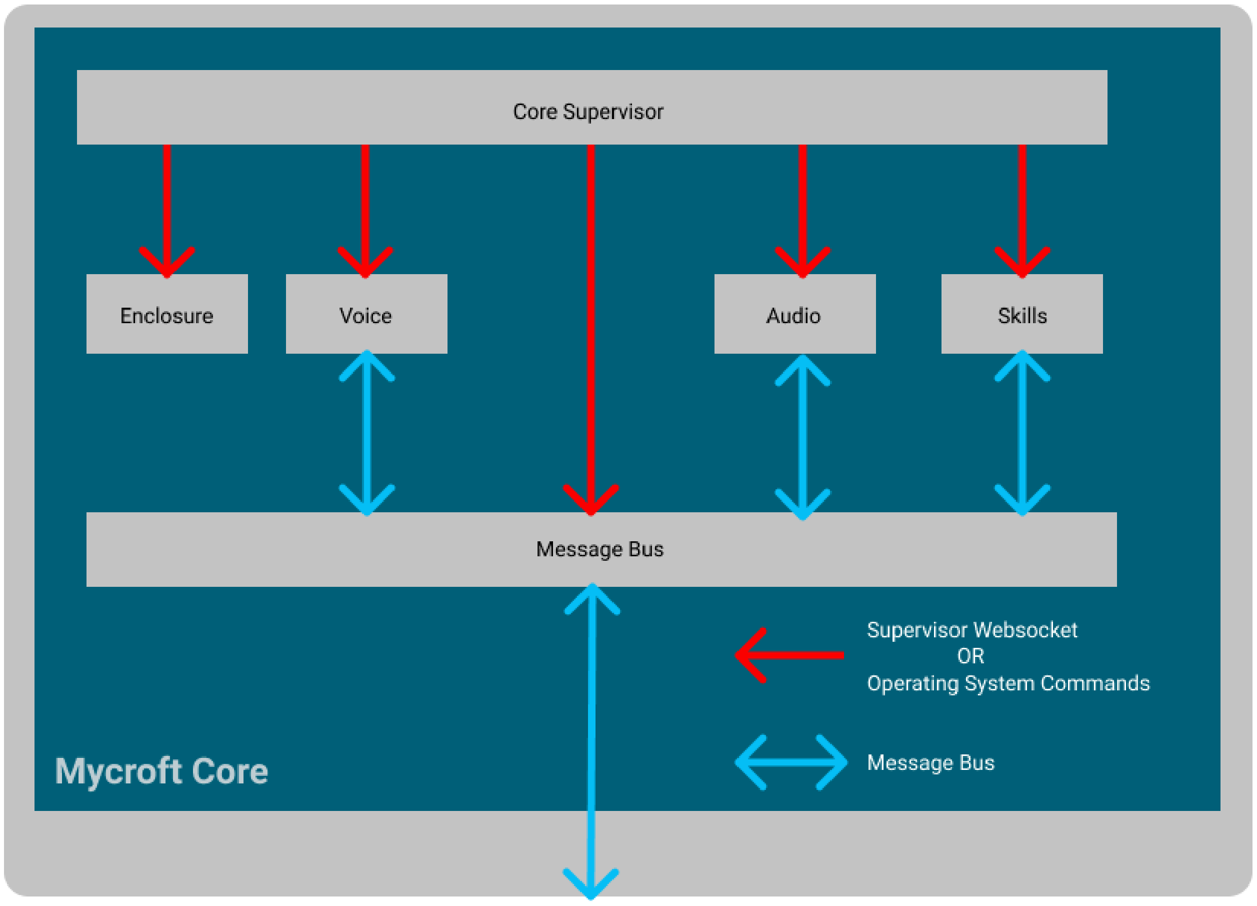
Mycroft Core Supervisor Service



Overview

The Mycroft Core Supervisor (MCS) service is how Mycroft is started on a system. It is a systemd service that launches, monitors the health of, and controls the Core Services required by the Mycroft voice assistant. The functionality of this service includes:

- Launching each Core Service as a daemon
- Monitoring each Core Service and coordinating the start-up sequence for a smooth user experience
    - Each Core Service goes through the following states upon launching:
        - Starting - it cannot respond to or generate Messages on the MMB
        - Loaded - it is ready to respond to and generate Messages on the MMB, but will not *generate* any Messages
        - Running - Normal operation: it can and will generate and respond to appropriate Messages
    - The transition from *Starting* to *Loaded* happens within the Core Service

    - The transition from *Loaded* to *Running* only happens when the Core Supervisor specifically tells that Core Service to do so.
- Monitoring various aspects of the health of each Core Service
    - is service running, healthy, ready, etc.
- Health checks are performed on a regular basis for each Core Service
- If a Core Service becomes unhealthy, it may restart it, other Core Services or even restart the device
- Additional watchdog checks that don't belong to a specific Core Service, such as the pre-existing microphone watchdog, are now located in the Core Supervisor
- It may restart Core Services on a periodic basis (such as daily) to mitigate memory leaks
- If a user opts in to data sharing, information about exceptions or crashes could be sent to Mycroft for diagnostic purposes.
- Responds to queries and commands to manually execute parts of the service.

Core Services

- Core Supervisor - this process
- Enclosure
- Skills
- Voice
- Audio
- Message Bus

Related Proposed Changes

https://mycroft-ai.gitbook.io/docs/mycroft-technologies/mycroft-core#process-status

Breaking Changes

- The following scripts were never intended for production use. They will be appropriately flagged as such and not included in production distributions:
    - `start-mycroft.sh`
    - `auto_run.sh`
    - `Stop-mycroft.sh`
- The above scripts will remain as placeholders, but will be moved to a subdirectory in the second major release after this change.

Questions / Tests

- Question - Should we rely on the message bus to send and receive messages from the Supervisor?
    - Without the bus, the rest of the system cannot operate, so it's already a critical dependency.

- Test - What happens when the message bus is down - intermittently or over an extended period?
- Question - should Services explicitly not communicate with each other during boot to prevent unintended dependencies
    - This might cause a new race condition where one service receives the approval to be Running, and pings another service that hasn't yet been switched to Running.
- Question - What should a Service (or Skills) be able to do during the Loaded status.
- Question - Do we need callbacks related to each status eg *service.on_error()*?
- Question - where does the CLI live? What is it? Is it just a utility included in the Core repo, but not part of the Core? If so,

Service Status

May also be used as overall system status

- Inactive - is not running, may mean that it's never been started
- Starting - state between the system starting it, but not finished loading.
- Loaded - but not yet running
- Running - all services are ready and operational
- Stopping - state between stop command issued and shutdown complete (inactive)
- Crashed - a critical error the service could not recover from
    - Does this need to be a status?
      Or are we just reporting the error and returning to *Inactive*

Monitoring method

- Systemd status
- Message bus events
    - Heartbeats
    - Direct queries
- Resource usage?
    - Do we need process ID's

Boot sequence

Starting Services

1. Supervisor Service will start via systemd. It will be enabled to run on boot. The assumption being that Mycroft devices using this service have no reason to not be running core on boot.
2. Immediately after the service starts, it will start all Mycroft Core services. The startup of these services will be monitored and the status of each updated as it starts.

3. Each Service will proceed to a Loaded state and await for the whole system to be Loaded. In the Loaded state, the Service should not respond to user queries.
4. When the Bus is ready, the Supervisor will report this to each Service.
5. In each Service, once the Service knows the Bus is available and the Service is Loaded. It reports Loaded to the Supervisor.
6. Once all Services are Loaded, then the Supervisor will report to each Service that it may switch to a state of Running.
7. A service should not interact with external requests (e.g. bus events) until it is in a running state

State Transitions

In all cases the MCS is responsible for setting the state of each Service.

Inactive > Starting

- The MCS starts each of the Services and sets the status accordingly

Starting > Loaded

- The MCS sends a message to each Service when the Bus is available
- Once the Bus is available and the Service has loaded, the Service tells the MCS that it is Loaded

When is each Service considered loaded?

- Skills
    - Default Skills and any installed Skills (except any blacklisted) have been loaded.
        - This means that the most common utterances can be handled - what time is it, who is ____, etc.
        - Intents for these should be cached, not generated on first boot.
    - Non-default Skills, or Skills requiring settings or other authentication may not be ready at this time.
        - Eg Spotify Skill may not be ready to play music but will report that it needs to sync settings first.
    - Skills may not be updated to the absolute latest version, but pre-loaded Skills should be able to handle requests.
- Audio
    - Sound can be output by the system
    - TTS is operational and dialog can be synthesized (on-device cache used prior to this)
- Speech
    - User can activate a device using the wake word selected in their device settings.

    - This should be using Precise shipped with the release (not downloaded on boot) but may fallback to PocketSphinx if necessary.
    - STT is operational and utterances can be transcribed
- Bus
    - Websocket server has been started and other services can connect.
    - Messages are being emitted and detected over the bus.
- Enclosure
    - Dependent on each platform
    - On Generic Enclosure - only that the Enclosure code is initialized and listening to Messages on the Bus.

Loaded > Running

- Once all Services are Loaded the MCS will update the state to Running and inform all Services.

Starting / Loaded / Running > Stopping

- The MCS may send a stop signal to the Service
- A Service may send notification to the MCS that it is stopping
- Can we detect SIGTERM / SIGKILL etc on other processes?
- Can we detect process shutdown via Python Multiprocessing?

Starting / Loaded / Running / Stopping > Crashed

- A Service will execute a callback from the MCS
- Can we detect process failure via Python Multiprocessing?

Stopping > Inactive

- Service will send a successful shutdown notification

Updates

Any image shipped should be able to run in that shipped state, fully loading before attempting to update.

Updates should only be performed once the system is in a Running state. This includes system packages, Skills, python packages, core updates, etc.

Heartbeat Process

The Supervisor service queries the running services once a second to determine if it is still running and healthy.

Watchdog Process

Implement a mechanism to check for the health of subsystems running within a service and act upon an unhealthy state. For example, if the microphone stops working, restart it.

Failure process

1. Send a notification of some sort regarding the symptoms of the failure if user is opted into the open dataset.
2. Attempt to restart process
3. Attempt to restart process via systemd
4. Ask enclosure to reboot
    a. Either directly or with a fail safe command:
        i. `mycroft.enclosure --reboot`

Implementation Phases

Phase 1

Coordinated start-up sequence only. No health monitoring.

Phase 2

Add command line options to selectively start/stop various Core Services

Phase 3

Add health monitoring and process restarting base on health

Phase 4

Move other watchdog processes into this Supervisor or Enclosure as appropriate