arm

BL31 LFA Design

John Powell and Manish Badarkhe

28th May 2026

Agenda

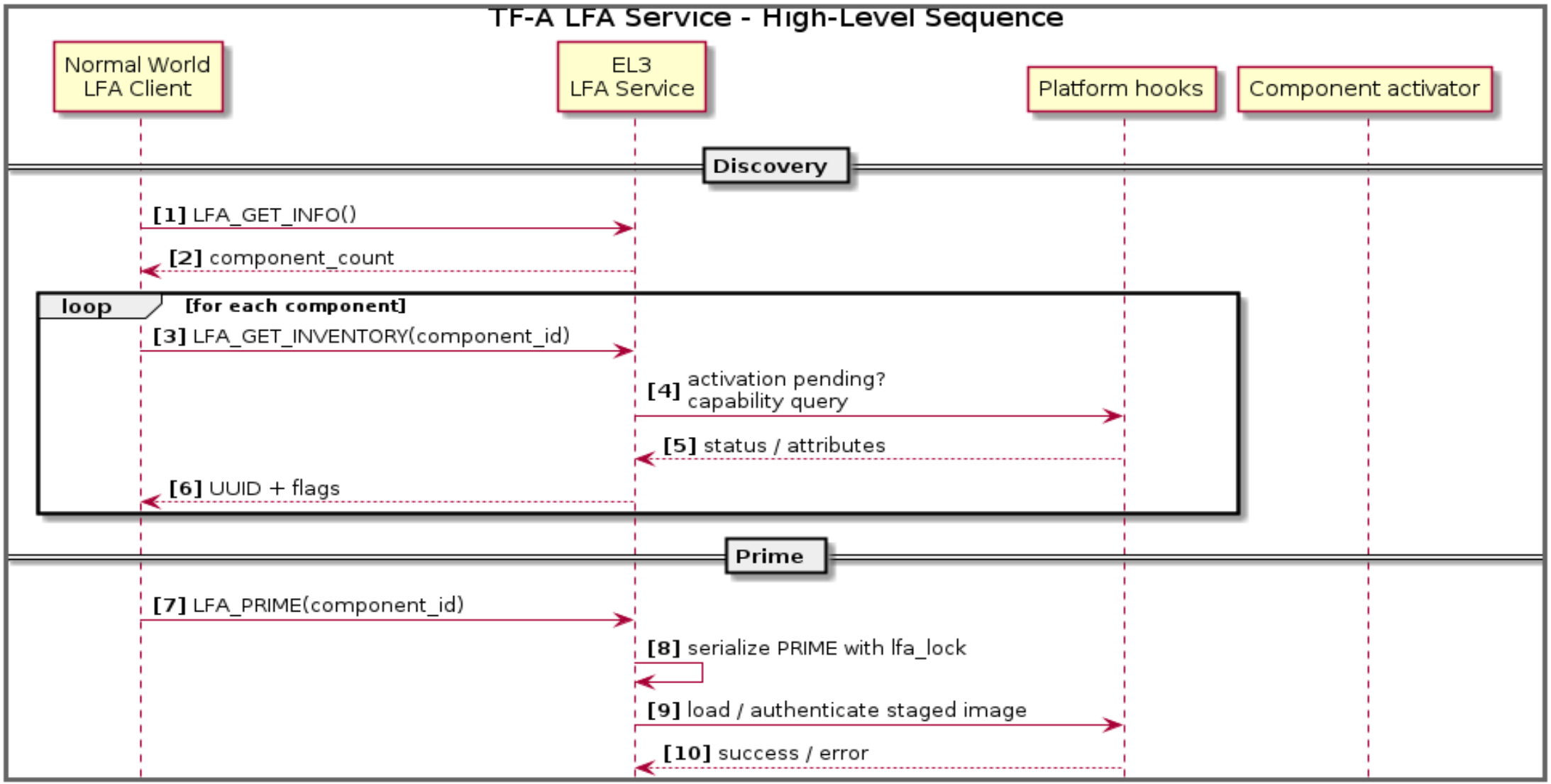
- Brief recap of the TF-A LFA service
- High-level service sequence
- Why serialization was needed
- Activation serialization
- Cancellation serialization
- Key takeaways and handoff to BL31-specific LFA

TF-A LFA Service Overview

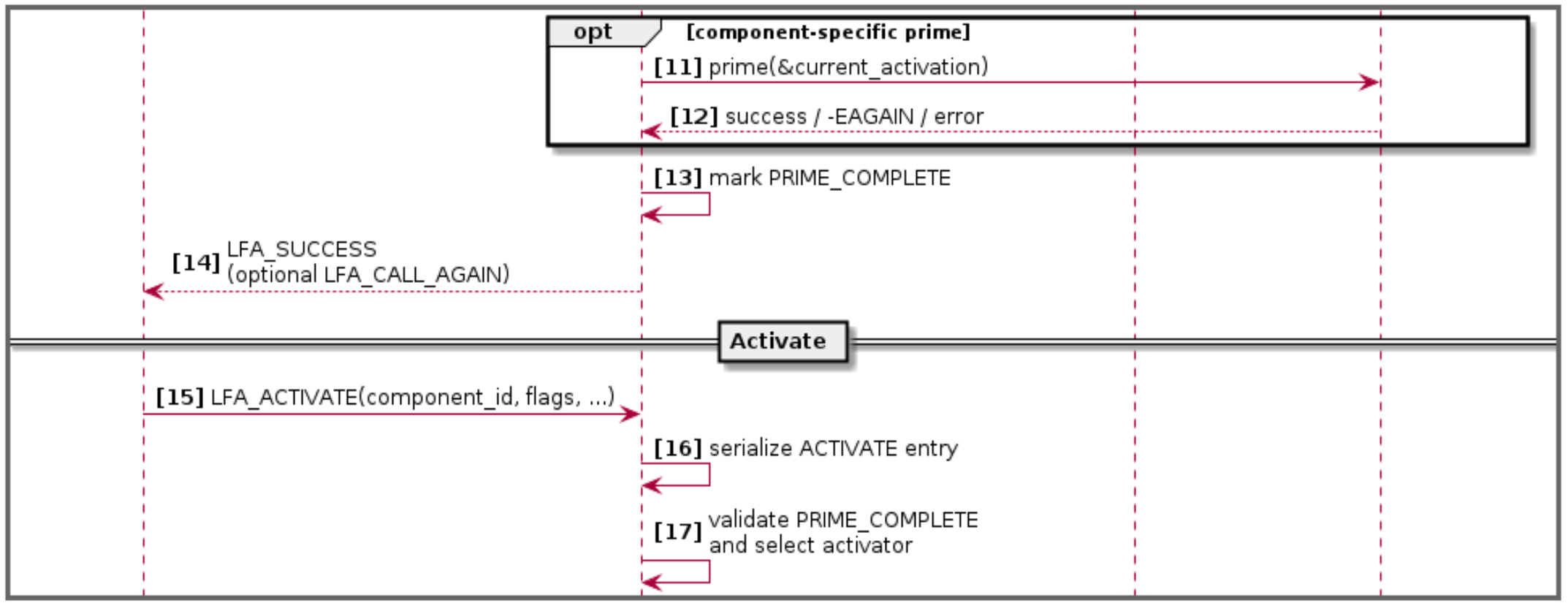
Generic service recap and recent serialization updates

- Overview
  - LFA is the EL3 runtime service in TF-A used to support live firmware activation.
  - It provides SMCCC calls for component discovery, preparation, activation, and cancellation.
  - The generic service owns the activation round state and validates the request flow.
- Scope
  - Component-specific behavior is delegated through activator callbacks.
  - This section focuses on the generic LFA service only.
  - BL31-specific activation internals are covered separately

LFA Service: High Level Sequence (1 / 4)

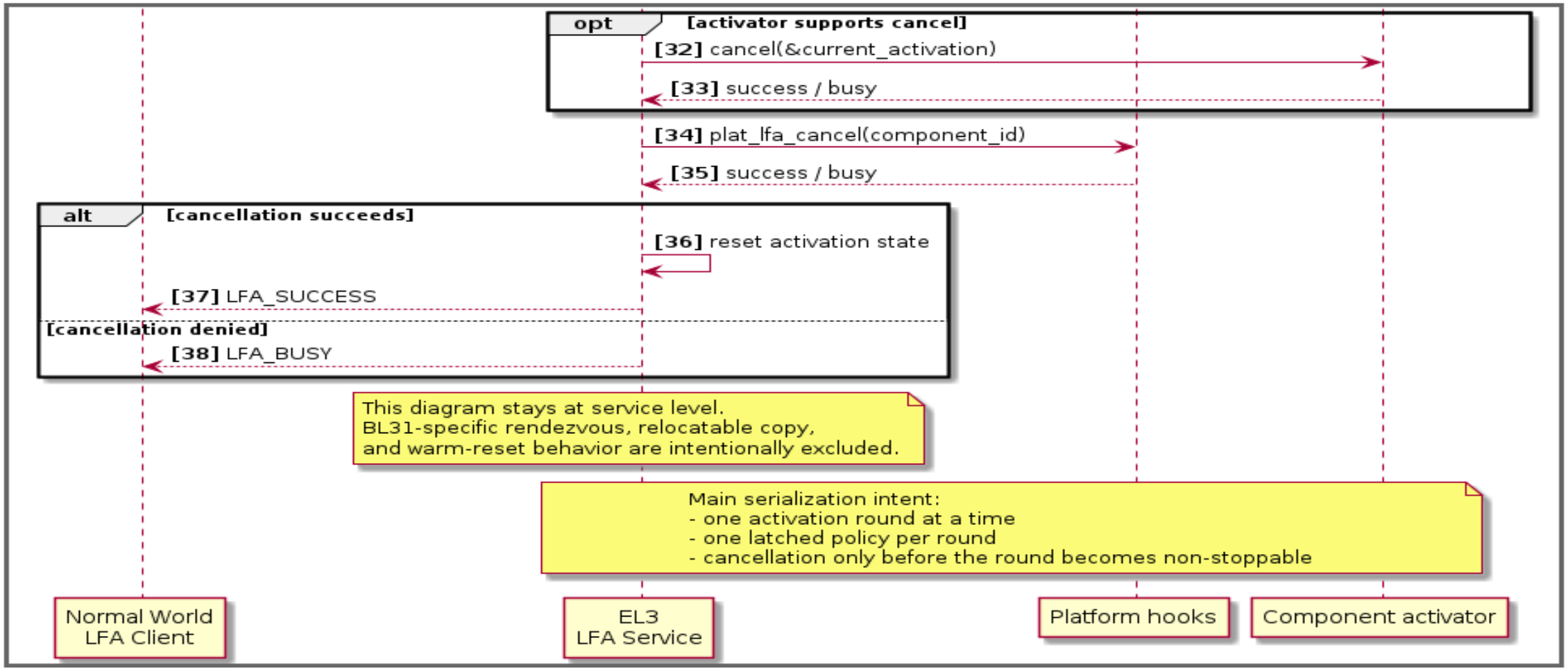


LFA Service: High Level Sequence (2 / 4)



LFA Service: High Level Sequence (3 / 4)

LFA Service: High Level Sequence (4 / 4)



Why Serialization Was Added

Shared EL3 state must be protected from concurrent SMCs

- Why it mattered
  - The LFA service uses shared activation state that is visible to all CPUs.
  - At the same time, SMC calls may arrive concurrently from different CPUs.
  - That creates race conditions between overlapping service operations.

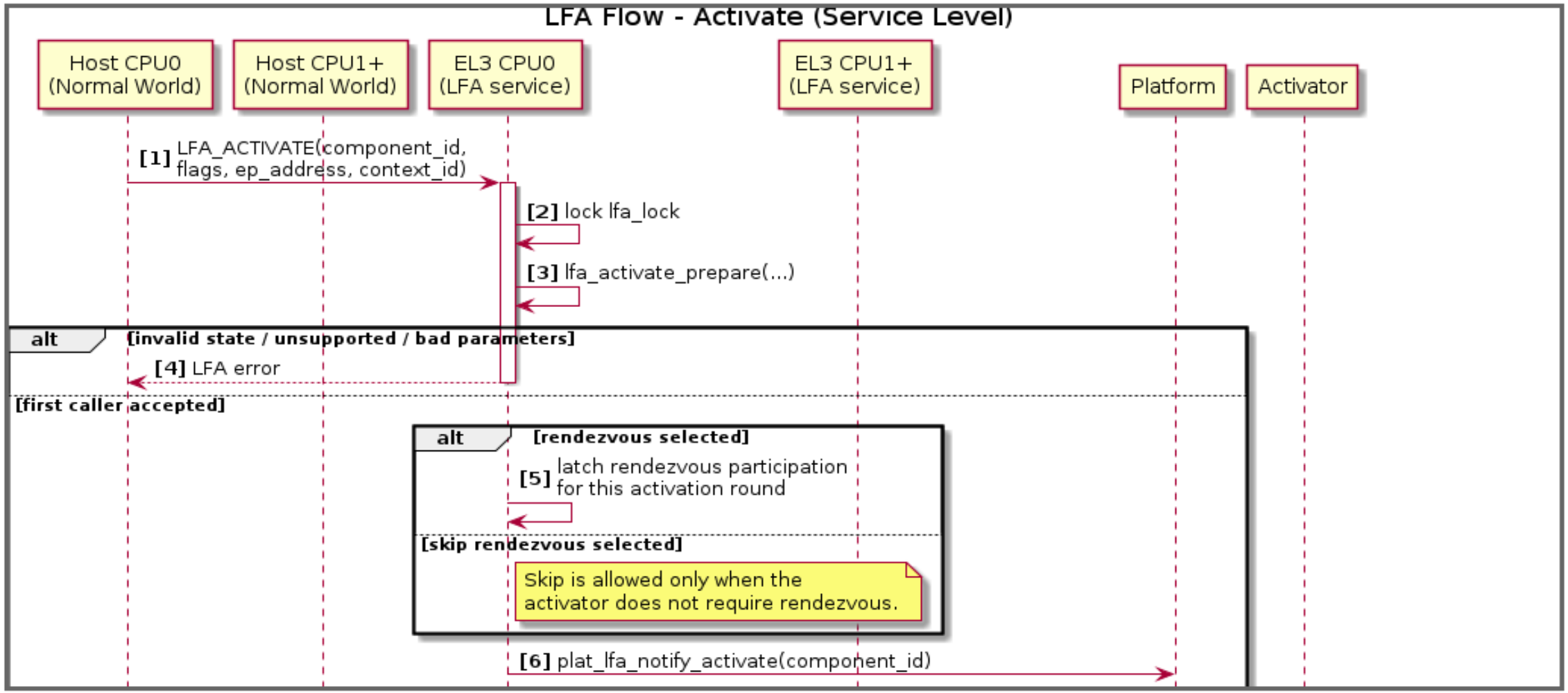
Activation Serialization

The first successful ACTIVATE caller defines the round

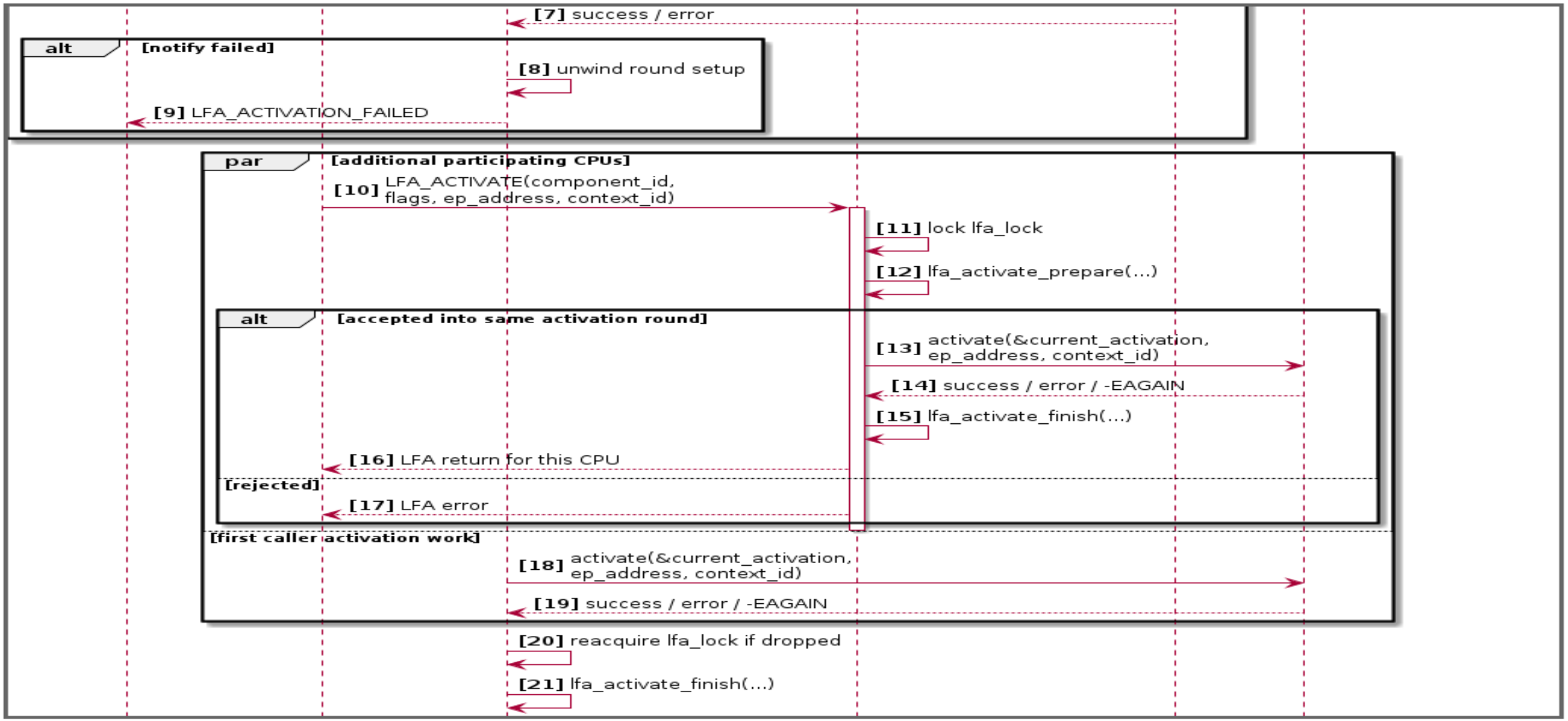
- What changes
  - Entry into the activation path is serialized using the generic LFA lock.
  - The first successful LFA_ACTIVATE caller performs round setup.
  - That caller validates that PRIME has completed successfully.

- Round behavior
  - The activator is selected once for the round
  - Activation policy is latched for the round.
  - Later callers either join the same round or are rejected.

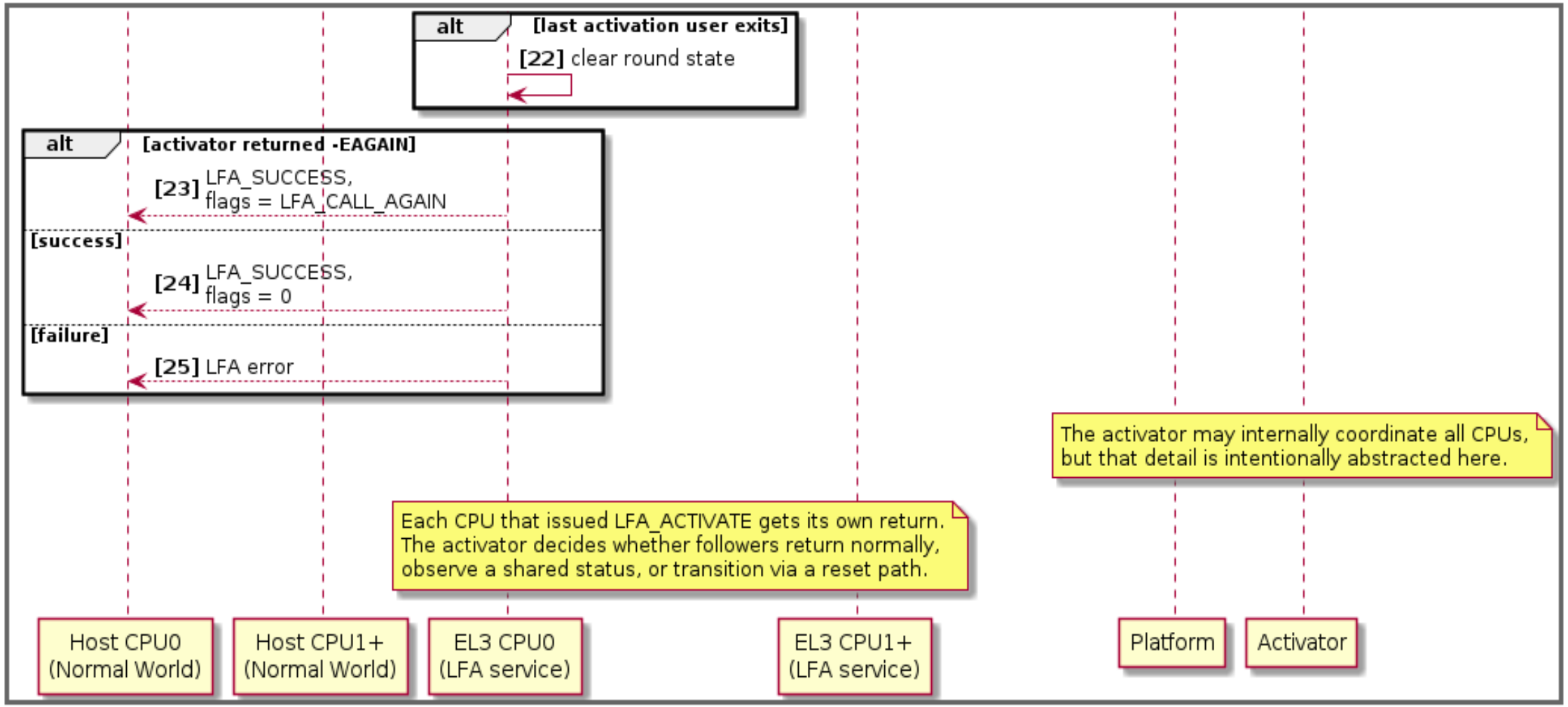
LFA Activate: Service Level (1 / 3)



LFA Activate: Service Level (2 / 3)



LFA Activate: Service Level (3 / 3)



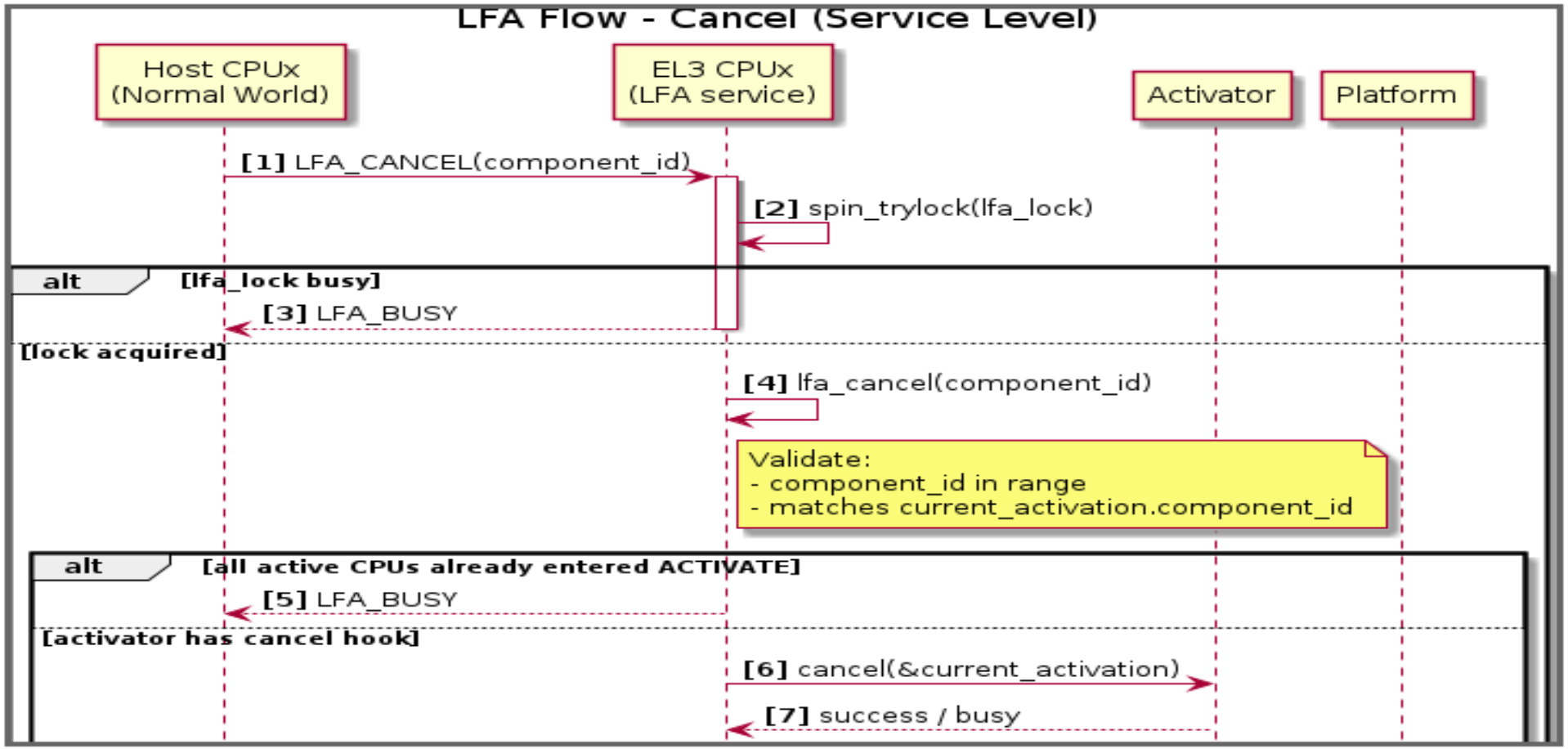
Cancellation Serialization

Cancellation is allowed only before activation becomes non-stoppable

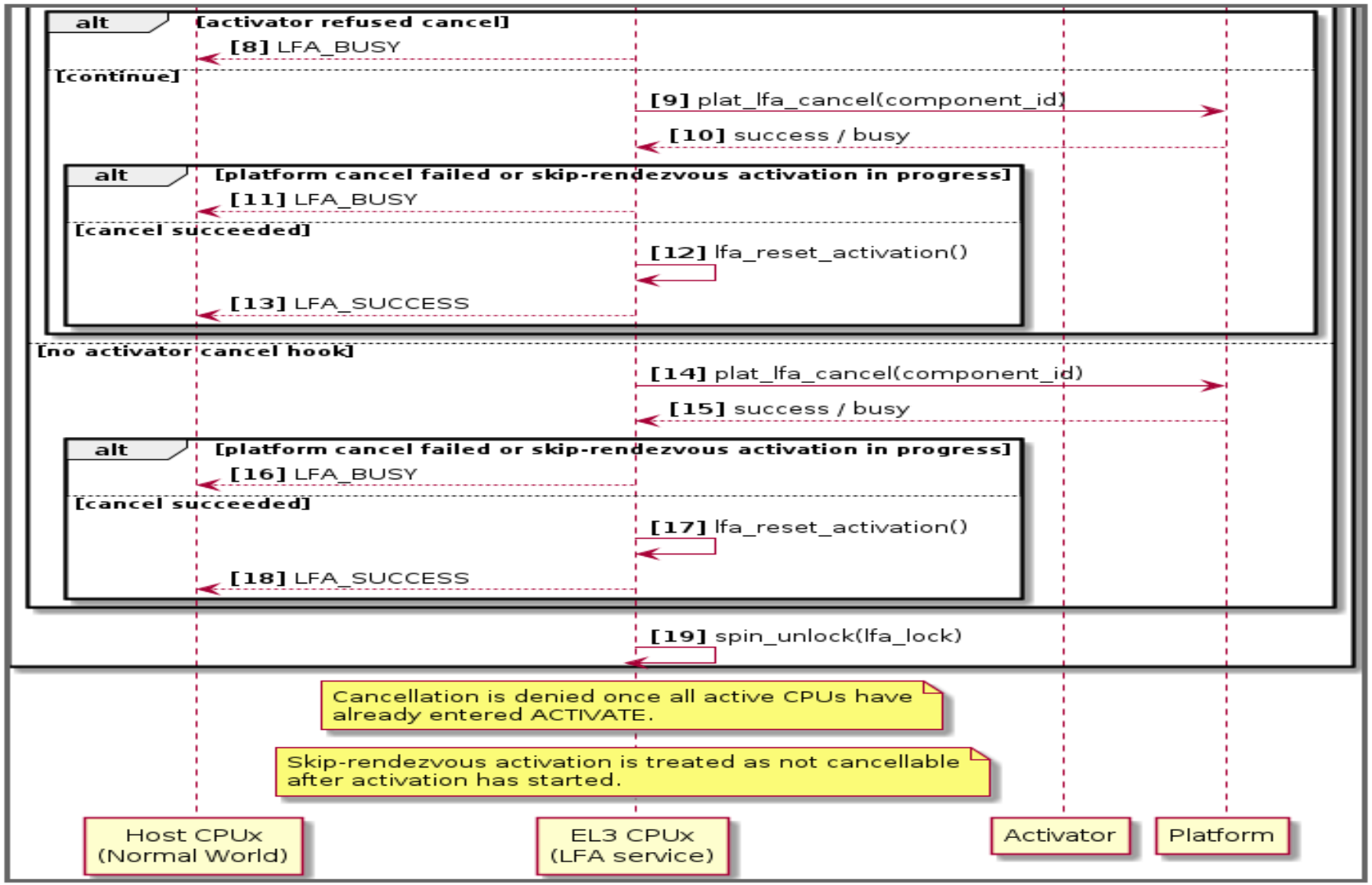
- Cancellation rules
  - LFA_CANCEL is serialized against the same shared activation state
  - The request must match the currently active component and round
  - Cancellation is accepted only while activation is still stoppable.

- Outcomes
  - Once the round progresses beyond that point, LFA_CANCEL returns LFA_BUSY
  - If cancellation succeeds, the generic LFA state is reset cleanly
  - This keeps cancellation inside the same controlled state machine as activation.

LFA Cancel: Service Level (1 / 2)



LFA Cancel: Service Level (2 / 2)



Key Takeaways And Handoff

Generic service behavior before the BL31-specific deep dive

- Takeaways
  - The generic LFA service is a shared EL3 state machine.
  - Recent work makes activation and cancellation explicitly serialized.
  - The first successful ACTIVATE caller owns round setup and round policy.
  - Cancellation is only valid before the round becomes committed.
- Next section
  - This improves determinism and makes the service behavior easier to reason about.
  - The next section covers the BL31-specific LFA activation path.

BL31 Live Activation

Overview

- Purpose is to replace running EL3 firmware without requiring a shutdown and full reboot.
  - Minimal system down time, this process is nearly invisible to running workloads.
  - Allows rapid deployment of CPU errata workarounds and small security fixes.
  - The current experimental implementation does have significant limitations.
    - We are somewhat at the mercy of the C compiler. When building images to live activate, the build environment should be the same as it was when the currently running image was built.
      - To attempt to work around this, changes are currently limited to assembly code changes which can be placed into their own section to avoid moving around other code.
    - EL3 should only manage context at the next highest exception level, so a hypervisor at EL2 would need to be aware of the activation process and save/restore registers at EL1, and so forth when a warm reboot is used after activation.

BL31 Live Activation

Platform Responsibilities

- Generic BL31 LFA library handles the actual activation of new images, but the platform has a responsibility to store, load, and authenticate new images.
  - plat_lfa_get_components
    - Returns list of LFA components supported by the platform.
  - is_plat_lfa_activation_pending
    - Checks whether a platform has a new image to activate for a given component.
  - plat_lfa_cancel
    - Let the platform know that an activation has been cancelled.
  - plat_lfa_load_auth_image
    - Load, authenticate, and measure a component for activation.
  - plat_lfa_get_image_info
    - Return information such as address and size of an image previously loaded and authenticated.
  - plat_lfa_notify_activate
    - Notifies the platform that an activation is starting.

BL31 Live Activation

High Level Activation Sequence

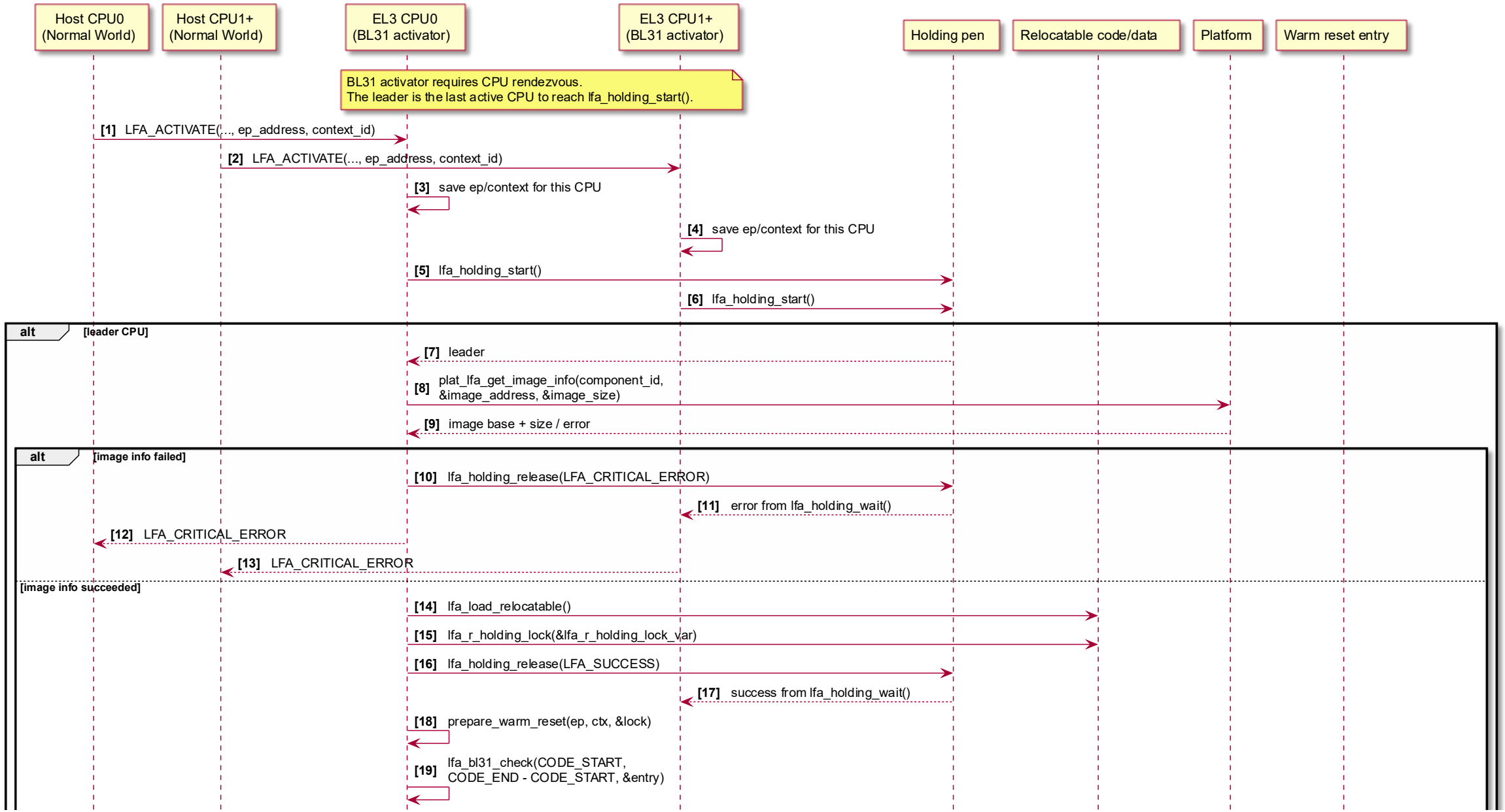
1. LFA_VERSION and LFA_FEATURES calls to ensure support/compatibility.
2. LFA_GET_INFO to query supported components.
3. LFA_GET_INVENTORY iteratively to find BL31 activation component and check that it is ready.
4. LFA_PRIME to ready the system for activation.
5. LFA_ACTIVATE
    1. If CPU rendezvous is needed, all online CPUs must call LFA_ACTIVATE to enter the holding pen for the duration of the activation process.
    2. The last CPU to call into LFA_ACTIVATE handles the bulk of the activation work
        1. Initializes a second relocatable holding pen for other CPUs to enter.
        2. Retrieves the new BL31 image via platform hook.
        3. Updates Xlat tables and copies the new BL31 image.

BL31 Live Activation

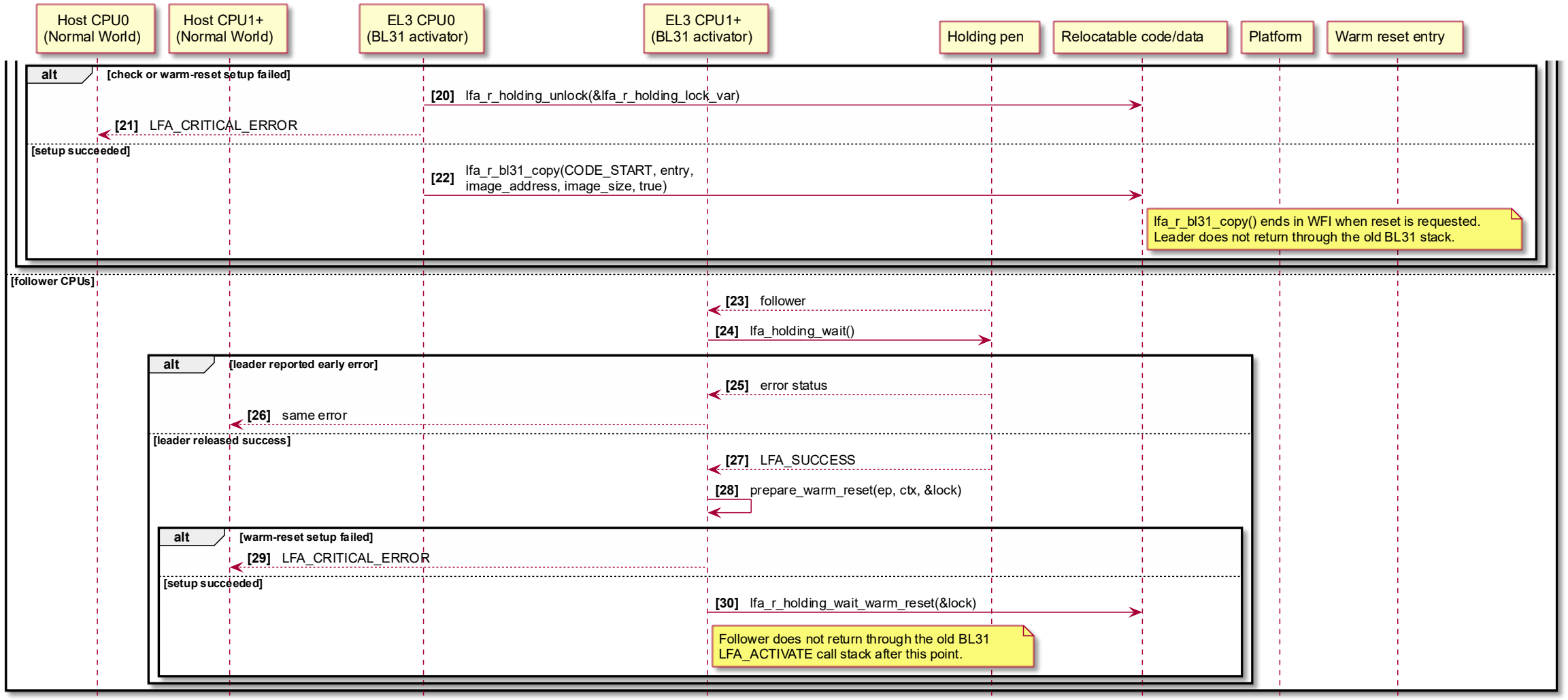
Holding Pens

- When CPU rendezvous is needed, there are 2 separate holding pens.
  - The first is where the first n-1 CPUs wait for the last CPU to call LFA_ACTIVATE
  - When CPU n calls LFA_ACTIVATE, it initializes the relocatable code section. A section of memory *outside* of the running BL31 image is initialized and a small block of code including a holding pen and copying function is copied into it.
  - CPU n then releases the first holding pen, and the first n-1 CPUs enter into the newly created relocatable holding pen where they wait for activation to complete.
  - CPU n enters the relocated copying function and safely overwrites code segments in the running BL31 image.
  - If a reset is requested, the CPUs will enter WFI afterwards then warm reboot and resume execution.

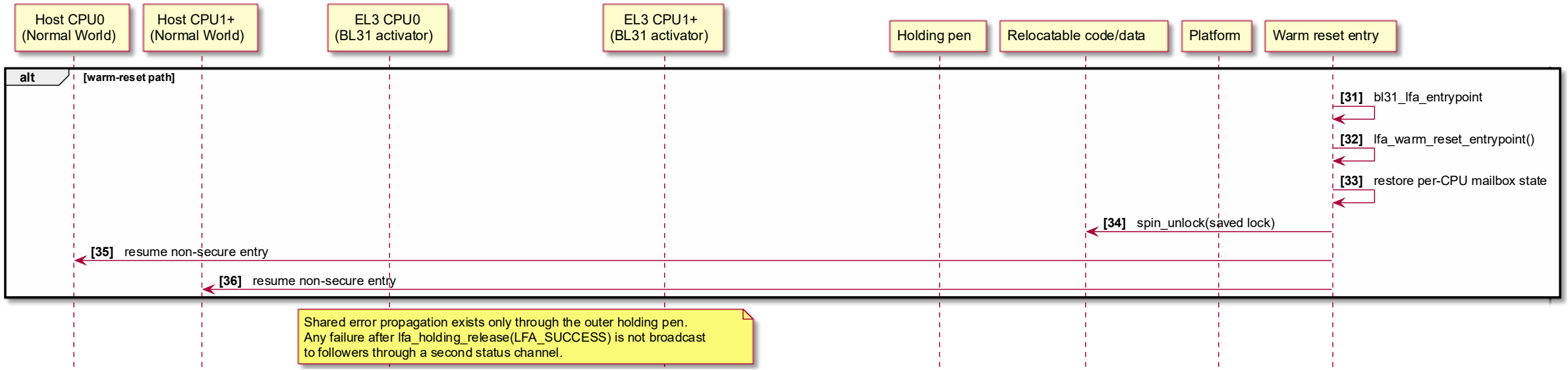
BL31 Live Activation Flow (reset = true)



BL31 Live Activation Flow (reset = true)



BL31 Live Activation Flow (reset = true)



BL31 Live Activation

Future Development

- TF-A automated CI integration of live firmware activation tests.
- Partner testing and feedback, use case discussions.
- Continued code updates as specification evolves.
- Live activation of partial images.
  - Not necessarily required to replace an entire image, cpu ops and errata sections should be enough for small changes.
- Code in review: https://review.trustedfirmware.org/q/topic:%22bl31_lfa_integration%22

BL31 Live Activation

FVP Demo

- Demo based on FVP platform with simple platform support.
  - New image loaded directly at fixed address and image size hard coded from a build parameter.
  - No verification, measurement, or authentication.
- Build BL31 and stash fip.bin and bl1.bin.
- Make changes in errata library and TF-A version number
- Build BL31 again
- Load the first fip.bin and bl1.bin as normal, and load the new bl31.bin into pre-determined address in FVP memory map
- Run TFTF to test live activation capabilities.

arm

Tack
ಧನ್ಯವಾದಗಳು
Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank you
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm