NAME

stopped − event signalling that a job has stopped

SYNOPSIS

stopped JOB=*JOB* **INSTANCE**=*INSTANCE* **RESULT**=*RESULT* [**PROCESS**=*PROCESS*] [**EXIT_STATUS**=*STATUS*] [**EXIT_SIGNAL**=*SIGNAL*] [*ENV*]...

DESCRIPTION

The **stopped** event is generated by the Upstart **init**(8) daemon when an instance of a job has stopped. The **JOB** environment variable contains the job name, and the **INSTANCE** environment variable contains the instance name which will be empty for single-instance jobs.

If the job was stopped normally, the **RESULT** environment variable will be *ok*, otherwise if the job was stopped because it has failed it will be *failed*.

When the job has failed, the process that failed will be given in the **PROCESS** environment variable. This may be *pre-start*, *post-start*, *main*, *pre-stop* or *post-stop*; it may also be the special value *respawn* to indicate that the job was stopped because it hit the respawn limit.

Finally in the case of a failed job, one of either **EXIT_STATUS** or **EXIT_SIGNAL** may be given to indicate the cause of the stop. Either **EXIT_STATUS** will contain the exit status code of the process, or **EXIT_SIGNAL** will contain the name of the signal that the process received. The **normal exit** job configuration stanza can be used to prevent particular exit status values or signals resulting in a failed job, see **init**(5) for more information.

If neither **EXIT_STATUS** or **EXIT_SIGNAL** is given for a failed process, it is because the process failed to spawn (for example, file not found). See the system logs for the error.

init(8) emits this event as an informational signal, services and tasks started or stopped by this event will do so in parallel with other activity. It is typically combined with the **starting**(7) event by services when inserting themselves as a dependency.

Job configuration files may use the **export** stanza to export environment variables from their own environment into the **stopped** event. See **init**(5) for more details.

EXAMPLE

A service that wishes to be running whenever another service would be running, started before and stopped after it, might use:

```
start on starting apache
stop on stopped apache
```

A task that must be run after another task or service has been stopped might use:

```
start on stopped postgresql
```

SEE ALSO

starting(7) **started**(7) **stopping**(7) **init**(5)