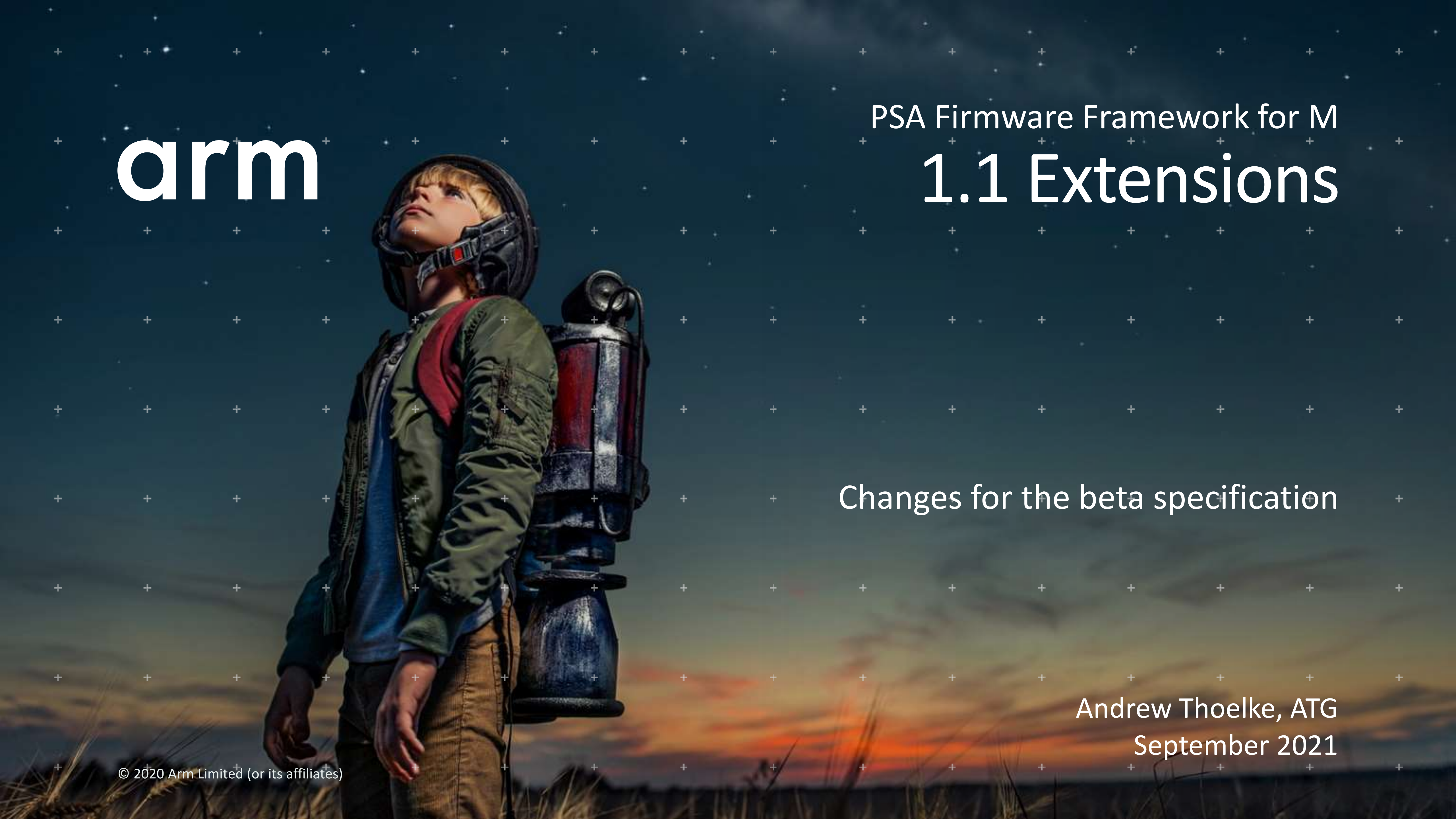

PSA Firmware Framework for M

1.1 Extensions

Changes for the beta specification

Andrew Thoelke, ATG

September 2021

PSA Firmware Framework for M – 1.1 Extensions (beta)

Content

- Status
- Objectives
- Overview
- Change details
- Next steps

Status of the specification

- The version 1.1 Extensions **alpha** specification was published in January 2021
  - See https://developer.arm.com/documentation/aes0039/a
  - Summary at https://www.trustedfirmware.org/docs/PSA-FF-M_1.1_Extensions-overview.pdf
- TF-M has been implementing this during 2021
  - This has provided feedback on the proposed changes
  - And additional feedback on the version 1.0 specification

- We are preparing a beta version of the 1.1 Extensions specification
  - This reflects increased confidence in the proposed set of changes
  - This still enables us to take community feedback before finalising version 1.1
- Integrating the 1.1 Extensions into the version 1.0 specification is complex
  - We will wait until the **release** version of the specification to make those changes
  - The beta is a specification of the **changes** between version 1.0 and 1.1
- The 1.1 Extensions **beta** specification should be available in a few weeks

Objectives for version 1.1

- A framework specification that can target a smaller, simpler system
  - There should be continuity between any new lighter-weight framework and FF-M version 1.0, enabling the same RoT Service code to be used with different types of framework
- Provide optimized mechanisms for some common Secure Partition RoT Service development patterns
  - Improving efficiency for these use cases requires a reduction in flexibility or security mitigation
- Improve the support for secure peripheral drivers – the APIs in version 1.0 are inadequate for many systems:
  - The signal-based mechanism makes it very difficult to handle interrupts in a bounded time
  - The FF-M 1.0 API assumes that the peripheral MMIO interface is sufficient to control its interrupts
- Maintain compatibility for RoT Service code written for version 1.0
  - Make migration to version 1.1 frameworks, and adoption of 1.1 features easy

Summary of the changes between ALPHA and BETA

- Changed the interaction of FLIH functions with the SPM
- Simplified the API for managing interrupt state
- Permit an implementation to *only* support stateless RoT Services

- Deprecated the Secure Partition doorbell feature
- Clarified the reserved ranges of status codes in the API definition
- Introduced an implementation-defined limit for the value of the message type in calls to psa_call()
- Clarified and expanded the description of the optional isolation rules
  - Encourage policy implementation to be based on threat models and security analyses

Interrupt handling and management

- Allowing an FLIH function to call the SPM is complex in some implementations, due to the context switch and caller validation required
- In the beta specification:
  - The FLIH function return value is extended to support the two primary use cases for calling the SPM
    - Disabling the current interrupt (to enable hand-off to the thread context)
    - Panicking the system on detection of an unrecoverable fault
  - The availability of any of the Secure Partition API from FLIH context is now Implementation defined

- The proposed interrupt management API could not be implemented and used in a way that was guaranteed to be free of race-conditions
  - Use within safety-critical systems might be problematic
- In the beta specification, this API is simplified to just being able to enable or disable a single interrupt, with no ability to query the current state
  - If the current state is needed by the Secure Partition, it can track the state itself

Increased implementation flexibility for optimisation

- In the alpha specification, an implementation was required to support both stateless and connection-based RoT Services
- In the beta specification, support for connection-based RoT Services is optional
  - This enables the simplest implementations to require no dynamic memory allocation within the SPM

- In version 1.0, the message type in psa_call() could be any non-negative 32-bit integer
- In version 1.1, the maximum supported value of the message type is Implementation defined
  - It must be at least 32767, to provide a guaranteed range that is portable between implementations

Improve the understanding of ‘optional isolation rules’

- In version 1.0, the *Optional isolation rules* section does not meet the original intention
  - These are meant to be **examples** of additional policies
  - Implementations are permitted to implement other isolation policies
  - Rule **I4** and **I5** imply that the implementation cannot use shared code libraries
- None of these rules are mandatory, under any specified isolation level
  - This reflects the diversity of hardware systems which can support a compliant implementation, and the variation in required security level for end products.
- In version 1.1, these rules are presented as example policies
  - Implementations are encouraged to select isolation policies based on threat modelling and security analysis
  - The description and rationale for these policies are moved to a new appendix
  - Each one is more clearly linked as a mitigation response to one or more attack techniques

Deprecated features

- The doorbell feature was provided in version 1.0 to help in the development of complex peripheral drivers
- The main uses cases that inspired this feature are better supported with the use of FLIH functions in an implementation of version 1.1
- From version 1.1, the Secure Partition doorbell feature is deprecated
  - Implementations of the framework must still support this functionality as described version 1.0
  - Secure Partition developers are recommended to use other mechanisms to replace use of this feature

Next steps

- Public release of the PSA FF-M 1.1 Extensions beta specification
- Please provide feedback on the 1.1 Extensions in the TF-M mailing list, or to arm.psa-feedback@arm.com
- Feedback and future updates to the specification can be discussed on the TF-M mailing list, and will also feature in future Technical Forums

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה