```
health_normal
broadcastEvent(new ev_HealthNormal());

ev_HealthChanged[health>normal]

health_low
broadcastEvent(new ev_HealthLow());

ev_HealthChanged[health>critical && health>normal]

health_critical
broadcastEvent(new ev_HealthCritical());

ev_HealthChanged[health>normal]

ev_HealthChanged[health<critical]
```
Attack events are quantized into a safe/under_attack state. This prevents unnecessary attack events being generated for each individual bullet strike, for instance. Continued attacks reset the safeTimer but do not generate events.
A Stateless Statechart. The listener generates the following events:

- ev_GrenadeThrown(Grenade)
- ev_GrenadeExploded(Grenade)
- ev_HasGrenade
- ev_NoGrenades
sensing

not_boarded

[vehicleExited()]/
broadcastEvent(new ev_VehicleExited(vehicle));

[vehicleBoarded()]/
broadcastEvent(new ev_VehicleBoarded(vehicle));

boarded

[vehicleLost()]/
broadcastEvent(new ev_VehicleOutOfView(vehicle));

[vehicleSpotted()]/
broadcastEvent(new ev_VehicleSpotted(vehicle));

[vehicleDestroyed()]/
broadcastEvent(new ev_VehicleDestroyed(vehicle));

[vehicleExited()]/
broadcastEvent(new ev_VehicleExited(vehicle));

[vehicleBurning()]/
broadcastEvent(new ev_VehicleBurning(vehicle));
A stateless Statechart. The listener broadcasts the following events:

- `ev_ItemSpotted(item)`
- `ev_ItemOutOfView(item)`
- `ev_ItemRemoved(item)`
- `ev_ItemAcquired(item)`
A stateless Statechart. The listener broadcasts new ev_PlayerSpotted(player), ev_PlayerLost(player), and ev_PlayerKilled(player) events.
A stateless Statechart. The listener will create and broadcast the following events:

ev_ObstacleSpotted(Obstacle)
ev_ObstacleRemoved(Obstacle)
Internal listener creates \textit{ev\_NewCommand} events.

\textit{ev\_NewCommand}/broadcastEvent(new \textit{ev\_CommandReceived(Command)})

\textit{ev\_IgnoreCommands}

\textit{ev\_ListenForCommands}
init

[rangedWeaponEquipped()]

[[meleeWeaponEquipped()]]

ev_WeaponChange[rangedWeaponEquipped()]

melee_equipped

broadcastEvent(new ev_MeleeWeaponEquipped())

ev_WeaponChange[meleeWeaponEquipped()]

ranged_equipped

broadcastEvent(new ev_RangedWeaponEquipped())

ammo_normal

broadcastEvent(new ev_AmmoNormal())

outOfAmmo

broadcastEvent(new ev_OutOfAmmo())

***NPCs don't run out of ammo in Halo, so tracking it is not necessary***

needs_reload

ev_ClipEmpty

ev_ReloadComplete
init

no_shield

broadcastEvent(new ev_NoShield())

![npcHasShield]

has_shield

broadcastEvent(new ev_HasShield())

![npcHasShield]

shield_full

broadcastEvent(new ev_ShieldFull())

shield_down

broadcastEvent(new ev_ShieldDown())

Internal listener generates ev_ShieldChange events

ev_ShieldChange[npc.shieldsUp]

ev_ShieldChange[npc.shieldsUp]