FF-A SP lifecycle

Document number	DEN0143
Document quality	ALP0
Document version	1.2
Document confidentiality	Non-confidential

FF-A SP lifecycle

Release information

Date	Version	Changes
2023/Dec/05	v1.2 ALP0	• Initial version based on FF-A base specification version 1.2 ALP0. • Introduction of new SP states NULL, Created, Starting, Stopping, Aborting, Stopped and Aborted. • Definition of the transitions for an SP starting, Stopping or Aborting. • Introduction of new framework messages to Start or Stop an SP. • Introduction of FFA_ABORT ABI to signal a fatal error.

Contents
FF-A SP lifecycle

FF-A SP lifecycle . . . ii
- Release information . . . ii
- Non-Confidential Proprietary Notice . . . iii

Preface

Additional reading . . . vi
Feedback . . . vii
- Feedback on this book . . . vii

Chapter 1 Overview

1.1 Scope . . . 10

Chapter 2 Concepts

2.1 Active SP . . . 11
2.2 Discoverable SP . . . 11
2.3 Resource Cleanup . . . 12
2.4 Fatal error . . . 12

Chapter 3 Lifecycle states

3.1 NULL state . . . 13
3.2 Created state . . . 14
3.3 Starting state . . . 14
3.4 Stopping state . . . 15
3.5 Stopped state . . . 15
3.6 Aborted state . . . 16

Chapter 4 Lifecycle transitions

4.1 Starting an SP . . . 18
4.2 Stopping an SP . . . 19
4.3 Aborting an SP . . . 21

Chapter 5 Lifecycle Setup

5.1 SP Lifecycle manifest field . . . 22
5.2 Abort manifest field . . . 23

Chapter 6 Lifecycle management interfaces

6.1 Component Start request . . . 25
6.2 Component Stop request . . . 25
6.3 Component start/stop response . . . 26
6.4 FFA_ABORT . . . 27

Chapter 7 Appendix

7.1 Live activation/update guidance . . . 29

Glossary

Preface

This document describes the FF-A SP Lifecycle and is a supplement of the FF-A specification [1] henceforth called the *Base specification*. This guidance has been separated into this document for the sake of brevity of the Base specification.

The guidance is treated as an extension of the Base specification. It is not versioned independently. Instead, it adopts that version of the Base specification in which changes to this guidance are released. It also fulfils the same compatibility requirements as the Base specification.

The guidance in this document can be at a different ALPHA quality level as compared to the Base specification. E.g. this document could be at ALP1 while the Base specification is at ALP3. To achieve alignment with the Base specification, this document must be at the BETA quality level for the Base specification to qualify for the same quality level. This approach allows this document to evolve somewhat independently of the Base specification w.r.t quality levels but also provides a point of alignment at the BETA quality level.

The reader is expected to use the guidance in this document in conjunction with the Base specification.

Additional reading

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Arm® Firmware Framework for Arm A-Profile Architecture*. See https://developer.arm.com/documentation/den0077/g

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an e-mail to errata@arm.com. Give:

- The title (FF-A SP lifecycle).
- The number (DEN0143 1.2).
- The page numbers to which your comments apply.
- The rule identifiers to which your comments apply, if applicable.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests PDFs only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the appearance or behavior of any document when viewed with any other PDF reader.

Chapter 1
Overview

I0001 The FF-A specification [1] defines runtime states and runtime models for an SP under the following assumptions:

- The SP is initialized during system boot.
- The SP remains operational until the next system shutdown or system reset.
- If the SP encounters a fatal error, this results in a system shutdown or system reset.

This supplement of the FF-A specification [1] extends the runtime states and runtime models to enable an SP to be stopped and started multiple times or to signal a fatal error.

I0002 This supplement extends the FF-A specification version 1.2.

I0003 The new mechanisms defined by this supplement enable new high level use cases:

- *Live activation*: A secure partition can be restarted to execute an updated version of its binary code.
- *Fatal Error handling*: A secure partition can inform the SPMC that it encounters an error that it cannot recover from and the SPMC can properly stop the faulting secure partition and eventually restart it.

I0004 This document is organized as follow:

- The Chapter 2 *Concepts* describes the concepts required by the SP lifecycle.
- The Chapter 3 *Lifecycle states* describes the states in addition to those described in the base specification.
- The Chapter 4 *Lifecycle transitions* describes the transitions between the states.
- The Chapter 5 *Lifecycle Setup* describes new manifest entries required to setup the SP lifecycle.
- The Chapter 6 *Lifecycle management interfaces* describes the interfaces and messages in addition to those described in the the base specification to support the SP lifecycle.

- The Chapter 7 *Appendix* contains examples of how the runtimes models can be used for specific use cases.

1.1 Scope

I_{0005} This version of the supplement of the FF-A specification considers only migrate-capable unicore SPs. Support for multi-core SPs or other types of endpoints is IMPLEMENTATION DEFINED and might be added in future versions.

X_{0006} For several transitions in this specification, defining how and when other vCPUs of the SP are also transitioning requires to carefully consider corner cases and will be handled in a future version of the specification.

I_{0007} The new states introduced by this supplement are defined for Secure partitions only and it is IMPLEMENTATION DEFINED to reuse them in an other scope.

I_{0008} When an SP is restarted following this specification, it is considered that it will start in exactly the same way as on its initial start during cold boot. Any way to optimize the restart by allowing the SP to save some data that would persist upon restart is IMPLEMENTATION DEFINED.

I_{0009} Informing a client of an SP that the SP has been stopped or restarted must be handled at this stage using IMPLEMENTATION DEFINED solutions.

Chapter 2
Concepts

2.1 Active SP

D0010 An SP is *active* if it is executing on a PE. Otherwise the SP is *inactive*.

R0011 In the states described in the FF-A specification [1], the SP is *active* in the *Running* state while it is *inactive* in the *Waiting*, *Blocked* and *Preempted* states.

2.2 Discoverable SP

D0012 An SP is *discoverable* if the presence of this SP can be determined through any of the following mechanisms:

- Via an invocation of an FF-A partition discovery mechanism.
- Via a successful invocation of any other FF-A ABI with the ID of this SP.

Otherwise, the SP is *non-discoverable* and both of the following are true:

- An invocation of an FF-A partition discovery mechanism does not report the presence of this SP.
- An invocation of any other ABI with the ID of this SP returns INVALID_PARAMETERS.

2.3 Resource Cleanup

D0013 When an SP is stopped, the actions to release, destroy or free in use resources are know as *cleanup actions*.

I0014 *Cleanup actions* can be performed by the SP and/or the SPMC.

I0015 *Cleanup actions* focus on the resources related to the FF-A specification [1]. The SP or the SPMC might have other IMPLEMENTATION DEFINED actions to perform for resources not related to FF-A.

D0016 *Cleanup actions* include, but are not limited to:

- Disable any interrupt assigned to the SP.
- Reclaim any memory shared or lent by the SP.
- Relinquish any memory shared or lent to the SP.
- Unmap RX/TX buffer pairs of the SP.
- Destroy any notification bitmaps of the SP.
- Complete or abort any pending transactions of the SP.
- Unmap all memory and peripherals from the SPs translation region.

2.4 Fatal error

D0017 An SP encounters a *Fatal error* when it is unable to continue its normal operations and any of the the following conditions are true:

- Clients of the SP are unable to access its services.
- The SP is unable to access services of others SPs.

I0018 An active SP detects that it has encountered a *Fatal error* and informs the SPMC by invoking the *FFA_ABORT* ABI (6.4 *FFA_ABORT*).

The SPMC detects that an SP has encountered a *Fatal error* e.g. upon triaging a synchronous or asynchronous exception that can be attributed to the SP or detecting an IMPLEMENTATION DEFINED error condition, for example an SP operation exceeding an IMPLEMENTATION DEFINED duration.

S0019 An active SP calls the *FFA_ABORT* ABI (6.4 *FFA_ABORT*) if it detects a *Fatal error*.

Chapter 3
Lifecycle states

I0020 This supplement defines new states in addition to the FF-A specification [1] and extends the rules and principles associated to them.

3.1 NULL state

D0021 The SPMC uses the *NULL* state as a placeholder for an SP before it starts initializing it or after it releases all resources associated to this SP.

R0022 An SP is in the *NULL* state when all of the following conditions are true:

- The SP is *inactive*.
- The SP has a partition manifest.
- The SPMC has not parsed the partition manifest to allocate any resources e.g. memory, interrupts, etc. for this SP.

R0023 An SP in the *NULL* state is *non-discoverable*.

R0024 The SPMC returns an *INVALID_PARAMETERS* error code to an endpoint contacting an SP in the *NULL* state.

3.2 Created state

D0025 The SPMC uses the *Created* state to allocate resources required to transition the SP to the *Starting* state.

R0026 An SP is in the *Created* state when all of the following conditions are true:

- The SP is *inactive*.
- The SPMC has allocated all resources required by the SP.
- The SP code is loaded in memory and ready to be executed.

R0027 An SP in the *Created* state is *discoverable*.

R0028 The SPMC returns a *BUSY* error code to an endpoint contacting an SP in the *Created* state and the endpoint should retry later.

3.3 Starting state

D0029 An SP uses the *Starting* state to perform IMPLEMENTATION DEFINED actions before it can call *FFA_MSG_WAIT* and transitions to the *Waiting* state to handle other endpoints requests.

R0030 An SP is in the *Starting* state when all of the following conditions are true:

- The SP is *active* and its last state was the *Created* state.
- The SP has not entered the *Waiting* state.

R0031 An SP in the *Starting* state is *discoverable*.

R0032 The SPMC returns a *BUSY* error code to an endpoint contacting an SP in the *Starting* state and the endpoint should retry later.

I0033 The *Starting* state is the equivalent of the *Running* state when the partition is initializing in the FF-A specification [1].

D0034 The FF-A specification instructs to use the *FFA_ERROR* ABI if an error occurs during the initialization however the *FFA_ABORT* ABI (6.4 *FFA_ABORT*) should be used instead when it is supported.

X0035 If the SP cannot transition to the *Waiting* state, the SPMC performs the required cleanup actions and transitions the SP to the *Stopped* state.

R0036 An SP in the *Starting* state cannot use the following ABIs:

- FFA_YIELD.
- Direct Response ABIs.
- FFA_RUN.

3.4 Stopping state

D0037 An SP uses the *Stopping* state to perform IMPLEMENTATION DEFINED actions and required SP cleanup actions (2.3 *Resource Cleanup*) before it acknowledges the stop request and transitions to the *Stopped* state.

R0038 An SP is in the *Stopping* state when all of the following conditions are true:

- The SP is *active*.
- The SP received a Component stop request (6.2 *Component Stop request*).

R0039 An SP in the *Stopping* state is *discoverable*.

R0040 The SPMC returns a *BUSY* error code to an endpoint contacting an SP in the *Stopping* state and the endpoint should retry later.

R0041 An SP in the *Stopping* state cannot use the following ABIs:

- FFA_YIELD.
- Direct Response ABIs.
- FFA_RUN.

3.5 Stopped state

D0042 The SPMC keeps the SP in *Stopped* state after it has done its own internal cleanups for the SP until it transitions the SP to the *Created* or the *NULL* state.

R0043 An SP is in the *Stopped* state when all of the following conditions are true:

- The SP is *inactive*.
- The SPMC has system resources allocated to the SP.
- The SP was previously active.
- All cleanup actions required have been performed.

R0044 An SP in the *Stopped* state is *discoverable*.

R0045 The SPMC returns a *BUSY* error code to an endpoint contacting an SP in the *Stopped* state and the endpoint should retry later.

3.6 Aborted state

D0046 The SPMC uses the *Aborted* state to handle cleanup actions required (See 2.3 *Resource Cleanup*) before it transitions the SP in the *Stopped* state.

R0047 An SP is in the *Aborted* state when all of the following conditions are true:

- The SP is *inactive*.
- The SP encountered a fatal error (2.4 *Fatal error*).

R0048 An SP in the *Aborted* state is *discoverable*.

R0049 The SPMC returns a *BUSY* error code to an endpoint contacting an SP in the *Aborted* state and the endpoint should retry later.

Chapter 4
Lifecycle transitions

I0050 This chapters describes the transitions between the various SP lifecycle states in the following scenarios:

- An SP is started. See 4.1 *Starting an SP*.
- An SP is stopped. See 4.2 *Stopping an SP*.
- An SP is aborted. See 4.3 *Aborting an SP*.

4.1 Starting an SP

R0051 Figure 4.1 shows the state transitions that apply when an SP is started.

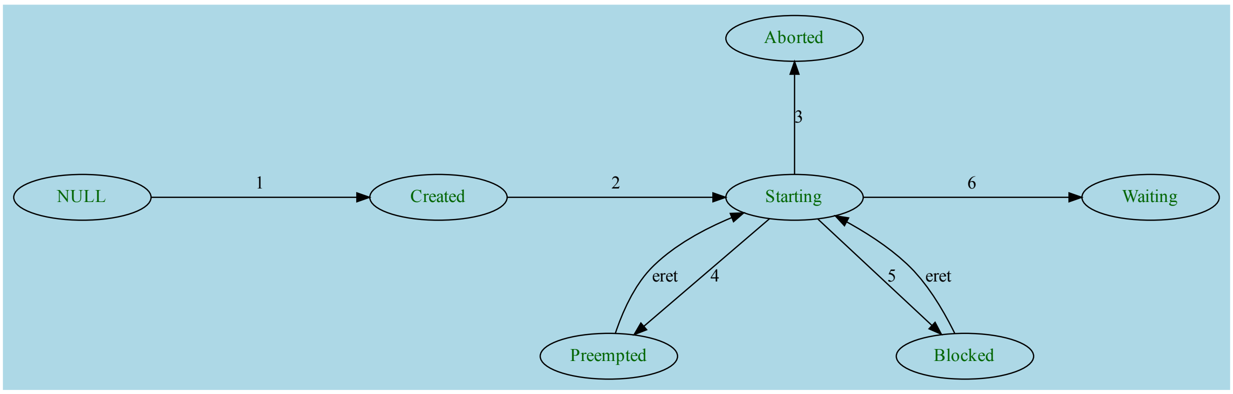

Figure 4.1: State machine for starting an SP

The numbered transitions in Figure 4.1 are described below:

1. **NULL state to Created state**
   - An SP undergoes this transition when it is ready to be executed after its resources have been allocated and initialized by the SPMC.
2. **Created state to Starting state**
   - An SP undergoes this transition when it starts executing.
3. **Starting state to Aborted state**
   - An SP undergoes this transition upon encountering a fatal error. See 2.4 *Fatal error*.
4. **Starting state to Preempted state**
   - An SP undergoes this transition upon being interrupted. Also see [1].
   - The SP transitions back to the Starting state from the Preempted state via the `ERET` instruction.
5. **Starting state to Blocked state**
   - An SP undergoes this transition when it waits for some work to complete on its behalf. Also see [1].
   - The SP transitions back to the Starting state from the Blocked state via the `ERET` instruction.
6. **Starting state to Waiting state**
   - An SP transitions from the *Starting* to the *Waiting* state when it calls the FFA_MSG_WAIT ABI. Also see [1].

4.2 Stopping an SP

R0052 Figure 4.2 shows the state transitions that apply when an SP is stopped.

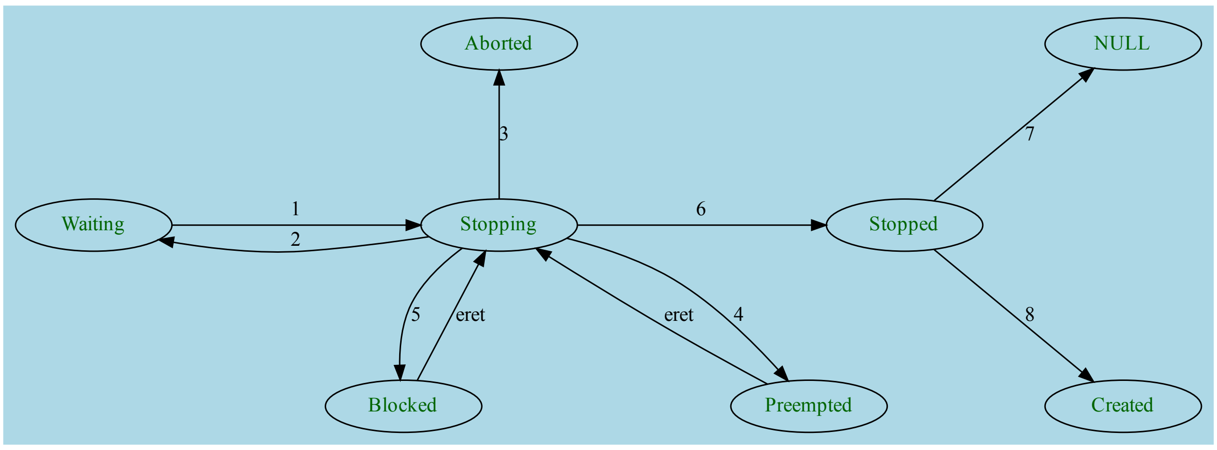

Figure 4.2: State machine for stopping an SP

The numbered transitions in Figure 4.2 are described below:

1. **Waiting state to Stopping state**
   - An SP undergoes this transition when it receives a *Component stop request*. See *6.2 Component Stop request*.
2. **Stopping state to Waiting state**
   - An SP undergoes this transition when it is unable to transition to the Stopped state and sends a *Component start/stop response* with a non-zero status code back to the SPMC. See 6.3 *Component start/stop response*.
3. **Stopping state to Aborted state**
   - An SP undergoes this transition upon encountering a fatal error. See 2.4 *Fatal error*.
4. **Stopping state to Preempted state**
   - An SP undergoes this transition upon being interrupted. Also see [1].
   - The SP transitions back to the Stopping state from the Preempted state via the `ERET` instruction.
5. **Stopping state to Blocked state**
   - An SP undergoes this transition when it waits for some work to complete on its behalf. Also see [1].
   - The SP transitions back to the Stopping state from the Blocked state via the `ERET` instruction.
6. **Stopping state to Stopped state**
   - An SP undergoes this transition when it sends a *Component start/stop response* with a *SUCCESS* status code to the SPMC. See 6.3 *Component start/stop response*.
7. **Stopped state to NULL state**
   - An SP undergoes this transition when its resources have been de-allocated by the SPMC.

8. **Stopped state to Created state**
    - An SP undergoes this transition when it is ready to be executed after its resources have been reinitialized by the SPMC.

4.3 Aborting an SP

R0053 Figure 4.3 shows the state transitions that apply when an SP aborts.

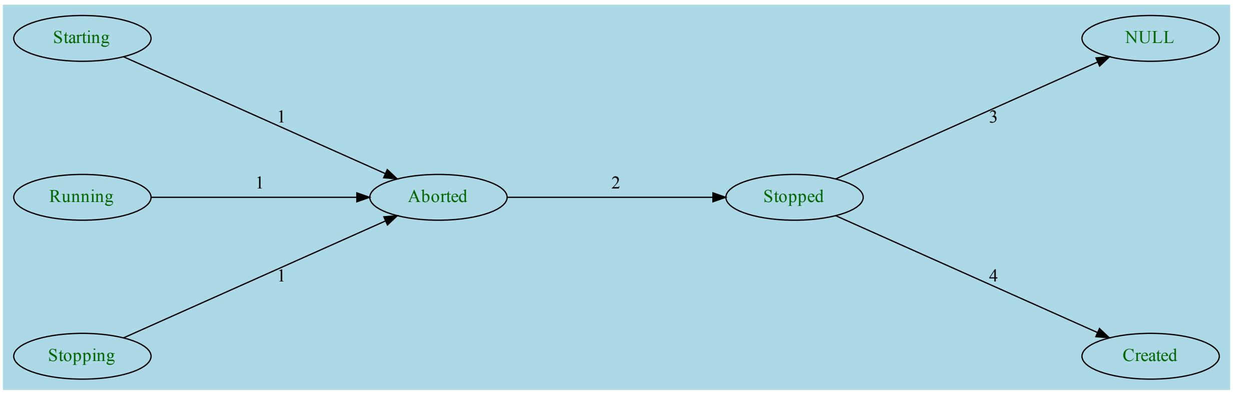

Figure 4.3: State machine for aborting an SP

1. **Any Active SP state to Aborted state**
    - An Active SP (in the Starting, Stopping or Running state) undergoes this transition upon encountering a fatal error. See 2.4 *Fatal error*.
2. **Aborted state to Stopped state**
    - An SP undergoes this transition when its resources have been uninitialized by the SPMC. See 2.3 *Resource Cleanup*.
3. **Stopped state to NULL state**
    - An SP undergoes this transition when its resources have been de-allocated by the SPMC.
4. **Stopped state to Created state**
    - An SP undergoes this transition when it is ready to be executed after its resources have been reinitialized by the SPMC.

Chapter 5
Lifecycle Setup

5.1 SP Lifecycle manifest field

R_{0054} The SPMC discovers whether an SP supports the complete SP lifecycle via its partition manifest.

I_{0055} An SP not supporting the SP lifecycle can use the *FFA_ABORT* to signal a critical error but it cannot be restarted.

Table 5.1: Partition properties

Information fields	Mandatory	Description
SP Lifecycle Support	No	• Presence of this field indicates that the partition can use the SP lifecycle described in this specification and supports to be stopped. • A partition not having this field might still use the abort ABI if it is supported by the SPMC.

5.2 Abort manifest field

R0056 The action to be performed when a partition is aborting can be defined in the manifest using the *Abort action* field described in Table 5.2.

Table 5.2: Partition properties

Information fields	Mandatory	Description
Abort action	No	• This field specifies the action to be done by the SPMC if the partition encounters a fatal error. – *Stop*. The SPMC should keep the SP in the Stopped state. – *Destroy*. The SPMC should transition the SP to the NULL state. – *Restart*. The SPMC should transition the SP to *Starting* state (via the *Stopped* and *Created* states). – *Propagate*. The SPMC shall abort itself to the SPMD. – IMPLEMENTATION DEFINED. An IMPLEMENTATION DEFINED action is taken by the SPMC. • In the absence of this field in the manifest, the action done by the SPMC is IMPLEMENTATION DEFINED.

Chapter 6
Lifecycle management interfaces

R0057 Lifecycle requests are sent using framework message through the FFA_MSG_SEND_DIRECT_REQ ABI.

R0058 Lifecycle requests are acknowledged through the *Component start/stop response* message.

D0059 The message to signal a fatal error has its own ABI (see 6.4 *FFA_ABORT*).

6.1 Component Start request

R0060 Component start requests can only be sent by the SPMD to the SPMC.

I0061 A start request requires the endpoint ID to be started. The ID can be found using discovery mechanisms which are listing endpoints which are not in the NULL state. Consequently, the start request can only be used for endpoints which where started on boot.

Table 6.1: Message to request to start an endpoint

Register	Parameter
w0	FFA_MSG_SEND_DIRECT_REQ Function ID (0x8400006F or 0xC400006F).
w1	• Sender and Receiver endpoint IDs. – Bit[31:16]: SPMD ID. – Bit[15:0]: SPMC ID.
w2	• Message flags. – Bit[31] = b'1: Framework message. – Bit[30:8] = 0: Reserved (MBZ). – Bit[7:0] = b'00001000: Message to request the SPMC to start an endpoint.
w3	Endpoint ID to be started.
w4-w7	Reserved (MBZ).

6.2 Component Stop request

Table 6.2: Message to request to stop an endpoint

Register	Parameter
w0	FFA_MSG_SEND_DIRECT_REQ Function ID (0x8400006F or 0xC400006F).
w1	• Sender and Receiver endpoint IDs. – Bit[31:16]: SPMD ID or SPMC ID. – Bit[15:0]: SPMC ID or endpoint ID.
w2	• Message flags. – Bit[31] = b'1: Framework message. – Bit[30:8] = 0: Reserved (MBZ). – Bit[7:0] = b'00001001: Message to request an endpoint to stop.
w3	• Component ID to be stopped. – Endpoint ID or SPMC ID if the SPMD is requesting the SPMC to stop. • MBZ for requests by the SPMC to an endpoint.
w4-w7	Reserved (MBZ).

6.3 Component start/stop response

I0062 *Component start/stop responses* are used by endpoints to acknowledge Lifecycle start or stop requests.

R0063 *Component start/stop responses* non success error codes should only be used to signal back errors related to the request.

X0064 If a fatal error occurs while the SP is in the *Stopping* state, the SP is not able to handle requests anymore so it cannot transition back to the *Waiting* state. In this case the *FFA_ABORT* ABI should be used instead of a response message.

Table 6.3: Lifecycle request response message encoding

Register	Parameter
w0	0x84000070: FFA_MSG_SEND_DIRECT_RESP Function ID
w1	• Sender and Receiver endpoint IDs. – Bit[31:16]: SPMC ID or SP ID. – Bit[15:0]: SPMD ID or SPMC ID.
w2	• Message flags. – Bit[31] = b'1: Framework message. – Bit[30:8] = 0: Reserved (MBZ). – Bit[7:0] = b'00001010: Message to acknowledge a Component start or stop request.
w3	• Lifecycle request status code. – 0: SUCCESS. ∗ The endpoint acknowledges the successful handling of the Component start or stop request. – -1: NOT_SUPPORTED. ∗ The targeted endpoint or the SPMC does not support the requested operation. ∗ The targeted endpoint remains in its current state. – -2: INVALID_PARAMETERS. ∗ One or more parameters contain invalid values. ∗ The targeted endpoint remains in its current state. – -6: DENIED. ∗ The endpoint cannot acknowledge the successful handling of the request due to an IMPLEMENTATION DEFINED reason. ∗ The targeted endpoint remains in its current state. – -7: RETRY. ∗ The targeted endpoint is in a state that prevents it from handling the request. The requester should resend the message. ∗ The targeted endpoint remains in its current state.
w4-w7	Reserved (SBZ).

6.4 FFA_ABORT

Description
• This ABI is invoked by an endpoint to stop its partition after encountering a fatal error (2.4 *Fatal error*). • Valid FF-A instances and conduits are listed in Table 6.5. • Syntax of this function is described in Table 6.6. • This function shall never return.

Table 6.5: FFA_ABORT instances and conduits

Config No.	FF-A instance	Valid conduits
1	Secure physical	SMC
2	Secure virtual	SMC,HVC,SVC

Table 6.6: FFA_ABORT function syntax

Parameter	Register	Value
uint32 Function ID	w0	• 0x84000090. • 0xC4000090.
First parameter register	w1/x1	• Reserved (SBZ)
uint64 IMPLEMENTATION DEFINED	w2/x2	• IMPLEMENTATION DEFINED value.
Other Parameter registers	w3-w7 x3-x17	• Reserved (SBZ)

Chapter 7
Appendix

7.1 Live activation/update guidance

I0065 This chapter provides an example of how a live activation or update of one or several SPs can be done using some of the concepts introduced in this supplement.

D0066 The FF-A SP Lifecycle supplement provides the ABIs and runtime model required to restart SPs but does not cover update specific aspects such as loading, updating or authenticating binaries.

I0067 Figure 7.1 provides an overview of the different steps possible to follow to perform a live restart of one or several SPs.

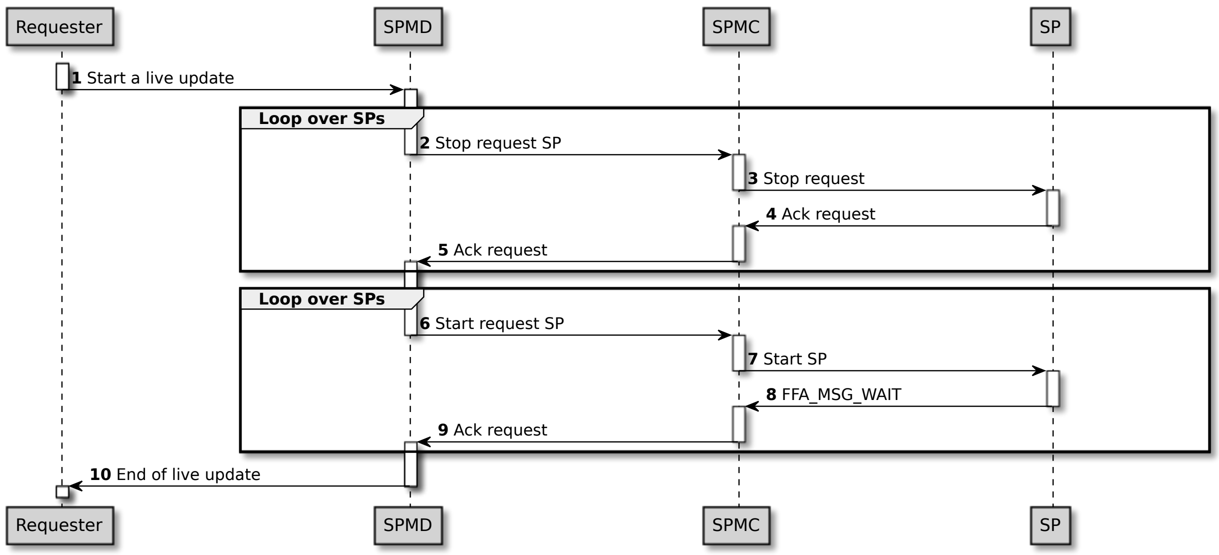

Figure 7.1: Example live restart flow

U0068 A live update of a group of SPs can be done through the following steps on a running system:

1. SPMD is requested to activate the updated version of a list of SPs using an IMPLEMENTATION DEFINED protocol.

- For each of the SPs in the list:
  2. The SPMD sends a Component Stop request to the SPMC for one SP.
  3. The SPMC sends a Component Stop request to the SP.
  4. The SP performs the cleanup actions required and sends back a Lifecycle request response.
  5. The SPMC sends a Component start/stop response to the SPMD.
- For each of the SPs in the list:
  6. The SPMD sends a Component Start request to the SPMC for one SP.
  7. The SPMC starts the requested SP.
  8. The SP initializes and uses the FFA_MSG_WAIT ABI to enter the *Waiting* state.
  9. The SPMC sends a Component start/stop response to the SPMD.

10. The SPMD can acknowledge that the update was done successfully.

U0069 Prior, during and after this flow, intermediates steps might be required to perform loading, authentication or validation of the update but those are IMPLEMENTATION DEFINED and out of the FF-A specification scope.

U0070 If the SPMD wants the SPMC to be stopped or started, it shall give the SPMC endpoint ID in the lifecycle requests.

Glossary

ABI

Application Binary Interface.

FF-A

Firmware Framework for A-profile.

HVC

Hypervisor Call.

MBZ

Must Be Zero.

PE

Processing Element.

SBZ

Should Be Zero.

SMC

Secure Monitor Call.

SP

Secure Partition.

SPM

Secure Partition Manager.

SPMC

Secure Partition Manager Core.

SPMD

Secure Partition Manager Dispatcher.

SVC

Supervisor Call.