목차

Systemd Unit 옵션 3
[Unit] Section 3
[Service] Section 3
[Install] section 6
참조링크 6

Systemd Unit 옵션

[Unit] Section

- Description= - 해당 유닛에 대한 상세한 설명을 포함한다.
- Requires= - 상위 의존성을 구성한다. 목록의 유닛이 정상적일 경우 유닛이 시작된다. (필요 조건)
- RequiresOverridable= - Requires= 옵션과 유사하다. 하지만 이 경우 **사용자에 의해서** 서비스가 시작하는데 상위 의존성이 있는 유닛 구동에 실패 하더라도 이를 무시하고 유닛을 시작한다. (즉 상위 의존성을 무시한다.) 자동 시작의 경우 적용 되지 않는다.
- Requisite=, RequisiteOverridable= - Requires= 와 RequiresOverridable= 와 유사하다. 상위 의존성 유닛이 시작되지 않은 경우 즉시 실패를 반환한다.
- Wants= - Requires= 보다 다소 완화된 옵션이다. 상위 의존성의 유닛이 시작되지 않더라도 전체 수행과정에 영향을 끼치지 않는다. 이 옵션은 하나의 유닛을 다른 유닛과 연계할 경우 사용하게 된다. (충분 조건)
- BindsTo= - Requires= 와 매우 유사하다. systemd 개입 없이 갑작기 서비스가 사라진 경우 (가령 NIC 가 물리적 장애가 난 경우) 해당 유닛도 같이 중지 하도록 설정하도록 한다.
- PartOf= - Requires= 와 매우 유사하다. 상위 의존성의 유닛을 중지하거나 재시작하는 경우 해당 유닛 또한 중지나 재시작을 수행한다. 만일 오라클과 오라클 리스너의 경우에서 처럼 하나의 서비스 다른 하나의 종속성을 가지게 되는 경우 필요한 설정이다.
- Conflicts= - 역의 관계를 설정한다. 만일 **유닛1** 의 Conflicts= 설정이 **유닛2** 로 되어 있다면 **유닛1**이 시작된 경우 **유닛2**가 중지되고, **유닛1**이 중지된 경우 **유닛2**가 시작한다. 이 옵션은 After= 와 Before= 옵션과는 독립적으로 작동한다. 각 서비스가 반대의 역활을 하거나 혹은 보조적인 서비스 역활을 수행한다면 서비스 실행 관리에 편리할 것 같다.
- Before=, After= - 유닛 시작의 전후 관계를 설정한다. 해당 설정은 Requires= 설정과는 독립적이다. Before= 에 나열된 유닛이 시작되기 전에 실행하고 After= 은 해당 유닛이 시작된 이후 나열된 유닛이 실행한다. 이 설정은 시스템이 종료(shutdown) 될때는 역으로 작동하게 된다.
- OnFailure= - 해당 유닛이 **실패** 상태가 되면 수행할 유닛 목록 (예 / 파일 시스템 마운트 실패시 복구 모드 수행)
- PropagatesReloadTo=, ReloadPropagatedFrom= - 리로드(reload) 명령을 다른 유닛에게 전달하거나 혹은 전달받아 해당 유닛도 리로드(reload)하게 된다.
- RequiresMountsFor= - 절대 경로로 지정하여 유닛을 구동하는데 필요한 마운트 목록을 자동으로 구성하여 Requires=, After= 을 수행한다. 즉 필요한 마운트 경로가 준비되어 있는지 점검하고 마운트를 미리 진행한다.
- OnFailureIsolate=[yes|no] - **yes**인 경우 OnFailure= 에 선언된 리스트와는 격리모드(isolation mode)로 작동한다. 즉 해당 유닛에 종속성이 없는 모든 유닛은 중지 되게 된다. OnFailureIsolate=yes 인 경우 OnFailure= 옵션에 오직 하나의 유닛만 설정 할 수 있다.

[Service] Section

- Type=[simple|forking|oneshot|notify|dbus] - 유닛 타입을 선언한다.
  - simple - (기본값) 유닛이 시작된 경우 즉시 systemd 는 유닛의 시작이 완료되었다고 판단한다. 다른 유닛과 통신하기 위해 소켓을 사용하는 경우 이러한 설정을 사용하면 안된다.
  - forking - 자식 프로세스를 생성이 완료되는 단계까지를 systemd 가 시작이 완료되었다고 판단하게 된다. 부모 프로세스를 추적할 수 있도록 PIDFile= 필드에 PID 파일을 선언해 주어야 한다.
  - oneshot - simple 과 다소 유사하지만 단일 작업을 수행하는데 적합한 타입니다. 또한 실행 이후 해당 실행이 종료되더라도 RemainAfterExit=yes 옵션을 통해 유닛이 활성화 상

태로 간주할 수 있다.
    - notify - simple 과 동일하다. 다만 유닛이 구동되면 systemd 에 시그널을 보낸다. 이때 시그널에 대한 내용은 libsystemd-daemon.so 에 선언 되어 있다.
    - dbus - DBUS 에 지정된 BusName 이 준비될때까지 대기한다. 즉 DBUS 준비가 완료된 이후 유닛이 시작되었다고 간주한다.
- RemainAfterExit=[yes|no] - 유닛이 종료 이후에도 유닛이 활성화 상태로 판단한다.
- GuessMainPID=[yes|no] - 이 옵션는 Type=forking 가 설정되어 있고 PIDFile= 설정이 되어 있지 않은 경우에 작동한다. systemd 가 유닛이 정상적으로 시작되었는지 판단이 명확하지 않는 경우 사용된다. 만일 여러개의 데몬으로 구성된 유닛의 경우 잘못된 PID 추측이 발생할 수 있다. 그로인해 오류 검출이나 자동 재시작 등의 작업이 불가능 할 수 있다. 이 옵션은 이러한 문제를 방지하는 기능을 한다.
- PIDFile= - PID 파일을 지정한다. 만일 유닛 타입이 forking 이라면 해당 설정을 추가해 주어야 한다. (절대 경로 사용)
- BusName= - D-Bus 의 버스 이름을 지정한다. Type=dbus 인 경우 필수 사항이다. 다른 Type 의 경우라도 D-Bus 버스 이름을 다르게 사용하는 경우 별도로 설정을 해주는 것이 좋다.
- Environment= - 해당 유닛 에서 사용할 환경 변수를 선언한다. 또한 반드시 Exec*= 옵션보다 상단에 위치해야 한다. 예제는 아래와 같다.

```
Environment="ONE=one" 'TWO=two two'
```

- EnvironmentFile= - 해당 유닛에서 사용할 환경 변수 파일을 선언한다. 환경 변수 파일에서 # 와 ; 로 시작되는 라인은 주석으로 처리된다. Environment= 와 같이 사용하는 경우 Environment= 옵션값이 먹게 된다. 또한 반드시 Exec*= 옵션보다 상단에 위치해야 한다.
- ExecStart= - 구동 명령어(스크립트)을 선언한다. 실행 명령어는 반드시 절대 경로 또는 변수(${STRINGS} 따위) 로 시작해야 한다. 다중 명령어를 지원한다. 예제는 아래와 같다.

```
ExecStart="commnad 1; command 2; command 3" 또는
ExecStart="commnad 1"
ExecStart="command 2"
```

- ExecStop= - 중지 명령어(스크립트)를 선언한다. ExecStart= 동일하게 사용하면 된다. 중지 방식은 KillMode= 로 지정된다.
- KillMode=[control-group|process|none] - 중지 방법에 대해서 선언한다.
    - control-group 은 해당 유닛의 그룹까지 모두 중지 시킨다. 기본값이다.
    - process 은 해당 유닛 즉 메인 프로세스만 중지 시킨다.
    - none 은 아무런 액션을 하지 않는다.
    - 그룹이란 유닛과 그 유닛에 종속성을 가지는 유닛의 묶음을 뜻한다.
- ExecReload= - 리로그(reload) 를 수행할 명령어를 선언한다.
- ExecStartPre=, ExecStartPost=, ExecStopPre=, ExecStopPost= - 유닛 시작, 중지 등의 엑션과 관련하여 수행할 추가 명령어를 선언한다. 사용법은 ExecStart= 동일하게 사용하면 된다.
- RestartSec= - 재시작 명령을 수행할때 중지 이후 다시 시작하는데 대기(sleep)하는 시간을 설정한다. 기본값은 100ms 이다. 각각 min, s, ms 단위로 설정한다. 해당 설정은 Restart= 옵션이 있는 경우에만 적용된다.
- TimeoutStartSec= - 유닛이 시작하는데 대기하는 시간을 설정한다. 기본값은 90초(90s)이다. 만일 Type=oneshot 인 경우 해당 설정이 해당 설정이 적용되지 않는다. 만일 시작 시간을 대기하지 않고 무한정 리턴값을 기다리게 설정할려면 TimeoutStartSec=0 으로 설정해 주면 된다.
- TimeoutStopSec= - 옵션을 중지하는데 대기하는 시간을 설정한다. 기본값은 90초(90s)로 위의 TimeoutStartSec= 옵션과 동일하게 TimeoutStopSec=0 으로 설정하면 무한정 리턴값을 기다리게 된다. TimeoutStopSec 옵션에 설정된 값 안에 종료되지 않으면 SIGKILL 시그널을 보내서 강제로 종료하게 된다.
- TimeoutSec= - TimeoutStartSec= 와 TimeoutStopSec= 을 동시에 설정한다.

- WatchdogSec= - 유닛이 시작된 이후 유닛 상태 감시(keep-alive ping)할때의 상태 값을 리턴하는데 대기하는 시간을 설정한다. Restart= 옵션이 on-failure, always 인 경우 유닛을 자동으로 재시작하게되고 이때 WatchdogSec= 설정을 해주어야 한다. 기본값은 0 으로 유닛 상태 감시를 사용하지 않는다.
- Restart=[no|on-success|on-failure|on-watchdog|on-abort|always] - 유닛이 죽었을때나 혹은 WatchdogSec= 만큼의 시간 동안 응답이 없는 경우 재시작한다. ExecStartPre=, ExecStartPost=, ExecStopPre=, ExecStopPost=, ExecReload= 에 설정된 유닛의 경우에는 포함되지 않는다. 즉 해당 유닛에만 해당된다.
  - no (기본값), 유닛을 다시 시작하지 않는다.
  - on-success 는 유닛이 정상적으로 종료되었을 때만 재시작한다. 종료시에 0 값을 리턴하여 종료되었거나 SIGHUP, SIGINT, SIGTERM, SIGPIPE 등과 같은 시그널 또는 SuccessExitStatus= 설정에서 지정된 리턴 코드 목록에 따른 시그널에 대해서 모두 성공으로 인식해 재시작을 하게 된다.
  - on-failure 유닛이 비정상적으로 종료되었을때 재시작한다. 리턴값이 0 이 아닌 경우, core dump 와 같이 비정상적인 시그널을 받고 종료된 경우, 타임 아웃값내 응답이 없는 경우 등일때 재시작 하게 된다.
  - on-watchdog WatchdogSec= 에 설정된 시간내 응답이 없는 경우에만 재시작 한다.
  - on-abort 지정되지 않은 리턴값을 받은 경우 재시작을 한다.
  - always 종료 상태 등과 무관하게 무조건 재시작한다. (사용자가 중지해도 시스템이 다시 띄우게 된다. 설정된 유닛 중지 시 주의가 필요하다.)
- SuccessExitStatus= - 성공으로 판단할 시그널을 설정해 준다. 문법은 아래와 같다.

```
SuccessExitStatus=1 2 8 SIGKILL
```

- RestartPreventExitStatus= - 재시작을 방지할 리턴 코드를 설정한다. 재시작을 하지 않을 리턴 코드를 설정하는데 유용하다. 문법은 아래와 같다.

```
RestartPreventExitStatus=1 6 SIGABRT
```

- PermissionsStartOnly=[yes|no] - User=, Group= 옵션 등과 같이 권한 설정 옵션을 적용하여 시작한다. 해당 설정은 ExecStart= 옵션에서만 적용 되며 ExecStartPre=, ExecStartPost=, ExecReload=, ExecStop=, ExecStopPost= 옵션에서는 적용되지 않는다.
- User=, Group= - 유닛의 프로세스를 수행할 사용자명, 그룹명 등을 지정한다.
- RootDirectoryStartOnly=[yes|no] - / 디렉토리를 지정한다. chroot() 함수를 사용하여 구동한다. jail 구성의 일반적인 형태이다. 해당 설정은 ExecStart= 옵션에서만 적용 되며 ExecStartPre=, ExecStartPost=, ExecReload=, ExecStop=, ExecStopPost= 옵션에서는 적용되지 않는다.
- RootDirectory= - chroot() 함수로 변경할 / 디렉토리를 지정한다.
- WorkingDirectory= - 프로세스의 작업 디렉토리를 지정한다. 별도의 지정이 없으면 유닛은 / 디렉토리를 작업 디렉토리로 사용한다. 특정 디렉토리에서 실행해야하는 프로세스에서 필요하다.
- NonBlocking=[yes|no] - 소켓 파일 디스크립션 (FD) 에 O_NONBLOCK 플래그를 선언한다. yes 일 경우 STDIN/STDOUT/STDERR 을 제외하고 모든 소켓에 O_NONBLOCK 플래그가 지정된다. 즉 non-blocking mode 로 작동하게 된다.
- NotifyAccess=[none|main|all] - 유닛 상태에 대해서 sd_notify() 함수를 사용하여 알림(notification) 소켓에 접근할 수 있도록 한다.
  - none 은 유닛 상태에 대한 모든 정보를 무시한다.
  - main 은 메인 프로세스에 대해서만 상태 정보 알림을 허용한다.
  - all 은 모든 유닛 즉 컨트롤 그룹의 유닛 상태 정보 알림을 허용한다.
  - Type=notify 또는 WatchdogSec= 가 설정된 경우 NotifyAccess= 을 접근 가능하게 (즉 main 이상) 설정해야 한다. NotifyAccess= 만 설정하고 값이 없는 경우 기본값은 main 이다.

- `Sockets=` - 유닛에서 사용하는 소켓의 이름을 지정한다. 기본으로 <유닛명>`.socket` 으로 생성되지만 지정된 이름으로 소켓을 사용하는 경우 별도의 설정이 가능하다. 또한 하나의 유닛에서 여러 소켓 목록을 일괄적으로 관리하는 경우 `Sockets=` 옵션이 여러번 사용 될 수도 있다. 만일 `Sockets=` 옵션이 아무런 설정값 없이 단독으로 사용되는 경우 소켓 목록이 리셋되게 된다.
- `StartLimitInterval=`, `StartLimitBurst=` - 위의 두 설정값을 이용하여 제한된 시간에 너무 많은 재시작 (`Restart=`) 이 발생되는 것을 방지해 준다. 기본값에 따르면 10초 간격으로 5번까지 서비스 시작을 허용하고 그 이상 더 재시작 이벤트가 발생하면 자동으로 재시작 하지 않도록 설정해 준다. (즉 1분내 5번 재시작 시도이후 복구 불가시 Fail 발생) 나중에 관리자가 수동으로 구동하여 복구 할 수 있도록 하여 무한대의 유닛 재시작 이벤트 발생을 방지한다.
  - `StartLimitInterval=` 옵션의 경우 기본값 10초(`10s`), `0` 으로 설정할 경우 비활성화한다.
  - `StartLimitBurst=` 옵션의 경우 기본값 5회
- `StartLimitAction=[none|reboot|reboot-force|reboot-immediate]` - 만일 복구 재시도가 제한된 설정 (`Service Recovery Limit > StartLimitInterval * StartLimitBurst`) 내에 마치지 못하면 다음 조치로 어떠한 방식의 작동을 할지 선언한다.
  - `none` (기본값), 아무런 액션도 하지 않습니다.
  - `reboot` 시스템을 재부팅 한다. (systemctl reboot 와 동일)
  - `reboot-force` 시스템을 강제 제부팅 한다. 단 **데이타 유실은 없다**. (systemctl reboot -f 와 동일, sync후 강제 재부팅)
  - `reboot-immediate` 시스템을 강제 재부팅 한다. **데이타의 유실이 있다**. reboot() 함수를 사용하여 즉각적인 재부팅을 수행한다. `sync` 과정없이 진행
- `Nice=` - 해당 유닛의 프로세스의 `nice` 값을 지정한다. `-20` 부터 `19` 까지 정수형으로 등록한다.
- `OOMScoreAdjust=` - `OOM(Out Of Memory)` killer 작동시 프로세스 조정값를 미리 지정할 수 있다. `-1000` 에서 `1000` 까지 정수형으로 등록한다.
- `UMask=` - `umask` 값을 선언한다. 별도의 설정이 없으면 기본값은 `0022` 이다.
- `SyslogFacility=` - 로그 카테고리를 설정할 수 있다. `kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, authpriv, ftp, local0, local1, local2, local3, local4, local5, local6, local7` 등의 값으로 설정 가능하다.
- `SyslogLevel=` - 로그 레벨을 설정할 수 있다. `emerg, alert, crit, err, warning, notice, info, debug` 등 설정이 가능하다.
- `TCPWrapName=` - TCP 래퍼를 사용하기위한 설정이다.
- `PAMName=` - PAM 보안 사설을 사용하기 위한 설정이다.

[Install] section

- `Alias=` - 유닛의 알리아스 이름을 지정한다. `systemctl enable` 명령어를 통해서 알리아스 이름으로 생성할 수 있다. 알리아스 이름은 유닛 파일 확장자(유닛 타입)를 가지고 있어야 한다. (`service`, `socket`, `mount`, `swap` 등이 있다. 예: httpd.service 의 Alias=apache.service)
- `WantedBy=`, `RequiredBy=` - `systemctl enable` 명령어로 유닛을 등록할때 등록에 필요한 유닛을 지정한다. 해당 유닛을 등록하기위한 종속성 검사 단계로 보면 된다. 따라서 해당 설정은 `[Unit]` 섹션의 `Wants=` 와 `Requires=` 옵션과 관계 있다.
- `Also=` - `systemctl enable` 와 `systemctl disable` 로 유닛을 등록하거나 해제할때 다른 유닛 또한 같이 등록, 해제를 하도록 구성할 수 있다.

참조링크

- https://www.freedesktop.org/software/systemd/man/systemd.unit.html
- https://www.freedesktop.org/software/systemd/man/systemd.service.html
- https://fmd1225.tistory.com/93

From:
https://atl.kr/dokuwiki/ - **AllThatLinux!**

Permanent link:
https://atl.kr/dokuwiki/doku.php/systemd_unit_%EC%98%B5%EC%85%98

Last update: **2023/09/05 01:45**