launchd keywords for plists

By Dennis German

Archived: 2026-04-06 01:21:08 UTC

There does not appear to be a `RestartSec` (delay restart in seconds) as in linux `systemctl.`

`Label` *`string`* required key uniquely identifies the job `Program` *`string`* first argument of `execvp`. Default: `ProgramArguments[0]`.
Required in the absence of `ProgramArguments`. `ProgramArguments` *`array_of_strings`* second argument of `execvp`.
Required in the absence of `Program`. Read execvp(3) carefully `EnableGlobbing` *`bool`* use the glob(3) mechanism to update the program arguments before invocation. `Disabled` *`bool`* see `/private/var/db/launchd/com.apple.xpc.launchd/disabled.plist` deprecated: a hint to launchctl(1) that it should not submit this job to launchd when loading a job or jobs.
Does NOT reflect the current state of the job on the running system.
Query launchd for the presence of the job using the launchctl(1) `list` or use the ServiceManagement framework's SMJobCopyDictionary() method.

Conveys a default value, which maybe changed with `-w` of launchctl load and unload subcommands which do not modify the configuration file,
Only use this key if the provided on-demand and `KeepAlive` criteria are insufficient to describe the conditions under which job needs to run. The cost to have a job loaded in launchd is negligible, so there is no harm in loading a job which only runs once or rarely.

`UserName` *`string`* `GroupName` *`string`* If UserName is set and GroupName is not, the the group will be set to the default group of the user. `inetdCompatibility` *`dictionary`* expect to run as if it were launched from inetd. `Wait` *`bool`* corresponds to the "wait" or "nowait" option of inetd.
`true`: the listening socket is passed via the standard in/out/error file descriptors.
`false`: accept(2) is called on behalf of the job, and the result is passed via the standard in/out/error descriptors.
`LimitLoadToHardware` `LimitLoadFromHardware` `LimitLoadToHosts` *`array of strings`* applies to the hosts listed with this key. set kern.hostname in sysctl.conf(5) `LimitLoadFromHosts` *`array of strings`* applies to hosts NOT listed with this key. set kern.hostname in sysctl.conf(5) `LimitLoadToSessionType` *`string`* Applies to sessions of the type specified (example

```
<key>LimitLoadToSessionType</key>
     <array>
     <string>Aqua</string>  <string>Background</string>
     <string>LoginWindow&lt;/string>
     </array>
```

Used with `-S` to launchctl. `EnableTransactions` *`bool`* Uses `vproc_transaction_begin` and `n_end` to track transactions that need to be reconciled before the process can safely terminate.
If there are outstanding transactions launchd should avoid sending `kill` `OnDemand` *`bool`* was used in Mac OS X 10.4 to control whether a job was kept alive or not. The default was true. been deprecated and replaced in 10.5 `KeepAlive` . `KeepAlive` *`bool`* or
`dictionary of stuff` `true` : unconditionally keep the job alive. implies `RunAtLoad`
`false` : only demand will start the job.
Default : `false`

Dictionary of conditions to control if the job is keep alive(restarted), Multiple keys are ORed. If launchd finds no reason to restart the job, it falls back on demand based invocation.
Jobs that exit quickly and frequently when configured to be kept alive will be throttled to converve system resources.
`"Service only ran for 0 seconds. Pushing respawn out by 10 seconds."`

`SuccessfulExit` *`bool`*	`true` : job will be restarted if program exits with a status of zero. i.e. job succeded last time, run it again, if failed don't start it again `false` : job will be restarted … status not zero. i.e. job failed; run it again maybe inputs or conditions have changed and it will work this time. job was sucessful(finally(?)) don't start it again. Implies `RunAtLoad` since the job needs to run at least once to can get an exit status.
`PathState` *`dictionary of bool`*	keep alive if: keys are file-system paths. true: job will be kept alive as long as the path exists. false: job will be kept alive if the path does NOT exist. Used so jobs may create semaphores in the file-system.
`OtherJobEnabled` *`dictionary of bools`*	keys are the label of another job. true: job is kept alive as long as the other job is enabled. false: job is kept alive as long as the other job is disabled. Not to be used instead of IPC.
`Crashed`	Job will be restarted if exited due to `sigILL` , `sigSEGV` *… +CAUTION+
`NetworkState` *`bool`*	true: will be kept alive as long as the network is up, i.e. at least one non-loopback interface being up and having IPv4 or IPv6 addresses . false: will be kept alive as long as the network is down.

`RunAtLoad` *`bool`* job is launched once at the time the job is loaded. default false. `RootDirectory` *`string`* a directory to chroot(2) to before running `WorkingDirectory` *`string`* chdir(2) to before running . `EnvironmentVariables` *`dictionary of strings`* additional environmental variables set before running . `UserEnvironmentVariables` *`dictionary of strings`* additional environmental variables set before running . `Umask` *`decimalInteger`* hugh? ed passed to umask(2) before running . `TimeOut` *`seconds`* idle time out. default will be supplied by launchd for use by the job at check in time. (deprecated) `ExitTimeOut` *`seconds`* Time after sending `sigTERM` before sending `sigKILL` allowing for cleanup.
Default: 20 seconds, zero is (poorly) interpreted don't issue `sigKILL` . `ThrottleInterval` *`seconds`* minimun before respawn. Default: 10 seconds.
Processes that last less than *`seconds`* generate message:
`Service only ran for` *`n`* `seconds. Pushing respawn out by` *`n`* `seconds.` Notice: newsyslog, locationmenu, These sholud have a smaller `ThrottleInterval` in their plist DGG

The principle behind this is that jobs should linger around in memory in case they are needed in the near future, reduceng the latency of responses, encourages developers to amortize the cost of program invocation.

`InitGroups` *`bool`* whether `initgroups(3)` should be called before running . Default: true . `UserName` must be set. `WatchPaths` *`array of strings`* start if any one of the listed paths are modified.
mac os as of 11/20/20

```
/System/Library/LaunchDaemon
/Library/Extensions                               kernelmanagerd
/AppleInternal/Library/Extensions                 kernelmanagerd
/etc/postfix/aliases                              newaliases
/Library/Preferences/com.apple.symptoms.plist     symptomsd
/etc/nfs.conf                                     nfsconf
/System/Library/LaunchAgents
/Applications/                                    helpd
/Applications/Utilities/                          helpd
/Library/Apple/Library/Bundles/TCC_Compatibility.bundle/Contents/Resources/AllowApplicationsList tcc
/System/Library/Sandbox/TCC_Compatibility.bundle/Contents/Resources/AllowApplicationsList       tcc
/Library/Application Support/com.apple.TCC/MDMOverrides                                         tcc
/private/etc/com.apple.screensharing.agent.launchd              screensharing.agent
~/Library/Application Support/Steam/Steam.AppBundle/Steam ~/Library/LaunchAgents/valvesoftware.steam
```

`QueueDirectories` *`array of strings`* like `WatchPaths` watchs for modifications, if the path is a directory and not empty. `StartOnMount` *`bool`* start when a filesystem is mounted. `StartInterval` *`seconds`* start every *`seconds`* .
If the system was asleep, when the job should have run, it will be started **once** the next time the system wakes . `StartCalendarInterval` *`dictionary of integers or array of dictionary of integers`* The semantics are similar to `crontab(5).` If the system was asleep when the job should have run, it will be started **once**, the next time the system wakes. `<key>Minute</key><integer>`*`mm`*`</integer>` … Hour *`hh`* Day *`dd`* Month *`mm`* Weekday *`w`* `0 and 7` are Sunday. Missing arguments are wildcard (i.e. **every**).

`StandardInPath` *`string`* `StandardOutPath` *`string`* `StandardErrorPath` *`string`* Example: `<key>StandardErrorPath</key><string>/var/log/`*`label`*`.err</string>` `Debug` *`bool`* log mask temporarily set to LOG_DEBUG `WaitForDebugger` *`bool`* instructs kernel to have the job wait for a debugger to attach before job is started. `SoftResourceLimits` Resource limits.

- `CPU` *`seconds`* maximum cpu time
- `FileSize` *`bytes`* largest size file created.
- `NumberOfFiles` *`integer`* maximum number of open files .
  Setting this in a system wide daemon sets `sysctl kern.maxfiles` (`SoftResourceLimits`) or `kern.maxfilesperproc (HardResourceLimits`) in addition.
- `NumberOfProcesses` *`integer`* maximum simultaneous processes for this user id. Setting in a system wide daemon will set `sysctl kern.maxproc (SoftResourceLimits)` or `kern.maxprocperuid (HardResourceLimits)` in addition
- `MemoryLock` *`bytes`* maximum lock into memory using `mlock` (2)
- `Data` *`bytes`* maximum size of the data segment, defines how far a program may extend its break with `sbrk` (2)
- `ResidentSetSize` *`bytes`* maximum a process's resident set size may grow. This imposes a limit on the amount of physical memory to be given to a process; if memory is tight, the system will prefer to take memory from processes that are exceeding their declared resident set size.
- `Stack` *`bytes`* maximum of stack segment ; this defines how far a program's stack segment may be extended. Stack extension is performed automatically by the system.

```
<key>SoftResourceLimits</key> <dict> <key>NumberOfFiles</key> <integer>58</integer> </dict>
```

`HardResourceLimits`
`<dict> <key>` *`key`* `⋯` *`integer`* `⋯` `ProcessType` *`string`* Intended purpose of the job for applying resource limits.
Default: light(sic) resource limits to the job, throttling its CPU usage and I/O bandwidth.

- `Standard` equivalent to no `ProcessType` being set.
- `Background` does work not directly requested by the user. The resource limits applied are intended to prevent them from disrupting the user experience. `<!-- slow down -->`
- `Adaptive` move between the Background and Interactive classifications based on activity over XPC connections. See `xpc_transaction_begin`
- `Interactive` Critical to maintaining a responsive user experience, run with the same resource limitations as apps, i.e. none.
  Use if an app's ability to be responsive depends on it, and cannot be made `Adaptive` .

`LegacyTimers` *`bool`* Default: timers created by launchd jobs are coalesced, batching the firing of timers with similar deadlines.
If `true` , timers will not be coalesced with others `AbandonProcessGroup` *`bool`* true: do NOT kill processes with the same process group if this job dies.
Default: when a job dies, launchd kills any remaining processes with the same process group ID as the job. `Nice`

integer <key>Nice</key> <integer>10</integer> <!-- slow down --> LowPriorityIO *bool* <key>LowPriorityIO</key> <true/> <!-- slow down --> LaunchOnlyOnce *bool* Job can only be run once, i.e. the job cannot be safely respawned without a reboot. MachServices *dictionary of bools or a dictionary of dictionaries* to be registered with the Mach bootstrap sub-system.
Keys are the names of services to be advertised. The value must be <true/> .

```
MachServices
    <dict>
        <key>com.apple.libquitd</key> <true/>
        <key>com.apple.spinreporterd</key> <true/>
        <key>com.apple.spindump</key> <true/>
    </dict>
```

MachServiceLookupPolicies

- ResetAtClose
  If false , the port is recycled, thus leaving clients to remain oblivious to the demand nature of job.
  If true , clients receive port death notifications when the job lets go of the receive right. Not all clients are able to handle this behavior.

  The port will be recreated atomically with respect to bootstrap_look_up() calls, so that clients can trust that after receiving a port death notification, the new port will have already been recreated.
  default: false .

- HideUntilCheckIn *bool* Reserve the name in the namespace, but cause bootstrap_look_up() to fail until the job has checked in with launchd.
  Finally, for the job itself, the values will be replaced with Mach ports at the time of check-in with launchd.

Sockets *dictionary of dictionaries... OR dictionary of array of dictionaries...* demand sockets that can be used to let launchd know when to run the job.
The job must check-in to get a copy of the file descriptors using APIs outlined in launch .
The keys of the top level Sockets dictionary can be anything. for the application developer to use to differentiate which descriptors correspond to which application level protocols (e.g. http vs. ftp vs. DNS...). At check-in time, the value of each key will be an array of descriptors.
Daemon/Agent writers should consider all descriptors of a given key to be to be effectively equivalent, even though each file descriptor likely represents a different networking protocol which conforms to the criteria specified in the job configuration file.

inputs to call getaddrinfo :

- SockType *string* default stream other valid values: dgram and seqpacket .
- SockPassive *bool* whether listen(2) or connect(2) should be called on the created file descriptor. Default: true ("to listen").
- SockNodeName *string* node to connect(2) or bind(2) to.
- SockServiceName *string* service on the node to connect(2) or bind(2) to.
- SockFamily *string* request that "IPv4" or "IPv6" socket(s) be created.

- `SockProtocol` *`string`* protocol to be passed to socket(2). `TCP`
- `SockPathName` *`string`* set to `Unix`, specifies the path to connect(2) or bind(2) to.
- `SecureSocketWithKey` *`string`* variant of SockPathName.
  Instead of binding to a known path, a securely generated socket is created and the path is assigned to the environment variable that is inherited by all jobs spawned by launchd.
- `SockPathMode` *`DECIMALinteger`* mode of the socket.
- `Bonjour` *`bool or string or array of strings`* request the service be registered with mDNSResponder(8).
  If the value is bool, the service name is inferred from `SockServiceName`.
- `MulticastGroup` *`string`* request that the datagram socket join a multicast group.
  If hostname, getaddrinfo(3) will be used to join the correct multicast address for a given socket family.
  If an explicit IPv4 or IPv6 address is given, `SockFamily` family must be set

---