] P. Bonneau, et al., *Development of the EPICS Software Input/Output Controller for Testing the Phoebus Alarm System of the Hall C Neutral Particle Spectrometer*, DSG Note 2022-06, 2022.


```
[bonneau@fedora Desktop]$ sudo systemctl start alarm_server.service
[sudo] password for bonneau:
[bonneau@fedora Desktop]$ sudo systemctl status alarm_server.service
● alarm_server.service - Phoebus Alarm Server
     Loaded: loaded (/etc/systemd/system/alarm_server.service; enabled; vendor preset: disabled)
     Active: active (running) since Thu 2022-11-17 17:51:17 EST; 19s ago
   Main PID: 7737 (procServ)
      Tasks: 63 (limit: 37873)
     Memory: 121.6M
        CPU: 2.031s
     CGroup: /system.slice/alarm_server.service
             ├─ 7737 /usr/bin/procServ --foreground --noautorestart --name alarm-server --chdir /home/bonneau/Downloads/phoeb
             ├─ 7742 /bin/sh ./alarm-server.sh -config Hall-C-NPS
             ├─ 7744 java -jar ./target/service-alarm-server-4.6.10-SNAPSHOT.jar -config Hall-C-NPS
             └─ 7787 /usr/lib/jvm/java-11-openjdk-11.0.17.0.8-2.fc35.x86_64/bin/java -classpath ./target/service-alarm-server

Nov 17 17:51:17 fedora systemd[1]: Started Phoebus Alarm Server.
```

FIG. 2. Automatic launch of alarm server via procServ.