TRUSTED® COMPUTING GROUP

SPECIFICATION

Cyber Resilient Module and Building Block Requirements

Version 1.0
Revision 0.100
March 1, 2022

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

Disclaimers, Notices, and License Terms

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.

Change History

REVISION	DATE	DESCRIPTION
1.00/0.08	October 19, 2020	Initial Release
1.00/0.10	March 1, 2022	Second release inclusive of feedback.

Contributors

Name	Organization
Matthew Areno	Intel Corp.
David Challener	John Hopkins University, Applied Physics Lab
Jonathan Cooke	Google
Shiva Dasari	HPE
Paul England	Microsoft
Nick Grobelny	Dell Inc.
Steve Hanna	Infineon Technologies
Jeff Jeansonne	HP Inc.
Dean Liberty	AMD Inc.
Andrey Marochko	Microsoft
Jim Mann	HP Inc.
Dennis Mattoon	Microsoft
Amy Nelson	Dell Inc.
Dave Riss	Intel Corp.
Adolph Seema	Microchip Technologies Inc.
Rob Spiger	Microsoft
Silviu Vlasceanu	Huawei
Dick Wilkins	Phoenix Technologies Ltd.

CONTENTS

1 Scope

Start of informative comment.

This specification defines capabilities that provide a foundation for Device Vendors to build Cyber Resilient Devices. Devices can be made of numerous individual components and layers, any of which could be compromised. To recover from compromise, a device may need servicing (patches or restoration) of the code and configuration of one or more components and/or layers. The primitive building blocks defined in this specification can be used to protect the bottommost layer and support its ability to perform recovery for the layer above it. To make the servicing capabilities general enough to be applicable across many different components and layers, this specification defines an abstract Cyber Resilient Module (CRM) that has (1) a lower layer comprising an engine assumed to be uncompromised that performs recovery and (2) an upper layer comprising a target that may be compromised but can be recovered by the engine. The CRM concept can be applied repeatedly to each successive layer and components to support recovery for every layer and every component in a device (except the bottommost layer, because there is no layer underneath it).

This specification includes a library of primitive building blocks that provide capabilities supporting read and write protection of persistent storage, initiation of recovery actions in the event of failure or compromise, and isolation of recovery capabilities from compromise. To implement cyber resilient capabilities, this specification combines the building blocks together into an abstract computing component called a CRM. Such a module might be an entire System on a Chip (SoC) integrated into an Internet of Things (IoT) device or might be a Microcontroller Unit (MCU) that is part of a component that is incorporated into a larger complex device. If an SoC or an MCU has numerous layers of code and configuration, the first and second layer could comprise a CRM and the second and third layers could be another CRM, and so on. A single Cyber Resilient Device may consist of one or many CRMs.

This specification anticipates that additional building blocks, including alternatives to the building blocks in this version of this specification, may be defined in the future. The limited set of building blocks in this version of the specification explicitly depend on the use of “early boot” as a “safe” environment for recovery activities. The premise being that if an engine that performs recovery or patching activities runs earlier than code that needs patching or might be compromised by malware, the recovery actions can occur unimpeded. Future versions could potentially use a “safe” runtime environment post-boot for recovery activities. Note: “Early boot” is a relative term and in some implementations, it could be quite late in boot.

As such, this specification is relevant for a variety of computing environments, such as:

- Central Processing Units (CPU), MCUs, SoCs
- Storage and peripheral device controllers (both discrete and integrated)
- PLDs (Programmable Logic Devices), programmable ASICs (Application Specific Integrated Circuits)
- Security co-processors (e.g., a Trusted Platform Module (TPM))

Section 5 of this specification includes an informative outline of a cyber resilient architecture built using protected capabilities. The architecture can better protect device resources (e.g., hardware) and information (e.g., code and data) and ensure consistent operation in response to a variety of threats. For example, depending on the implementation, the capabilities can help the maintainer of a device perform local and/or remote servicing.

The mechanisms in this specification are well suited to constructing devices that meet the requirements of NIST SP 800-193 [1]. The Device Vendor can start with capabilities satisfying the requirements in this specification and then add platform and use case specific requirements and functionality to complete a finished device.

This specification does not attempt to protect the engine that performs recovery from all threats. It has requirements specifically protecting whatever performs recovery from whatever may be compromised. Implementers should refer to other specifications to mitigate other sources of threats to the engine that performs recovery.

Mitigating and recovering from physical attacks is out of scope for this specification.

End of informative comment

1.1 Document Structure

Start of informative comment.

This specification contains an introduction in section 4 that explains the motivation for the specification.

Section 5 provides a high-level overview of cyber resilient concepts and device considerations that inform secure and reliable capabilities for management and recovery– even if most of the main device firmware or subcomponents of a device are compromised. The major architectural elements for resilience activities are also in section 5. These elements help create a distinction between what might need to be serviced (by preventative patching or recovery after compromise), whatever performs the recovery, and how recovery actions are authorized. Section 5 describes the relationship between whatever performs the servicing and whatever is serviced, by grouping them together as a CRM. It introduces the concept of "resetting" the CRM so servicing actions can occur safely even if the portion of the module that can be serviced is uncooperative or compromised by malware. A Resilience Orchestrator architectural element has the responsibility to make sure the interactions between all the elements internal and external to a CRM are well coordinated.

Section 6 introduces Cyber Resilient Building Blocks internal to a CRM that support the relationship between whatever will be serviced and whatever does the servicing. It explains how building blocks can be used for resilience and may provide benefits for other scenarios. The section includes design guidance and considerations for using the building blocks.

Section 7 considers the impact of frequent recovery attempts and the consequences when disconnected from a network.

Section 8 describes the threat model, which defines the types of attacks considered in scope versus those that are out of scope and defines the security boundaries related to the cyber resilient architectural elements.

Section 9 formally describes the building blocks and details their requirements.

For the scope of CRMs defined in this specification (e.g., using "early boot" as a "safe" environment) sections 10 and 11 provide context and requirements on how the primary architectural elements are intended to be used by a complete design. In section 11 the different flavors of CRMs (self-recoverable versus symbiotic) have separate requirements. Section 11 includes additional recommendations and requirements for attestation and Secure Boot.

Section 12 defines nomenclature for Device Vendors to concisely describe what capabilities their device has implemented.

The Appendix has four informative sections. Section 13.1 describes how Protection Latches can also be helpful to implement Roots of Trust for attestation and Secure Boot. Sections 13.2 and 13.3 look at two holistic examples of applying the concepts in this specification to mitigate threats for a simple device and in a subcomponent of a more complex device. Section 13.4 discusses how the building blocks in this specification could be applied in different IoT scenarios.

End of informative comment

1.2 Key Words

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document's normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.3 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment.

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 References

[1] National Institute of Standards and Technology, "SP 800-193, Platform Firmware Resiliency Guidelines," May 2018. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-193/final.

[2] Trusted Computing Group, "TCG Glossary Version 1.1, Revision 1.0," 11 May 2017. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-glossary/.

[3] Trusted Computing Group, "Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 1.59 (or higher)," 8 November 2019. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/.

[4] Trusted Computing Group, "DICE Layering Architecture Version 1.0 Revision 0.19," 23 July 2020. [Online]. Available: https://trustedcomputinggroup.org/resource/dice-layering-architecture.

[5] Trusted Computing Group, "TCG Guidance for Secure Update of Software and Firmware on Embedded Systems," 10 February 2020. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-guidance-for-secure-update-of-software-and-firmware-on-embedded-systems/.

[6] International Electrotechnical Commission, "IEC TS 62443-1-1:2009: Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models," 30 July 2009. [Online]. Available: https://webstore.iec.ch/publication/7029.

[7] International Electrotechnical Commission, "IEC 62443-2-1:2010: Industrial communication networks - Network and system security - Part 2-1: Establishing an industrial automation and control system security program," 10 November 2010. [Online]. Available: https://webstore.iec.ch/publication/7030.

[8] International Electrotechnical Commission, "IEC TR 62443-2-3:2015: Security for industrial automation and control systems - Part 2-3: Patch management in the IACS environment," 30 June 2015. [Online]. Available: https://webstore.iec.ch/publication/22811.

[9] International Electrotechnical Commission, "IEC 62443-2-4:2015+AMD1:2017 CSV Consolidated version: Security for industrial automation and control systems - Part 2-4: Security program requirements for IACS service providers," 24 August 2017. [Online]. Available: https://webstore.iec.ch/publication/61335.

[10] International Electrotechnical Commission, "IEC TR 62443-3-1:2009: Industrial communication networks - Network and system security - Part 3-1: Security technologies for industrial automation and control systems," 30 July 2009. [Online]. Available: https://webstore.iec.ch/publication/7031.

[11] International Electrotechnical Commission, "IEC 62443-3-2:2020: Security for industrial automation and control systems - Part 3-2: Security risk assessment for system design," 24 June 2020. [Online]. Available: https://webstore.iec.ch/publication/30727.

[12] International Electrotechnical Commission, "IEC 62443-3-3:2013: Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels," 7 August 2013. [Online]. Available: https://webstore.iec.ch/publication/7033.

[13] International Electrotechnical Commission, "IEC 62443-4-1:2018: Security for industrial automation and control systems - Part 4-1: Secure product development lifecycle requirements," 15 January 2018. [Online]. Available: https://webstore.iec.ch/publication/33615.

[14] International Electrotechnical Commission, "IEC 62443-4-2:2019: Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components," 27 February 2019. [Online]. Available: https://webstore.iec.ch/publication/34421.

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. Additional terms are also defined in the TCG glossary [2].

3.1 Glossary

Term	Definition
Attention Signal	An output signal for attention generated by a component (e.g., a Watchdog Counter elapsing or the press of a physical button).
Authorized Deferral Watchdog Counter (ADWC)	A Watchdog Counter that can be deferred only by providing a Deferral Ticket satisfying its Deferral Policy or using a protected capability.
Authorized Deferral Wakeup Watchdog Counter (ADWWC)	A Watchdog Counter that is both an Authorized Deferral Watchdog Counter and a Wakeup Watchdog Counter.
Basic Watchdog Counter (BWC)	A Watchdog Counter that can be cancelled or indefinitely deferred without authorization.
Cyber Resilience	The ability of an information system to protect against and detect attack or compromise, to continue to operate while under attack or compromised, even if in a degraded or debilitated state, and to reestablish proper operation after attack or compromise.
Cyber Resilient Building Block (CRBB)	A building block (e.g., a Watchdog Counter or a Protection Latch) useful for supporting cyber resilient capabilities in a Cyber Resilient Module.
Cyber Resilient Device	A device containing one or more Cyber Resilient Modules.
Cyber Resilient Module (CRM)	A computing module (as described in section 5.4) that implements the requirements of this specification consisting of a protected Resilience Engine that can service a Resilience Target even when the target is compromised.
Deferral Policy	A policy specifying the requirements for generating and validating a Deferral Ticket.
Deferral Ticket	An authorization satisfying a Deferral Policy used to defer an Authorized Deferral Watchdog Counter.
Detection	A process of discovering when a device becomes compromised or begins manifesting aberrant behavior. Detection can be performed locally or remotely (e.g., by a Resilience Authority).
Device	A device integrates a programmable component with other optional programmable components and peripherals.
Device Vendor	An entity that produces a device or a Cyber Resilient Device and provides it to users.
Latchable Watchdog Counter (LWC)	A Watchdog Counter that cannot be deferred while it is Enabled without the use of a protected capability.
Latchable Wakeup Watchdog Counter (LWWC)	A Watchdog Counter that is both a Latchable Watchdog Counter and a Wakeup Watchdog Counter.
Initialize Signal	A signal used to securely convey security relevant events, like power-on, restart, etc., which resets or reinitializes a Cyber Resilient Building Block.
Module Reset	A reset event (as described in section 5.4) of a Cyber Resilient Module that sends an Initialize Signal to all its Cyber Resilient Building Blocks and starts its Resilience Engine.
Persistent Storage	Storage for code and data, the contents of which are not affected by power loss.
Protection	A capability that resists unauthorized modification of its target's behavior.
Protection Latch	A Cyber Resilient Building Block that can be activated to prevent reading from and/or writing to storage.
Read-Protection Latch (RPL)	A Protection Latch that prevents reading when activated.

Read-Write-Protection Latch (RWPL)	A Protection Latch that prevents reading and writing when activated.
Reconfiguration Policy	A policy specifying the requirements for generating and validating a Reconfiguration Ticket.
Reconfiguration Ticket	An authorization satisfying a Reconfiguration Policy used to reconfigure an Authorized Deferral Watchdog Counter.
Recovery	The process of restoring integrity, functionality and trustworthiness of a device that has been compromised or suffered a fault.
Resilience Authority (RA)	An entity that authorizes a Resilience Engine to perform servicing actions on a Resilience Target.
Resilience Engine (RE)	An engine that services one or more local Resilience Targets. A RE recognizes one or more RAs for servicing instructions.
Resilience Orchestrator (RO)	Functionality that coordinates interactions between external components and things internal to the Cyber Resilient Module.
Resilience Target (RT)	A mutable engine that is serviceable by one or more REs. The RT cannot include any of its REs.
Root of Trust	This term is defined in the TCG glossary [2].
RT Domain	The RT and any resources or information the RT can modify or control directly or indirectly.
Secure Execution Environment (SEE)	A Cyber Resilient Building Block that prevents computation that occurred in the past or may occur concurrently from affecting the initial conditions or computation in the environment.
SEE Resources	Resources and information the RT cannot modify or control (directly or indirectly)
Self-Recoverable Cyber Resilient Module	A Cyber Resilient Module that does not need assistance to initiate recovery actions.
Symbiotic Cyber Resilient Module	A Cyber Resilient Module that needs assistance to initiate recovery actions.
Temporal Protection	A form of protection achieved by making sensitive data or functions inaccessible (e.g., by using a Protection Latch or erasing data from Random Access Memory (RAM) and registers) before passing control to other, more vulnerable, entities.
Wakeup Watchdog Counter (WWC)	A Latchable Watchdog Counter or an Authorized Deferral Watchdog Counter that is unaffected by power management performed by the RTs in the same Cyber Resilient Module.
Watchdog Counter	A type of Cyber Resilient Building Block that sends an Attention Signal after time or events have elapsed. This is a generic term for a Basic, Latchable, Authorized Deferral or Wakeup Watchdog Counter.
configuring	Setting of a Watchdog Counter's Enable/Disable state, expiration value and/or other parameters.
deferral	A reconfiguration of a Watchdog Counter that delays the expiration event.
expiration value	The period of time or number of events that will elapse before a Watchdog Counter generates an Attention Signal.
Write-Protection Latch (WPL)	A Protection Latch that prevents writing when activated.

3.2 Abbreviations

Acronym	TERM
ATA	Advanced Technology Attachment
ADWC	Authorized Deferral Watchdog Counter
ADWWC	Authorized Deferral Wakeup Watchdog Counter
BMC	Baseboard Management Controller
BWC	Basic Watchdog Counter
CDI	Compound Device Identity
CPU	Central Processing Unit

CRM	Cyber Resilient Module
CRBB	Cyber Resilient Building Block
DoS	Denial-of-Service
IACS	Industrial Automation and Control Systems
IoT	Internet of Things
KDF	Key Derivation Function
LWC	Latchable Watchdog Counter
LWWC	Latchable Wakeup Watchdog Counter
MCU	Microcontroller Unit
NVMe	Non-Volatile Memory Express
OS	Operating System
RA	Resilience Authority
RE	Resilience Engine
RT	Resilience Target
RAM	Random Access Memory
ROM	Read Only Memory
RO	Resilience Orchestrator
SCSI	Small Computer System Interface
SEE	Secure Execution Environment
RPL	Read-Protection Latch
RWPL	Read-Write-Protection Latch
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
RTV	Root of Trust for Verification
SoC	System on a Chip
SP	Special Publication
SMM	System Management Mode
TCB	Trusted Computing Base
TPM	Trusted Platform Module
UDS	Unique Device Secret
WC	Watchdog Counter
WWC	Wake-up Watchdog Counter

4 Introduction

Start of informative comment.

NIST SP 800-193 [1] identifies the following three principles to support the resilience of platforms against potentially destructive attacks:

> ***Protection***: *Mechanisms for ensuring that Platform Firmware code and critical data remain in a state of integrity and are protected from corruption, such as the process for ensuring the authenticity and integrity of firmware updates.*
>
> ***Detection***: *Mechanisms for detecting when Platform Firmware code and critical data have been corrupted.*
>
> ***Recovery***: *Mechanisms for restoring Platform Firmware code and critical data to a state of integrity in the event that any such firmware code or critical data are detected to have been corrupted, or when forced to recover through an authorized mechanism. Recovery is limited to the ability to recover firmware code and critical data.*

All Internet-connected devices should be designed to protect themselves against network-based attacks. As such, Device Vendors employ a wide range of hardware and software-based protection technologies to keep devices secure. Unfortunately, bugs and misconfigurations still lead to damaging exploits.

Recovering a badly compromised computing device today usually involves manual intervention, which can be costly or time consuming, especially at scale. For example, new firmware or a new OS must be loaded from an external storage device or a second computer. The device must then be rejoined to network services using passwords, or other credentials, often under conditions of physical security.

The rapid growth in the use of IoT is driving an order-of-magnitude increase in the number of computing devices. These devices will be built from the same imperfect software in use today, but manual remediation will be much less practical or even infeasible because the devices are too numerous, too inaccessible, or lack a suitable interface.

Technologies that support reliable and secure remote computer management and recovery are already available for more costly devices. For example, Service Processors or Baseboard Management Controllers (BMCs) are employed to manage desktops and servers, and intelligent backplanes are used to manage blades in data centers. However, these technologies are either unsuitable or inefficient for IoT because of their cost, form factors, power needs, or the lack of an out-of-band management channel.

This specification defines a minimal set of capabilities or mechanisms that enable Cyber Resilient Devices to be built even at the lowest end of the cost/performance/complexity spectrum. This includes IoT devices and microcontrollers used in a wide range of applications. It also supports more complex devices by providing resilient capabilities to independent subcomponents of devices that may be directly or indirectly connected to a network.

The capabilities described are designed to be both simple to implement and simple to use. The simplicity decreases the chance that the security-critical components that operate them will be vulnerable, while also minimizing cost, power consumption and size of the hardware.

End of informative comment

5 Introduction to Architectural Elements

Start of informative comment.

The next three sections describe the behavior and interactions between the Resilience Target (the entity that is serviced), the Resilience Engine (the entity that performs the servicing action), and the Resilience Authority (the entity that authorizes a servicing action). The entities and their relationships are illustrated in Figure 1.

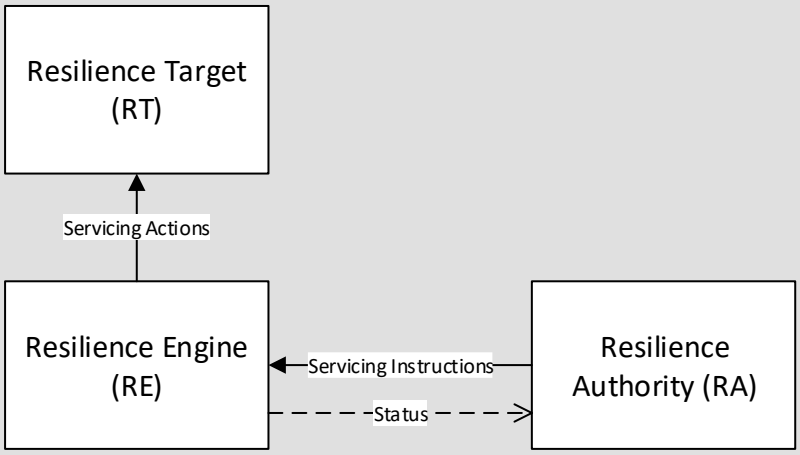

Figure 1: Relationships between the Resilience Target, Resilience Engine, and Resilience Authority

End of informative comment

5.1 Resilience Target (RT)

Start of informative comment.

A Resilience Target (RT) is a mutable engine. The engine's resources could be hardware, firmware, software, configuration information, runtime state, etc. Because the RT is mutable, it is assumed to be at risk of compromise by malware. The RT could be big or small. It might include an entire Operating System (OS) and its applications on a general-purpose computer, a single dedicated function in an IoT appliance, or the firmware and configuration for a subcomponent of a device, such as a graphics card.

This specification neither prescribes nor limits the design of the RT. It may range from a simple monolithic application package (e.g., in a sensor-style IoT device), a firmware image (e.g., in a hardware component of a larger module), through a stage-booted OS running applications where each stage verifies the next one, to a full-fledged hypervisor running multiple OSes and supporting hardware resources. The RT may use other available runtime protection technologies or security extensions, such as CPU privilege levels, System Management Mode (SMM), etc.

The purpose of defining a RT is to draw a boundary around the mutable resources that other components in this specification can service. This specification places no integrity protection requirements on the RT. Use of existing industry best practices for implementing and maintaining the integrity of the RT are encouraged and relevant. The ability to detect the loss of integrity of a RT can be a useful technique to decide whether to initiate recovery actions. This specification does not mandate Detection capabilities for the RT, but it does include requirements for support for attestation of the RT in section 11.3.1. Device operators could use attestation to confirm that recovery actions on the RT completed successfully. An alternative to Detection is to periodically assume the RT has lost its integrity and to regularly recover it.

The purpose of this specification is to ensure that a device containing a RT is resilient by providing the capability to reliably service and restart the RT due to:

- An actual loss of integrity of the persistent image of the RT, including code and configuration.
- Misbehavior or a loss of integrity or stability of the RT runtime.

- A need to reconfigure, update or patch the RT.
- Other reasons not foreseen at the time this specification was written.

RT servicing actions could result in small or large changes to the persistent stored image of the RT (e.g., code or configuration) and may include a restart of the RT from its persistent stored image, a backup image or a new image downloaded from a remote location. Cooperation from the RT is not a precondition for service actions to succeed. The servicing actions need to be possible if the RT has crashed, is compromised by malware, deadlocked, asleep, in a low power state or uncooperative for any other reason.

One approach to resilience is physically visiting devices, turning them off, starting a recovery mode and loading new code and configuration. This specification does not exclude optional support for the physical servicing of a RT, but its intent is to support the servicing of RTs without relying on physical presence. It is anticipated in many IoT scenarios the devices may be too numerous, deployed to inconvenient physical locations or lack interfaces for servicing using physical presence to be practical. For subcomponents of devices, similar challenges may necessitate the need for remote servicing.

Many existing devices include their own patching capabilities whereby an OS or application regularly checks for updates from its manufacturer, downloads patches and installs them. This specification does not preclude the RT from using such an architecture to update itself and a RT that is not compromised may be well suited to installing updates while minimizing impact to the device functionality and users. This specification focuses on how to service the RT when it is not functioning properly due to compromise or other reasons.

End of informative comment

5.2 Resilience Engine (RE)

Start of informative comment.

A Resilience Engine (RE) is a component that can service one or more RTs. The RE is responsible for servicing RTs even when they are uncooperative. In a CRM, the RE requires direct access to the RT(s) to perform servicing. The RE is local to its RTs because it needs to function even if the remote network is unavailable. The RE may support configurable policies for when and how it performs servicing actions. To be able to respond to future circumstances not foreseen at the time the device was created, the RE is expected to be able to receive new instructions from an authorized entity called a Resilience Authority (RA).

Devices may only be intermittently connected to Resilience Authorities through, for example, a network, a physical servicing cable connection, or other architectures. Networks and other connections can sometimes be unavailable, so the architecture in this specification assumes that servicing actions on the RT may need to be invoked with or without connectivity between the RA and RE. Communication between the RA and RE is discussed in more detail in section 5.3.

A RT might become compromised, and subsequently might try to prevent the RE from performing servicing actions. The RE protects itself from compromised and misbehaving RTs by using various CRM capabilities, some of which are summarized below:

- Invocation – The RE receives control when servicing actions need to be taken (even if the RT is uncooperative).
- Secure Execution Environment (SEE) – Used in this specification to isolate and protect RE actions from interference from its RTs, such that the RE is not subverted by any previous or concurrent RT actions on the device whether direct or indirect (see section 6.1 for more details).
- Control of the RT's persistent storage – The RE can read and write the persistent storage that holds the RT image and configuration.
- Control of the RE's persistent storage – An ability to read, write and protect the RE code and configuration and RT recovery code and/or data (e.g., local backup copies of the RT) such that the RT cannot modify it.

The RE needs to be able to perform the following activities:

- Interact with one or more Resilience Authorities – The RE can interact (directly or indirectly) with RAs to receive instructions and patches without interference from its RTs.
- Obtain patches (e.g., download them from the Internet, or through other networks or buses) without interference from its RTs.
- Configure the frequency with which the RE receives control to perform servicing per the current policy of the RE unless the frequency is fixed (hardcoded).
- Modify the stored image of the RT as authorized by a RA (e.g., update or reconfigure). This is generically referred to as servicing.
- Turn on protections for the RE's persistent storage before invoking the RT.

The RE may optionally be used to perform the following actions due to its ability to implement capabilities without interference from its RTs:

- Assess the health of the RE and/or its RTs via Detection by cooperating with Roots of Trust:
  - Root of Trust for Verification (RTV) to scan for integrity loss of the RT.
  - Root of Trust for Measurement (RTM), Root of Trust for Storage (RTS) and/or Root of Trust for Reporting (RTR) to attest the RT locally or remotely.
- Follow other instructions from a RA (e.g., quarantine the RT module).
- Take defensive measures to reduce potential damage if the RT is suspected of being compromised.
- Implement defensive policies if communication (direct or indirect) with its RAs is not available.
- Update the RE's own code, configuration, and policy.
- Prevent the RT from being downgraded.
- Generate and maintain logs reporting events related to the attestation and/or servicing of the RT.

A clear distinction between the RE and RT is which one can be serviced when compromised. This specification provides capabilities to service the RT (via the RE) when the RT has lost integrity or is not cooperative. In contrast, this specification does not provide capabilities to reliably service the RE when its integrity is lost, or it is not cooperative. It is conceivable a layered architecture could repeat the paradigm by designating an RE of an upper layer as an RT of a lower layer.

This specification does not mandate Detection capabilities on a RE, but it does include requirements to support attestation of the RE in section 11.3.1.

End of informative comment

5.3 Resilience Authority (RA)

Start of informative comment.

Attack techniques are always improving, and products believed to be secure when initially manufactured inevitably require patches during the lifecycle of devices for unanticipated vulnerabilities or mistakes.

For consumer scenarios, automatic updates are an effective tool to keep devices patched. In enterprise or government settings, administrators often want the flexibility and control to choose what updates or configuration changes are deployed and when. For high security or safety critical products, multiple approvers may need to agree to servicing actions.

The Resilience Authority (RA) is an entity that authorizes a RE to perform servicing actions on a RT. This specification abstracts the complex set of activities that can occur before the final decision to conduct servicing actions. The RA may encapsulate any number of decision makers (e.g., a Device Vendor, the owner, an owner-operated cloud management service, an IT administrator, etc.) authorized to control servicing actions on the RT

(e.g., restore from a backup copy) and optionally the RE (e.g., firmware updates or reconfiguration). In practice, activities that occur prior to a RA authorizing service actions could involve Device Vendor and customer incident response, patch management, testing and acceptance, organizational operational controls and change management and many other steps.

In a consumer setting the authorization by a RA could be as simple as the Device Vendor signing a code update so it can be verified locally by the RE prior to installation. For a more complex scenarios the RA implementation could require multiple entities to sign updates or policy changes with a device unique key that the RE can verify. This specification allows one or more RAs to independently issue servicing instructions to an RE.

The RA is expected to continue functioning and approve recovery strategies when a RT is compromised, and it should be able to respond to new circumstances not anticipated when the device was manufactured. The RA credentials used to authorize updates to the RE are an obvious target for attacks because of the RA credential's ability to cause changes to the RE and RT.

The communication channel between the RA and the RE could be a direct encrypted connection over the Internet or achieved by sending data through many indirect nodes that are infrequently connected. For IoT scenarios, a RA could be a cloud management service that the device is tied to during provisioning. The device owner could connect to a corresponding management portal to approve servicing changes to the device. In a high security server scenario, the RA might be a data center operator management console in conjunction with BMCs (local to the device) that receive and sign instructions for each individual server's RE with server specific keys.

If the recovery functionality of a RE depends, at least partially, on a RA, the RE should have a fallback behavior in case the RA is unavailable when recovery actions need to be performed. This behavior should preserve resilience characteristics of the module while containing adverse effects of a RT compromise.

End of informative comment

5.4 Cyber Resilient Module (CRM) and Module Reset

Start of informative comment.

The primary interactions between the RA and RE and between the RE and RT are shown below in Figure 2. There may be additional bidirectional communication between the components. For example, the RE may help measure the RT code and configuration and attest the information to the RA.

The RT and RE have domain boundaries that do not correspond to explicit resources in a device architecture. Some examples are:

- The RT and RE share hardware resources and in addition:
  - The RE includes device firmware and the RT includes device software.
  - The RE includes a software layer that executes before the RT in the boot sequence.
  - The RE includes an OS and the RT includes an application.
- The RE is host processor and firmware and the RT is the microcontroller and firmware of an add-in card.
- The RE is a BMC and the RT is the host platform firmware and OS.

This specification defines a relationship between the RT and its RE, and requires multiple supporting protected capabilities so the RE can service the RT, but the RT cannot modify the RE. These capabilities are (a) protected storage available to the RE, (b) a way to reliably transfer control to the RE, and (c) isolation and protection of the RE actions from interference from its RTs. A single device could have multiple sets of components with RE and RT relationships, each interacting with each other independently, performing servicing actions authorized by their RA.

This specification defines a CRM as all the components needed to enable a Device Vendor to construct a RE and a RT such that (1) the RE can service the RT, (2) the persistent storage of the RE is protected from modification by

the RT, (3) the RE can gain execution control from the RT and (4) the RE execution environment is protected from interference by the RT.

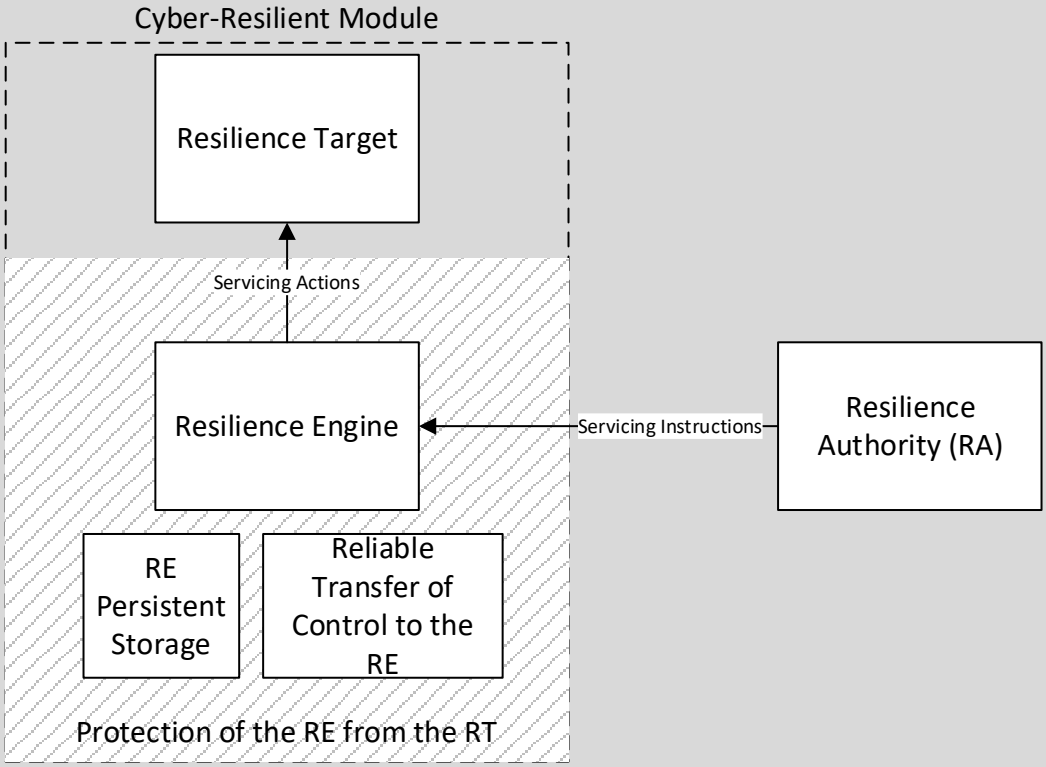

Figure 2: CRM

One or more CRMs can be used to implement a Cyber Resilient Device.

A reset of a CRM does not necessarily correspond to a reset of an entire device, although it could for some implementations. The reset of a CRM means resetting all components internal to the module. Later in this specification in sections 6 and 9, each of the capabilities supporting the CRM are defined as individual building blocks, most of which can receive an Initialize Signal that resets them. The term Module Reset is defined as the reset of a CRM that sends an Initialize Signal to all its individual building blocks and transfers control to the RE before the RT.

Figure 3 illustrates the high-level architecture of a CRM with examples of components that might be used in an implementation.

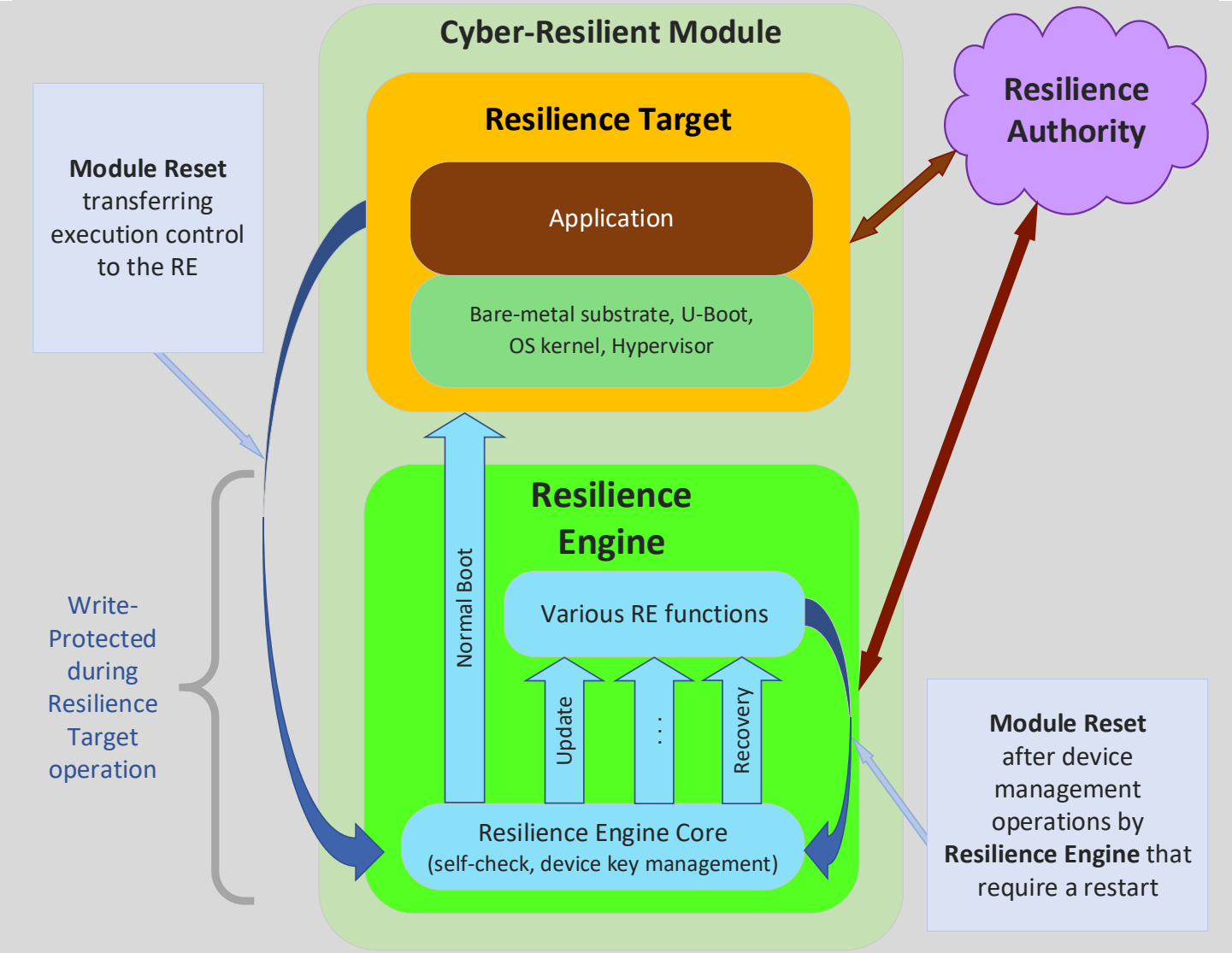

Figure 3: Example CRM using the building blocks

A complex device may include multiple CRMs, some of which can be reset without resetting the others. For example, a PC Client device could have a first CRM consisting primarily of the host CPU, main memory, and a hard drive (in addition to other building blocks defined in this specification). The host platform RE might consist of the UEFI environment and the host CPU, and the RT might consist of the OS software and host CPU. Additional CRMs might consist of mostly independent subcomponents like a camera, fingerprint reader or graphics card, each with their own MCU, storage, firmware, and software. Each subcomponent might be designed to use early initialization firmware in its RE and use runtime firmware in its RT (with the RE and RT both including the subcomponent's microprocessor and other computing resources). While it could be possible to reset CRMs or subcomponents independently, if the host CPU CRM is reset, it would likely require resetting all the CRMs in the device.

A simpler device having only a single CPU could also have multiple CRMs. The first CRM might be designed so the earliest boot code is used by its RE and second stage boot loader is used by the RT. The second CRM might be designed with the second stage boot loader used as part of its RE with any software that runs later used as part of its RT. In this example, it would not be possible to perform Module Reset on either CRM individually.

End of informative comment

5.5 Resilience Orchestrator (RO)

Start of informative comment.

The architectural elements above (RT, RE, RA and the CRM) interact with supporting building blocks and need to be connected to other device components in the larger context of their environment. To coordinate activities that involve multiple components interacting in concert to perform activities like Module Reset, an additional architectural element called the Resilience Orchestrator (RO) is defined. The RO coordinates interactions between external components (e.g., device power management, other CRMs, physical signals, etc.) and things internal to the CRM (i.e., RT, RE and its internal building blocks). It receives and processes notifications when the RE needs to execute and acts by resetting building blocks, performing Module Reset, and invoking the RE. It also interfaces with power management (if needed) and embodies the interconnection of any physical signals. The implementation of the RO is expected to be Device Vendor and device specific.

While a RO implementation can be sophisticated in its management of notifications to and from individual building blocks, this is not a requirement. RO logic could be as simple as the assertion of a hardware reset signal (i.e., a typical power-on reset) in response to an indication that control should be transferred to the RE.

End of informative comment

6 Cyber Resilient Building Blocks Introduction

Start of informative comment.

This section introduces the concept of individual building blocks that support resilience. More detailed informative text and requirements are in section 9. The individual building blocks supporting device resilience are called Cyber Resilient Building Blocks (CRBBs). The section also includes guidance to help Device Vendors and support other specifications that provide precise requirements to recover RTs on specific device types. Building blocks defined in this specification can be utilized outside the context of a CRM. This specification does not specify requirements on how CRBBs are used outside the context of a CRM.

End of informative comment

6.1 Secure Execution Environment (SEE) for the Resilience Engine

Start of informative comment.

A Secure Execution Environment (SEE) is a CRBB that prevents computation that occurred in the past or may occur concurrently from affecting the initial conditions or computation in the environment.

General examples of a SEE are: (1) running Read Only Memory (ROM) code just after power cycling a device because everything that the device used to be doing is flushed when power is lost and (2) a BMC in a server environment because whatever is running on the host cannot influence the code running on BMC.

Instead of requiring a SEE to protect the RE from all prior computation, this specification requires the SEE to be protected only from prior computation by the RT or anything the RT can control (directly or indirectly). This means the RE might trust and use resources the RT cannot control or influence. Also, the RE might still access information controlled by the RT and use it as untrusted information.

This specification does not require the SEE to be protected from all concurrent activities, but only from concurrent RT activities and concurrent activities by anything the RT can control (directly or indirectly) until the RE completes. Resources that execute concurrently to the RE may be able to influence the RE computation (and may be able to influence computation that occurs in the SEE before the RE completes), provided the other concurrent resources are not able to be controlled by the RT, either directly or indirectly. After the RE completes, the RT can execute in the SEE. In a layered CRM architecture, where there are multiple layers of REs and RTs, the RT for the lowest layer can be the RE for the next layer. Running the lowest layer RT in the SEE (after the lowest layer RE completes) can help protect it from subsequent RTs.

This specification specifically protects the RE from the RT (directly and indirectly). Other common design requirements to protect the RE from non-RT related attacks are out of scope but remain important for resilient products.

End of informative comment

6.2 Signals and Programmatic Interfaces

Start of informative comment.

The CRBBs defined in this specification have two classes of interface: Programmatic Interfaces and Signals.

- **Programmatic Interfaces** are freely accessible. For example, they could be used by software running in the RE and/or the RT. Examples of Programmatic Interfaces include the interfaces to configure or activate or enable building blocks such as Protection Latches or Watchdog Counters. Some programmatic actions require authorization.

- **Signals** are used to communicate security relevant events. The propagation of a signal from its source to its destination without interference will be implemented as a protected capability by the Device Vendor.

This specification defines two different types of Signals each with their own specific protected capability.

- **Attention Signals** – used when a source needs to notify a destination that something needs to happen (e.g., when a transfer of control to the RE needs to occur). An Attention Signal is generally used to transfer control back to the RE so it can perform servicing actions.
- **Initialize Signals** – used by a source to change the security relevant settings or protections of a destination (e.g., turning off write protection for storage). An Initialize Signal is generally used to reset or reconfigure a building block.

Note: Despite the use of the word "Signal," a signal is not required to be implemented literally as an electrical signal on a physical wire.

End of informative comment

6.3 Storage Protection

Start of informative comment.

A portion of a RE may be implemented in the form of code and associated configuration data kept in a device's persistent storage. Since an untrusted RT may share the same hardware resources as the RE, special precautions may be required to protect the RE and its associated data.

A RE may need storage protection capabilities for two primary purposes.

- **Read protection-** to prevent secret values used by the RE from being read by the RT.
- **Write protection-** to prevent modification to the RE code and configuration data after the RE passes control to the RT.

End of informative comment

6.3.1 Read-Protection

Start of informative comment.

Examples of secret values that need read protection include:

- A device-unique secret that is used to establish device identity.
- Cryptographic keys used for data encryption.

A Read-Protection Latch (RPL) is a protected capability that allows read access to one or more regions of persistent storage to be disabled until the next Module Reset event re-enables access. Module Reset must be the only way to unlock the latch. The RE will typically read a secret value from storage, and then activate a RPL for that region of storage before passing control to the RT.

End of informative comment

6.3.2 Write-Protection

Start of informative comment.

Examples of data that need write protection include:

- The RE firmware
- Security-critical RE configuration (e.g., public keys or certificates used to validate patches or the RA)
- Optionally, the RT recovery information (e.g., a backup RT image)

When the RE receives control after Module Reset, it should be able to perform an update of the RT, and possibly its own resources (e.g., firmware, software, configuration, etc.). The RE may need to be updated to extend or

change its functionality, or to reconfigure it. This requires the persistent storage holding the RE code and configuration to be writable following Module Reset.

The RE may also need to protect one or more recovery or configurations images from modification. The RE may be responsible for protecting other important device-specific state as well.

These needs can be addressed by a Write-Protection Latch (WPL) – a mechanism that prevents write access to one or more regions of storage. Module Reset must be the only way to unlock the latch.

Any potentially harmful actions, such as external input analysis, network communication, or initializing the RT and transferring control to it, should be undertaken by the RE after all its WPLs are activated.

End of informative comment

6.4 Attention Signal Generators

6.4.1 Introduction

Start of informative comment.

While the RE has the means to recover a damaged or compromised RT, it has to get control over execution in order to perform its functions. However, without additional protected capabilities, it can be hard for the RE to regain control. For example, when the RT is hung because of a software bug, or when the RT is compromised and malware intentionally prevents it from triggering Module Reset (that is needed to transfer control to the RE).

A solution to this problem is provided by capabilities called Attention Signals – an output signal for attention generated by a component. This specification uses Attention Signals to interrupt the RT and provide an opportunity for the RE to run (e.g., perform servicing actions, etc.) before the RT is invoked again. Attention Signals can be generated remotely (e.g., by a RA), locally (e.g., by hardware, software, etc.) or other means (e.g., by an interactive user via Device Vendor defined mechanisms).

This specification defines a set of building blocks to generate Attention Signals. More building blocks that generate Attention Signals may be added over time.

End of informative comment

6.4.2 Watchdog Counters

6.4.2.1 Introduction

Start of informative comment.

A Watchdog Counter is a mechanism that triggers an Attention Signal unless it is serviced (i.e., reconfigured and/or restarted) before its expiration value is reached. This specification describes four Watchdog Counter variants:

- Basic Watchdog Counter (BWC)
- Latchable Watchdog Counter (LWC)
- Authorized Deferral Watchdog Counter (ADWC)
- Wake-up Watchdog Counter (WWC)

If a Watchdog Counter is implemented in such a way that while enabled, it cannot be disabled or reconfigured by the RT, it is considered cyber resilient.

End of informative comment

6.4.2.2 Basic Watchdog Counter (BWC)

Start of informative comment.

A Basic Watchdog Counter (BWC) is a Watchdog Counter that can be serviced at any time. It may be indefinitely deferred. A BWC is useful for increasing reliability, but malware may interfere with it. Since a BWC that is accessible

to the RT can be indefinitely deferred by malware, a BWC cannot be relied on to start a recovery process if malware did compromise the RT. The BWC has fewer security features than other Watchdog Counters in this specification and is not cyber resilient.

End of informative comment

6.4.2.3 Latchable Watchdog Counter (LWC)

Start of informative comment.

A Latchable Watchdog Counter (LWC) is a Watchdog Counter that, while enabled, cannot be disabled, or deferred, until it generates an Attention Signal, or it receives an Initialize Signal. Thus, a RE can use an LWC to ensure that control returns to it periodically even in the presence of malware. In contrast to a BWC that only fires when the device is hung (assuming the BWC is regularly deferred during normal operation), the LWC triggers during normal device execution, which may interfere with the device's normal operation.

End of informative comment

6.4.2.4 Authorized Deferral Watchdog Counter (ADWC)

Start of informative comment.

An Authorized Deferral Watchdog Counter (ADWC) is a Watchdog Counter that can be deferred with authorization. In contrast to a BWC that can be indefinitely deferred by malware, an ADWC can be configured so it can only be deferred if the RA allows it, even when malware actively tries to prevent control from being transferred to the RE. When the RA decides that a device is not compliant (e.g., displays aberrant behavior or fails to install an update), then it stops authorizing deferral. In the absence of the deferral, the ADWC generates an Attention Signal which can cause a Module Reset and give the RE an opportunity to perform any necessary servicing.

End of informative comment

6.4.2.5 Wakeup Watchdog Counter (WWC) and waking up from Low Power States

Start of informative comment.

Devices often support low power states, potentially ranging from completely off through any number of states that lower power consumption. If malware does gain control of a RT, malware could attempt a Denial-of-Service (DoS) attack by putting a CRM or an entire device into a power state that is unresponsive. If a RE relies on an LWC or an ADWC that has a dependency on the power state of a device or its CRM to generate its Attention Signal, something needs to prevent the RT from controlling the power states or some other mitigation is necessary.

One simple solution is to rely upon a human operator of the device to notice the device is stuck in a low power state and use manual actions (e.g., a physical button or some other means) to power cycle the device, forcing the RE to gain control. For example, by unplugging a device and plugging it in again.

A second solution could be for the architecture of the power management of a device or a CRM to limit the amount of time allowed in a low power state and have the RE gain control following exit from the low power state. With such an architecture, the upper bound on the time possible in the low power state would mitigate the risk of the RT using low power states to make a device or CRM persistently unresponsive.

A third solution is to make the functioning of an LWC or an ADWC independent of power states the RT controls. For example, an LWC could be powered by a source which is independent of the RT and able to generate its Attention Signal that causes the RE to gain control even if the RT changes power states. An LWC or an ADWC whose functioning cannot be interfered with by the RT's ability to do power management is called a Wakeup Watchdog Counter (WWC).

A BWC cannot serve as a WWC because malware with control of the RT could easily disable or alter a BWC prior to entering a low power state.

Power state management is often a complex aspect of devices with diverse implementations. The solutions described above to mitigate the risks of the RT attempting to manipulate power states to cause a DoS attack are only examples and other solutions are feasible.

Depending on the architecture of a device, it may have multiple independent CRMs, each with their own power states and power management. Simpler architectures may consist of multiple CRMs that share a common set of power states and power management. Depending on the architecture of a device, a single mitigation against any RT on the device putting the device into an unresponsive low power state could be sufficient. On more complex devices, different CRMs might need their own individual mitigation against DoS attack attempts if their RT is compromised.

End of informative comment

7 Design Considerations

7.1 Unnecessary Attention Signals

Start of informative comment.

Watchdog Counters and other sources of Attention Signals may themselves disrupt device availability by generating unnecessary Attention Signals periodically (in case of an LWC) or occasionally (in case of an ADWC, when a Deferral Ticket cannot be obtained) when no servicing of the RT needs to be performed. (The action to perform Module Reset and run the RE will disrupt the RT.) Therefore, the cyber resilient Watchdog Counters described in this specification will not be appropriate for all use cases. When Watchdog Counters are implemented, their expiration values should be chosen to balance the recovery latency with the operational requirements of the device (tolerance to its service interruptions). If cyber resilient Watchdog Counters are inappropriate, then Device Vendors may consider employing alternative designs to generate Attention Signals beyond those described in this specification or may rely on human interaction.

Apart from the interruption to the RTs availability, generating an Attention Signal that results in a Module Reset is considered benign.

End of informative comment

7.2 Loss of Connectivity

Start of informative comment.

Reliance on control via a network by a remote RA entails exposure to network faults. The inability of a RE to communicate with its RA constitutes itself a special kind of degraded state. It is special because instructions or assistance from the RA are unavailable to the RE. For example, a healthy RT may also be unable to retrieve Deferral Tickets from the RA to defer an ADWC.

A simple way for a RE to react to a lack of network connectivity is to put the device in a stand-by (or quarantine) mode in which the device is prevented from performing its primary functions (i.e., the RT is not executed) until the RA becomes available again. This behavior will not always be appropriate: depending on the kind of device responsibilities, the RE behavior (and, if necessary, the RT design) can be modified to avoid or, at least, mitigate unwanted device service interruption. In some cases, the device may be allowed to continue to operate with unrestricted functionality. Alternatively, the RT may be run in a special mode with:

- limited functionality of the device's operational logic,
- limited exposure to the external world,
- a special alternative of the device's operational logic (e.g., in case of a cloud-managed traffic light – all lights flashing red in the United States as a signal for drivers to stop at the light).

In some designs, the RE may be given a priori instructions and authority regarding actions to take without the need to communicate with the RA prior to remediation. In such designs, secure logging of actions taken should be performed by the RE and communication of those actions to the RA should be undertaken once connectivity is restored.

Other designs are possible. For example, in a design where occasional loss of connectivity is anticipated, the design could allow a physically present person to press a button that delays the generation of Attention Signals by Watchdog Counters.

End of informative comment

8 Threat Model

Start of informative comment.

The threat model of this architecture is built around four entities: The RE, RT, RA, and RO. It includes the following non-malware related threats:

- Powering off the device because of temporary external power loss.
- Powering off the device because of an intermittent failure at any point of the RE or RT execution.
- Powering off the device because of a fault in the RT caused by a bug in its implementation.
- Hanging the device because of an intermittent failure at any point of RE or RT execution.
- Hanging the device because of a fault in the RT caused by a bug in its implementation.

The primary malware-related threats addressed by this cyber resilient architecture are:

- Networked entities subverting the RT.
- A compromised RT attempting to subvert the RE by powering off a subcomponent or the entire device.
- A compromised RT generating an Initialize Signal to Watchdog Counters or Protection Latches internal to the CRM without a corresponding Module Reset.
- A compromised RT preventing the propagation of an Attention Signal or an Initialize Signal between a source and destination.
- A compromised RT preventing the RO from completing a Module Reset after receiving an Attention Signal.
- Networked entities attempting to subvert the RE directly.
- Man-in-the-middle attacks (including replay) from stockpiling, reusing, or tampering with Authentication values for deferring or reconfiguring ADWCs (specifically including cases when authentication values may traverse a compromised RT).

The following threats are out of scope for this specification:

- Attacks on the RA
- Attacks involving physical access to the device (though Device Vendors are encouraged to consider physical threats in their design).

The threat model comprises the following security boundaries.

- Between the RE and RT that coexist locally on the same device, possibly sharing some or all their hardware. The RT is considered vulnerable to attacks and thus unreliable at any time. The SEE and the functioning of the RE are required to be isolated from any interference by the (potentially compromised) RT.
- Between a RA and REs that it controls. Within the scope of this specification, the RE must trust the RA as it has no means of evaluating its integrity. Thus, any information confirmed to have been received from the RA is trusted. Methods of verification could be (a) using properly secured channels between the RE and the RA, (b) using untrusted channels (e.g., through the RT) with some other mechanism to validate authenticity (e.g., data signed by the RA with rollback protection), or other schemes.
- For the signals, the security boundary encompasses the source, propagation, and the destination, serving to protect the signals from interference from other components.
    - The generation of Initialize Signals for Watchdog Counters and Protection Latches internal to the CRM (including the exclusive ability to generate an Initialize signal).
    - The generation of Attention Signals by Watchdog Counters internal to the CRM
    - The propagation of Initialize and Attention Signals from source to destination.
    - Receipt of signals by a destination (if the destination intends to process the signal).

This specification neither mandates nor precludes any specific security design of a RA.

Connectivity loss is acknowledged as a legitimate threat. Since the reaction to such loss is highly dependent on the nature of a device's responsibilities, this specification only provides general guidance in this regard (see section 7.2).

End of informative comment

9 Cyber Resilient Building Block Requirements

Start of informative comment.

This section contains requirements for CRBBs. It augments the informative introduction to CRBB capabilities in section 6.

End of informative comment

9.1 Secure Execution Environment (SEE)

Start of informative comment.

A SEE is a CRBB that prevents computation that occurred in the past or may occur concurrently from affecting the initial conditions or computation in the environment.

This specification uses Temporal Protection to protect the RE from the RT. The SEE is used to ensure that the RE is executed without interference from the RT. The Module Reset event establishes the SEE and passes control to the RE before the RT. The RE code is executed, possibly using the same hardware resources as the RT, completes all its resilience related activities, applies all the necessary protections to its resources and information, and only then passes control to the RT.

Temporal Protection contrasts with Spatial Protection in which a SEE provides isolation of the RE from the RT that allows both to run at the same time. Though only requirements for Temporal Protection are in scope for this specification, both variants of a SEE can provide adequate protection for the RE from the RT. For example, devices that instead rely on physical isolation of the RE from the RT could have Attention Signals that result in guaranteed execution of the RE without triggering a Module Reset.

End of informative comment

9.1.1 Module Reset Established SEE used for Temporal Protection of the RE from the RT

Start of informative comment.

The approach adopted by this specification is Temporal Protection of the RE from the RT after a Module Reset. At a conceptual level, this method ensures protection of the RE by requiring that the SEE is established, the RE is executed and then only after the RE is finished does the RT start to execute. Temporal Protection of the RE from the RT also requires any resources that could be used before the RE executes to have the same protections from the RT as the RE.

A Module Reset is required for CRMs that provide Temporal Protection of the RE from the RT. Module Reset establishes a SEE to protect the RE from the RT regardless of the prior execution states of the device and anything that could occur concurrently after Module Reset, but before RE execution completes. Establishment of a SEE will typically involve resetting internal registers and caches. The SEE needs protection from all resources the RT can control directly or indirectly.

Some implementations perform Module Reset after power on. Some implementations use a warm restart to perform Module Reset. When a CRM is a subcomponent of a device, an implementation could perform Module Reset through warm restart of the component. Module Reset can be initiated both voluntarily (e.g., by device hardware, software, a request from the RT, etc.) and involuntarily (e.g., when a Watchdog Counter expires).

End of informative comment

9.1.1.1 The RT Domain and SEE Resources

Start of informative comment.

The RE might need to use resources and information. If the resources or information the RE uses can be modified by the RT directly or indirectly, the RE needs to validate the resources and information before using them. To distinguish what resources and information the RE needs to validate before use, resources and information are

divided into two mutually exclusive security groups call the “RT Domain” and “SEE Resources.” The term “RT Domain” is used to refer to the RT and resources and information the RT can modify or control directly or indirectly. Resources and information the RT cannot modify or control (directly or indirectly) are referred to as “SEE Resources”.

After Module Reset and SEE establishment, the SEE needs to protect the RE from all resources in the RT Domain, including hardware resources. In this context, resources and information need not be included in the RT Domain if they are (1) not part of the RT and (2) the ability for the RT to modify or control them can be suspended after Module Reset, until the RE completes its functions. Implementers can decide to place resources and information in the RT Domain for their convenience. Example benefits of assigning resources or information to the RT Domain are (1) the implementer does not need to analyze whether the RT can directly or indirectly influence or control the resource or information and (2) it can be beneficial to minimize the size of the SEE Resources.

This specification does not constrain how SEE Resources might access each other or influence computation in the SEE. An implementation might impose constraints on SEE Resources accessing each other or their ability to access the SEE.

End of informative comment

9.1.1.2 Launching and Running RE capabilities within a SEE

Start of informative comment.
An initial component or capability launched in the SEE needs to come from or be intrinsically protected by SEE Resources because it cannot be validated. A simple example diagram of this concept is shown in Figure 4.

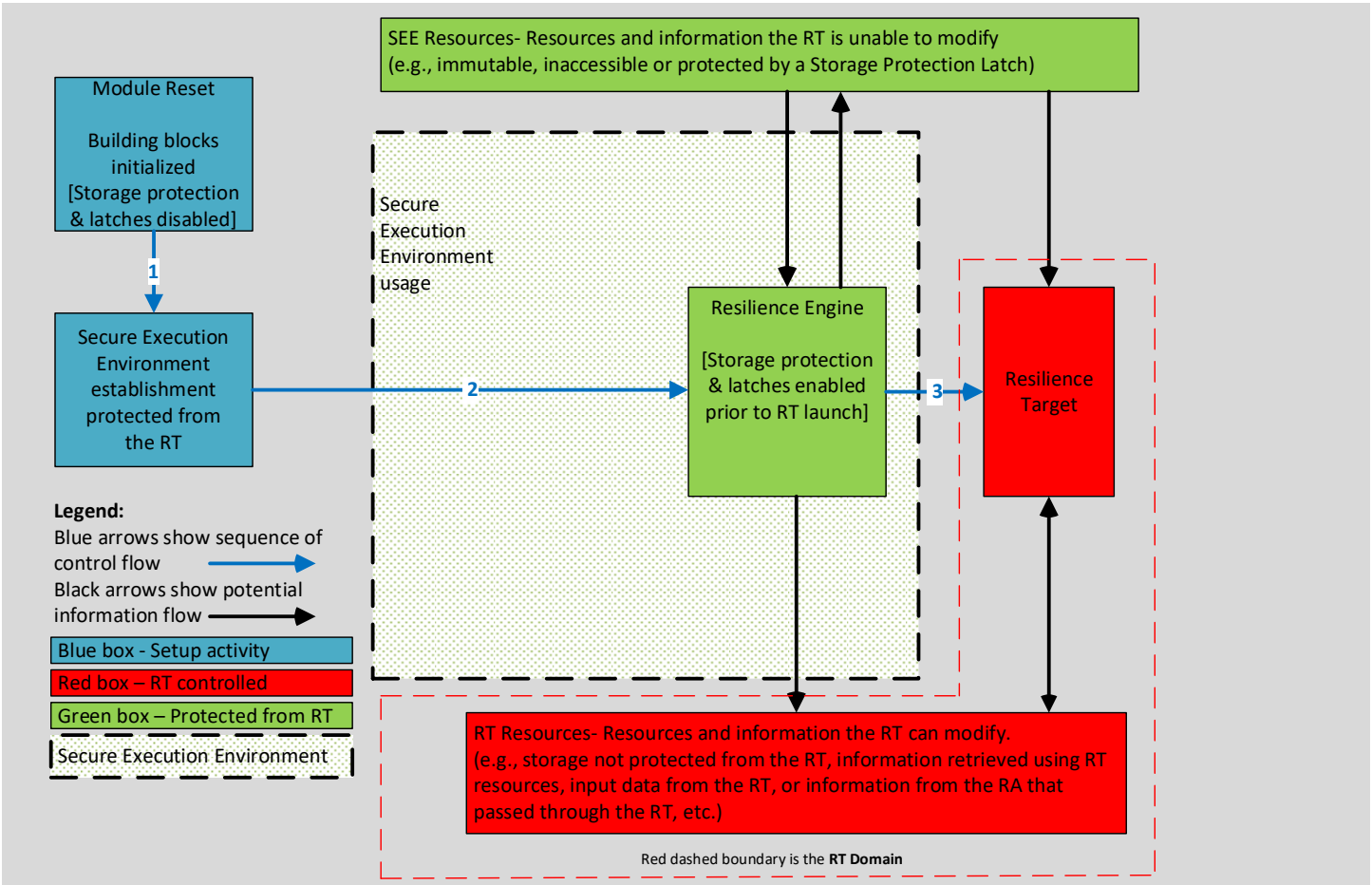

Figure 4: Simple Boot flow when using a SEE with Temporal Protection of the RE from the RT

This example has two classes of resources:

(1) SEE Resources that the RT cannot modify directly or indirectly. The SEE Resources could store the RE itself or other important data. The SEE Resources that the RT cannot modify could still be updated by information that passes through the RT Domain, but only after validation. An example of validation is checking that code or data has been signed by the RA.

and

(2) Resources and information in the RT Domain that the RT can modify (directly or indirectly) are labeled RT Resources. The RT Resources could be available for the RT to store its information or reprogram modules under its control.

The flow of control in Figure 4 starts in the top left where Module Reset occurs and causes the storage protection latches to be disabled, and then transitions to the next blue box where a SEE is established for Temporal Protection of the RE from the RT. The RE software is loaded from SEE Resources and the flow of control transitions to the RE executing in the SEE. Since the RE consists of resources protected from the RT, the RE resources can be used without any validation. This specification focuses on just protecting the RE from the RT (directly and indirectly), so other elements in the device are not prohibited by this specification from changing the contents of the SEE Resources. For this reason, an arrow is shown indicating that the RE can modify the SEE Resources. More specifically, the RE could freely modify the contents of the SEE Resources after Module Reset and prior to launching the RT. If the design uses a storage protection latch to protect its SEE Resources from the RT, the latch needs to be enabled prior to launching the RT, in order to prevent the RT from directly modifying SEE Resources. To prevent

the RT from indirectly modifying SEE Resources, the RE also needs to enable storage latch protections before the RE uses information without validation from the RT Domain or invokes resources from the RT Domain without validation. This is because a compromised RT could tamper with any of the resources in the RT Domain, and they need to be treated at the same trust level as the RT itself.

End of informative comment

9.1.1.3 SEE or RE access to the RT Domain

Start of informative comment.

By definition, the RE needs to be able to service the RT. To do this the RE needs a secure way to access and modify the RT.

Anything running in the SEE may write to the RT Domain. For example, the RE may need to modify the RT hardware, code or data stored in the RT Domain for recovery or patching scenarios. For this reason, in Figure 4 a downward arrow is shown going from the RE to the RT Domain, indicating the RE is writing to the RT Domain. Figure 4 also shows information being read from SEE Resources by the RT. This is acceptable because the RT is not able to write to the SEE Resources. However, some implementations may want to protect the confidentiality of SEE Resources from the RT. In those cases, the Storage Protection Latch for the SEE Resources could provide both read and write protections and the arrow from the SEE Resources to the RT would not exist. Finally, Figure 4 also shows the RT writing and reading RT Resources.

This specification only protects the RE and its dependencies from the RT. This means any information loaded from the SEE Resources and used in the SEE is protected from the RT and can be written back to the SEE Resources with or without SEE or RE validation.

End of informative comment

9.1.1.4 SEE or RE usage of data acquired from RT Domain

Start of informative comment.

It is acceptable for the RE to retrieve information or use resources from the RT Domain if the information or resources can be fully validated by the RE, possibly via components running in the SEE. The SEE provides a safe execution environment for the RE, but it does not prevent the RT from influencing RE behavior if the RE uses RT Domain resources or information. The RE is always responsible for careful validation of any input or resources from the RT Domain.

Device Vendors may optionally allow RAM contents to survive Module Reset. This allows the RE to work cooperatively with the RT and cyber resilient Watchdog Counters. For example, the RT may quiesce normal operations and voluntarily invoke the RE through a programmatic Module Reset at a convenient time. If the RE determines that the device is healthy, it can re-configure the Watchdog Counter and resume the suspended RT, thereby avoiding a longer boot process.

Some examples of the RE accessing the RT Domain include:

- The RE might read RT configuration data from storage or unprotected memory locations in the RT Domain to detect compromise or corruption of the RT. The RE could validate the RT configuration data to determine whether servicing actions are required.
- The RE might use an unprotected network interface to download new servicing instructions, such as recovery software or policies, from the RA. The RE would verify that the downloaded instructions are signed by the RA before allowing it to run or be consumed in the SEE.
- Some RE implementations may rely on the RT to cooperatively transfer patches or instructions from the RA to the RE under normal operation, without immediately interrupting the RT by performing a Module Reset. To support the RT passing deferred or delayed RA issued information to the RE, it may be desirable for there to be a way for the RT to pass input data to the RE by allowing some RT Domain data to persist

across a Module Reset. This might be accomplished by allowing the RT to place RA issued data in the RT Domain (e.g., persistent storage or volatile memory), allowing this data to persist across a Module Reset, and performing a Module Reset during a scheduled downtime window. The RE will check the information downloaded is signed by the RA before allowing it to run or be consumed in the SEE.

End of informative comment

9.1.1.5 Sequence of RE execution relative to other actions in the SEE

Start of informative comment.

The RE is a logical component. It does not need to be the first code launched in the SEE. Any number of earlier components could run before the RE. The only restriction is that the RE is required to run before the RT or any resources from the RT Domain.

Figure 5, below, shows a more complex example of a boot flow: After the flow of control arrow labeled "2" a new initial component is shown being launched in the SEE along with any number of additional component layers launched before the RE. Figure 5 also shows the code in the SEE retrieving information from the RT Domain and validating it before use.

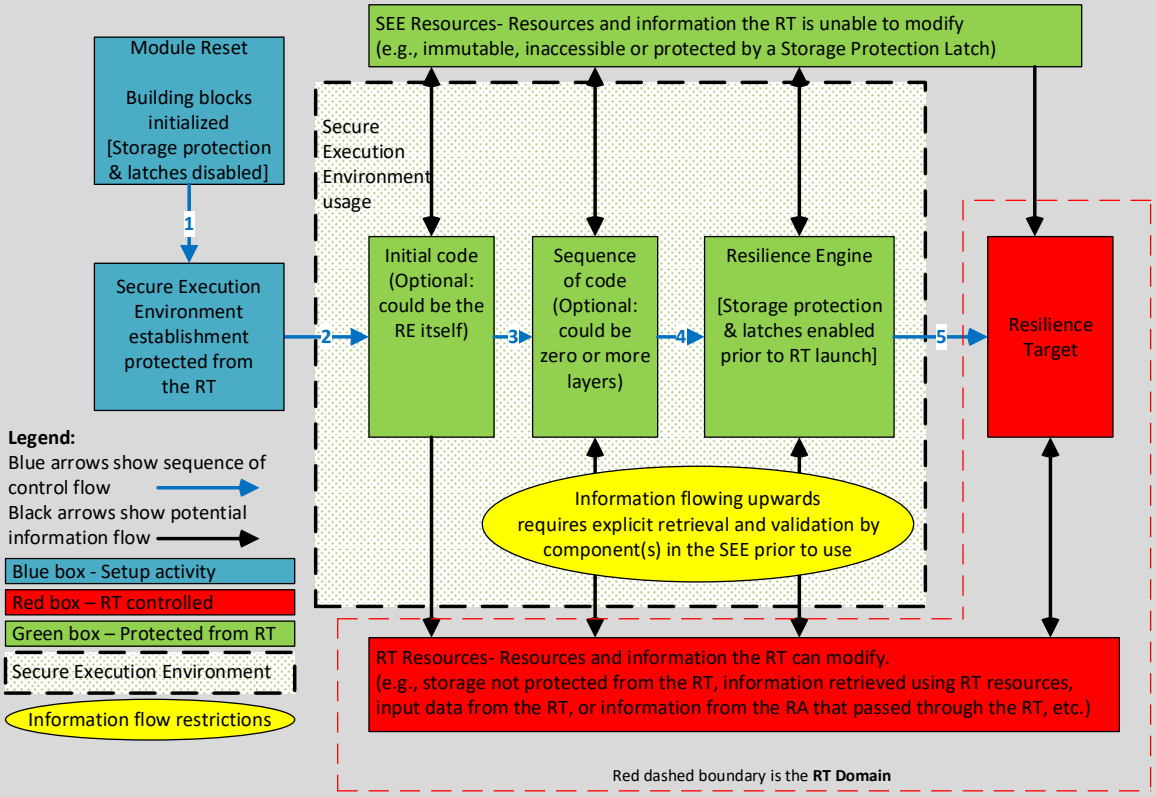

Figure 5: Complex boot flow when using a SEE with Temporal Protection of the RE from the RT

End of informative comment

9.1.1.6 Preventing the RT Domain from accessing protected SEE or RE resources

Start of informative comment.

Device designs need to prevent any direct or indirect way for the RT to interfere with its RE. For example, it cannot be possible for the RT to directly interrupt its RE or for the RT to reprogram another component that subverts its RE indirectly. (For example, designs need to prevent the RT from reprogramming a component that has bus mastering capabilities that might modify the RE runtime code.) An implementation could prevent the RT from communicating with the other component or, alternatively, implement Module Reset to force bus mastering devices to be inactive until the RE or another component enables them safely.

Protections for SEE Resources can be implemented by immutability, physical or logical isolation, suspending resources (e.g., turning off a bus mastering device, quiescing and holding a processor in reset), denying access to resources (e.g., not providing a CPU needed to execute software), enabling a storage protection latch, etc. The protection needs to be continuous, but it might be achieved by initially denying the RT access to a resource like a processor after Module Reset, then transition to protection by enabling a storage latch before providing a processor to execute RT software.

An example of using different protection mechanisms at different times is shown in Figure 6 below. Figure 6 shows a timeline of activity following a Module Reset, going from left to right:

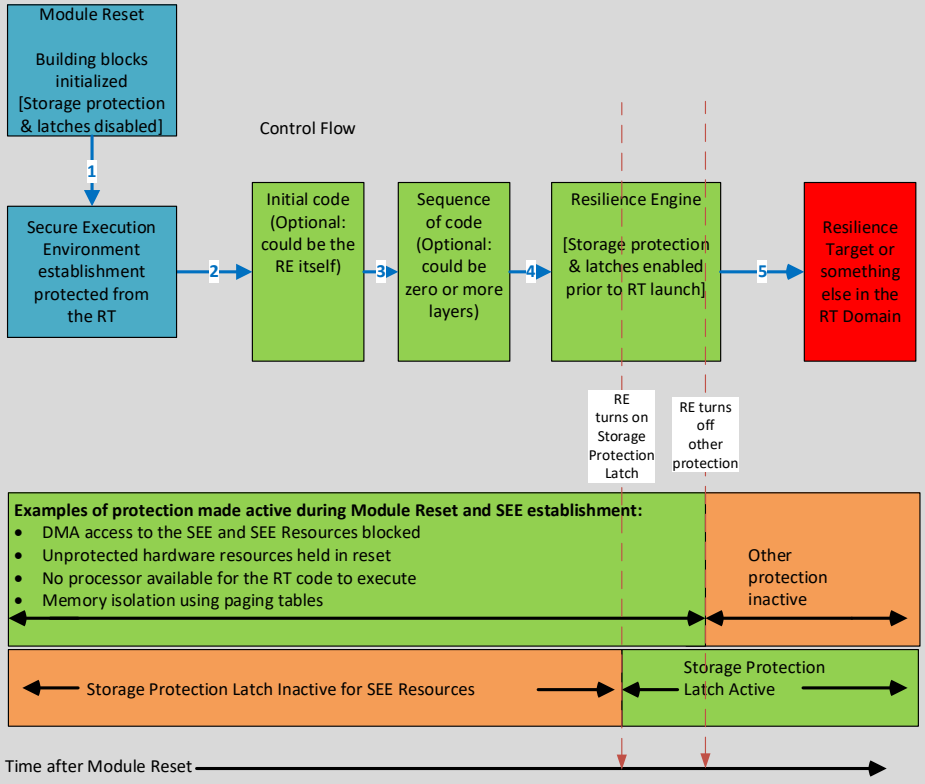

Figure 6: Continuous protection example using a Protection Latch prior to launching the RT

At all times during the sequence in Figure 6, starting with Module Reset and eventually ending with transfer of control to the RT, the RT is unable to modify or access the SEE Resources:

1) Immediately following the Module Reset, implementation-specific mechanisms protect the SEE Resources that will be used by the RE.
2) Sometime later, after the RE starts, the RE turns on a Storage Protection Latch that provides Write protection for the SEE Resources.
3) After establishing the write protection, the RE or SEE can disable the implementation-specific protections and subsequently launch the RT or something else in the RT Domain.

For the RE to properly arm protections for SEE Resources, and isolate the RT (and RT Domain) from the RE and SEE Resource, the RT cannot be allowed to interrupt the RE. The RE needs to control its availability until it completes its function. The RE also needs to access its SEE Resources and communicate securely with them. This does not mean the SEE Resources need to be available all the time, just that their availability and ability to provide software, data or capabilities to the SEE needs to be under the control of the RE or other resources that operate in the SEE after Module Reset.

The RE should be able to distinguish between a Module Reset event and something else (e.g., hardware, software, etc.) simply transferring control to the RE's entry point. For example, if control is transferred to the RE without Module Reset occurring, the RE might notice that the Storage Protection Latches are already activated.

After a Module Reset, there is no requirement the RT is invoked. However, there is a requirement that after a Module Reset, the RE needs to be invoked before anything in the RT Domain (including the RT itself). Any servicing actions the RE needs to perform on the RT could be delayed indefinitely as long as nothing in the RT Domain is invoked.

Despite the examples above in Figure 4 and Figure 5 showing the RT outside of the SEE, the RT can execute in the SEE provided the RE has completed and protections are in place for SEE Resources. Depending on the implementation, SEE Resources can be protected persistently from the RT Domain or can rely on some form of continuous protection as illustrated in Figure 6.

End of informative comment

9.1.1.7 Ensuring SEE and RE resource availability

Start of informative comment.

Implementers should be careful to consider availability of the RE. For example, if an implementation was incorrectly designed to store a critical portion of the RE in the RT Domain and depended on retrieving and validating the critical portion of the RE from the RT Domain, that could create an opportunity for the RT to perform a DoS attack on the RE by deleting the RE information stored in the RT Domain.

End of informative comment

Requirements for using Module Reset to establish a SEE for Temporal Protection of the RE from the RT:

1) Protection for the SEE Resources from the RT Domain:
   a) The integrity and availability of the SEE resources SHALL be continuously protected.
   b) The confidentiality of the SEE Resources MAY be protected.
2) Protection for the SEE and SEE communications from the RT Domain until the RE completes:
   a) The integrity and availability of the SEE and SEE communications SHALL be continuously protected.
   b) The confidentiality of the SEE and SEE communications MAY be protected.
3) The effect of Module Reset:
   a) The SEE SHALL be established.
   b) All CRBBs internal to the CRM SHALL receive an Initialize Signal (see section 10.4).
   c) Module Reset SHALL be the only way to send Initialize Signals to the CRBBs internal to the CRM (see section 10.4).
   d) The cause of Module Reset (e.g., a power-on, a Watchdog Counter Attention Signal, programmatic request, etc.) SHOULD be recorded and accessible from inside the SEE.
4) SEE Usage:
   a) If the RE is invoked,
      i) The RE SHALL be invoked in the SEE.
      ii) The RE SHALL be invoked before anything in the RT Domain.
   b) The SEE and SEE Resources SHALL validate all data and information retrieved from (or provided by) the RT Domain
   c) After a Module Reset, code that runs in the SEE SHOULD be able to retrieve input data from the RT Domain.

9.2 Signals

Start of informative comment.

This section contains requirements for Signals.

There are no requirements for Programmatic Interfaces.

End of informative comment

9.2.1 Attention Signals

Start of informative comment.

An Attention Signal is an output signal for attention generated by a component (e.g., a Watchdog Counter elapsing or the press of a physical button). It may be unnecessary for a destination to act on an Attention Signal. For example, when an Attention Signal is used outside of the context of a CRM, a destination asleep in a low power state or busy dealing with a higher priority issue that needs to be addressed to prevent a thermal overload could completely ignore an Attention Signal intended to dim a screen because of user inactivity. There are requirements in section 10.4 that describe how an RO is required to act upon Attention Signals it receives from a building block internal to its CRM.

End of informative comment

Requirements for Attention Signals:

1) A CRBB Attention Signal output SHALL be deterministic and discoverable.

9.2.2 Initialize Signals

Start of informative comment.

An Initialize Signal is a signal used to securely convey security relevant events, like power-on, restart, etc., which reset or reinitialize a CRBB.

An Initialize Signal received by a destination may result in a lower security configuration (e.g., by turning off write or read protection for storage or disabling a Watchdog Counter). For this reason, Device Vendors need to carefully consider the sources able to generate Initialize Signals. More guidance on orchestration of Initialize Signals is in section 10.4.

Device requirements are outside the scope of this specification so the requirements in this section focus on what requirements can be implemented at the scope of individual CRBBs. It is recommended that only sources designated by the Device Vendor are capable of generating an Initialize Signal and CRBBs only accept Initialize Signals from sources designated by a Device Vendor.

End of informative comment

Requirements for Initialize Signals:

1) A CRBB SHALL perform its initialization function upon receipt of an input it interprets as an Initialize Signal.

9.3 Protection Latches

Start of informative comment.

Section 6.3 provides an overview of storage protection mechanisms applicable to CRM designs and discusses the use of Protection Latches. This section contains requirements for an abstract basic building block called a Protection Latch which prevents storage read and/or write actions when active. For readability, this section uses the word “latch” interchangeably to refer to the term Protection Latch. A latch prevents read or write actions to provide protection for stored data.

A latch can be activated programmatically. While a latch is activated, it prevents read and/or write actions on storage. A latch cannot be programmatically deactivated. A latch uses an external reset as the security mechanism to control how the protections are turned off. The external reset is communicated to a latch through an Initialize Signal.

A Protection Latch is a building block that is intended to be used in a wide variety of settings. The term "external," as in "external reset," means outside of the boundary of a Protection Latch. "External" is not intended to imply external to the device or component that includes a Protection Latch. It is outside the scope of this section to dictate what controls the external reset mechanism.

End of informative comment

9.3.1 Protections for Storage

Start of informative comment.

When active, a Protection Latch completely blocks actions (read, write or both) on storage, protecting the stored data from disclosure (read blocked), modification (write blocked) or both. Blocking reading prevents read access of the data in the storage. Blocking writing prevents modifying the data in the storage. Protection occurs when a latch is active and does not occur when a latch is inactive.

End of informative comment

9.3.2 Initial State

Start of informative comment.

The initial state of a latch is inactive, which means it does not protect storage. The inactive state means read and write actions on storage are allowed insofar as a latch is concerned. Something other than a latch could block read or write actions.

Note: The initial state of Protection Latches in this specification is different than some other TCG storage technologies. For example, after provisioning, self-encrypting drives need to be unlocked before their contents are accessible. In contrast, Protection Latches need to be activated before they provide read/write protection for their storage.

End of informative comment

9.3.3 Active State and Protections

Start of informative comment.

A latch can be transitioned to its active state programmatically (by code). When a latch is active it completely blocks storage actions of read, write or both, providing storage protection.

End of informative comment

9.3.4 Latch Reset

Start of informative comment.

An external Initialize Signal (versus a direct programmatic method for disabling) is the only way for a latch to be reset to the inactive state.

End of informative comment

9.3.5 Permanent or Reconfigurable Storage Protection

Start of informative comment.

Protection Latches may be implemented permanently bound to one or more storage locations, or with the ability to be configured programmatically.

An example where it could be useful to allow expanding the region of storage protected against write actions is an untrusted environment appending an error log. After a write to storage that appends a new log entry, an untrusted environment could dynamically increase the size of the region of storage with write protection to include the new entry. This kind of strategy would incrementally protect log entries up until the point in time the environment was compromised, and it would prevent earlier log entries written prior to compromise from being erased.

Care should be taken when implementing reconfigurable Protection Latches to prevent malware from expanding protection regions in ways which could create device issues such as a DoS. One potential mitigation of this concern would be preset limits for regions of protected storage to prevent unbounded expansion.

End of informative comment

9.3.6 Overlap

Start of informative comment.

More than one Protection Latch could protect the same storage address. If more than one Protection Latch covers a storage location, protection is activated by the Logical OR of the covering Protection Latches:

- If code attempts a read action on storage, any active latch blocking read will prevent read.
- If code attempts a write action on storage, any active latch blocking write will prevent modification.

End of informative comment

9.3.7 Indication of Blocked Actions

Start of informative comment.

When blocking occurs, the result is Device Vendor defined. For example, the result of a blocked action could silently ignore an attempted write, silently return arbitrary data from a read request, return an error, etc. Because blocked actions do not have any beneficial outcome, they could be an indication something is wrong. It is preferable for the implementation to return an error or provide some other discoverable failure indication in a Device Vendor defined manner.

Code might be able to determine whether a read or write action is blocked. A Device Vendor could provide interfaces to determine whether a latch is active or not, and what storage is protected by a latch. Alternatively, a Device Vendor could allow code to attempt a blocked action and analyze the return result or look for an indication of blocking.

If a Protection Latch is blocking reads, in no circumstance can the indication of a blocked action return the actual data from the protected storage location.

End of informative comment

9.3.8 Protection Latch Abstract Diagram Example

Start of informative comment.

Figure 7 illustrates Protection Latch implementation choices that could be helpful for a reader to better visualize Protection Latch interfaces and state transitions. Optional items are indicated in italics text followed by an asterisk. Table 1 describes the meanings of the different Protection Latch states. Table 2 describes the meaning of the different actions associated with state transitions of a Protection Latch. Table 3 describes configuration parameters and values that an implementation might use to represent how a Protection Latch is configured.

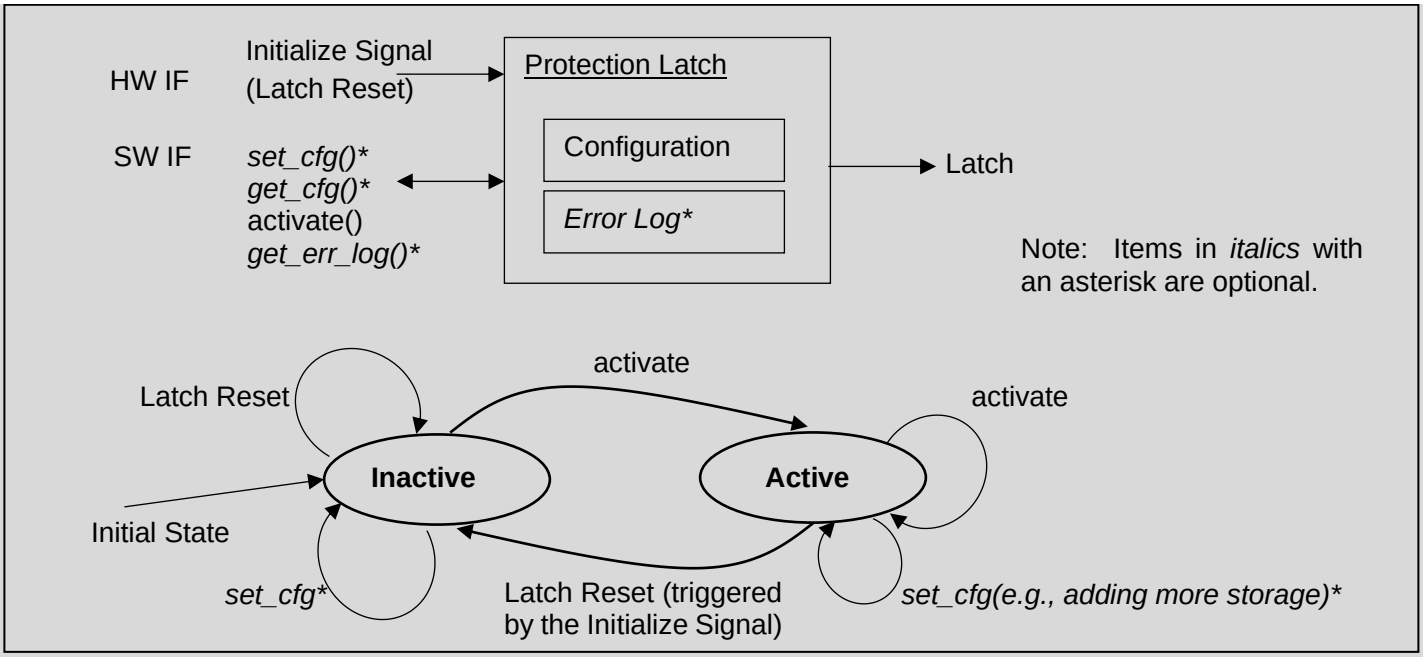

Figure 7: Protection Latch block and state Diagram

State	Description
Inactive	Access is allowed to the storage regions associated with the latch. All APIs are enabled. Can set configuration.
Active	Access is blocked to the storage regions associated with the latch. Configuration changes are limited to adding protection to additional storage regions (but not removing protection from any).

Table 1: Protection Latch States

Action	Description
Initialize Signal	Reset Latch and transition state to inactive.
Activate	Transition state to active.
set_cfg	Change the configuration of storage locations the latch protects and/or other latch settings.

Table 2: Protection Latch State Transition Actions

Parameter	Value	Description
Addr Regions	1 … n	Defines the number of regions that can be access controlled
Region Start, End	addr_start[1], addr_end[1] addr_start[2], addr_end[2] addr_start[x], addr_end[x]	x = 1, 2, … n
Protection Mode	0 … 2	0: block read 1: block write 2: block read and write
Response to blocked read access	0 … 1	0: deterministic pattern 1: return random data Note: “Ignore” is not an option for blocked reads

Response to blocked write access	0 … 1	0: silently ignored 1: error indication
Logging of blocked actions	0 … 3	0: blocked read 1: blocked write 2: blocked read or write 3: no logging

Table 3: Protection Latch Configuration

End of informative comment

9.3.9 Protection Latch Requirements

A Protection Latch is an access-control mechanism for actions on a storage resource that meets the following requirements:

States

1) A Protection Latch SHALL have protection states of Active and Inactive.

Binding with Storage Locations

2) A Protection Latch SHALL be able to be bound to at least one storage location.
3) If a Protection Latch supports reconfigurability then it SHALL provide an interface to support configuration for binding to storage location(s).
4) A Protection Latch's binding to any storage location SHALL NOT be unbound when the latch is in the Active State.
5) A Protection Latch's configuration SHALL NOT change when the latch is in the Active State with the exception that the bound storage region MAY be increased.

Actions and Control

6) Actions prevented by a Protection Latch SHALL be one of:
   a) Read: The latch SHALL block all Reads of a bound location when Active.
   b) Write: The latch SHALL block all Writes to a bound location when Active.
   c) Both Read and Write: The latch SHALL block all Reads from and all Writes to a bound location when Active.
7) When a blocked Action is attempted, an indication of the block SHOULD be discoverable using a Device Vendor defined mechanism.

Latch Reset and State Transitions

8) Latch Reset SHALL place the Protection Latch in the Inactive State.
9) Latch Reset SHALL be triggered by an Initialize Signal.
10) A Protection Latch SHALL support programmatically transitioning from Inactive to Active.
11) A Protection Latch SHALL only support transitioning from Active to Inactive via Latch Reset.

Querying

12) A Protection Latch SHOULD provide an interface for querying metadata about storage location(s) it protects.
13) If a Protection Latch supports reconfigurability then it SHALL provide an interface for querying metadata about storage location(s) it protects.
14) A Protection Latch SHOULD provide an interface for querying the state of the latch (Active or Inactive).
15) It SHALL be possible for code to query whether a storage location currently has read and/or write protection.

Composition of Protection Latches

16) If more than one Protection Latch is present, the following composition rules SHALL apply:

a) The state and configuration of Protection Latches SHALL be independent. Transitioning one Protection Latch to the Active state SHALL NOT affect other Protection Latches. Binding a storage location to one Protection Latch SHALL NOT affect other Protection Latches.
b) If two or more Protection Latches are bound to the same storage location, then an action on that storage location SHALL be blocked if ANY of the associated Protection Latches block the action.

9.4 Watchdog Counters

Start of informative comment.

For general description of Watchdog Counter variants, their basic functionality, and how they fit into the cyber resilient architecture of this specification, see section 6.4.2. This section describes the various Watchdog Counter types and provides requirements for each.

Even though this document defines multiple types of Watchdog Counters, it does not imply that they always must be separate components. A single Watchdog Counter device may implement behaviors of several Watchdog Counter types and expose them simultaneously via an appropriately designed Programmatic Interface.

End of informative comment

9.4.1 Watchdog Counters and Power States

Start of informative comment.

Modern processors and devices aggressively put subcomponents into low-power states when the device is idle. Some or all processor/device clocks can be slowed or stopped in these low-power states.

In the context of a Cyber Resilient Device, Watchdog Counters can be used to reliably schedule servicing actions by having their Attention Signal result in eventual invocation of a RE. To provide resilience, a RE needs to be invoked when:

- A device malfunctions (e.g., “freezes” or stops responding to a RA).
- Malware gains control of a RT and tries to:
  - delay or prevent servicing actions through DoS attacks, including putting the CRM into low power states, or
  - mimic an uncompromised device.
- Preventative patching or software updates need to be deployed.
- Power loss occurs during an update process.
- Etc.

Ideally, expiration of Watchdog Counters supporting cyber resilient capabilities should be based on a mechanism that is resistant to tampering by malware or device failure. If an enabled Watchdog Counter can detect tampering that could potentially cause it to fail to reliably send an Attention Signal, then Detection of tampering could result in the Watchdog Counter sending the Attention Signal immediately. Implementers are encouraged to provide transparency regarding power management and tamper resistance. Clearly documenting properties and behavior will help adopters adequately address resilience requirements.

End of informative comment

9.4.2 Basic Watchdog Counter

Start of informative comment.

This specification defines a Basic Watchdog Counter (BWC) as a building block that accepts an Initialize Signal and produces an Attention Signal. An Initialize Signal is used to securely trigger reinitialization of a Watchdog Counter. The Attention Signal indicates to something external to a BWC that a counter was enabled, and expiration has occurred.

When enabled, a BWC generates an Attention Signal when its current count reaches its expiration value. The default state for a BWC can be enabled or disabled. While disabled, it can be configured with an expiration value and enabled. After a BWC is enabled, it can be cancelled or reconfigured with a new expiration value. A reconfiguration of a BWC that increases the time or number of events remaining to reach the expiration value is called a "deferral."

Many implementations using a BWC today continually defer the counter when a device is functioning normally. Further, some Device Vendors may elect to have the Initialize Signal not return a BWC to its initial default state, but instead have a BWC initialized to a different expiration value and/or a different enabled/disabled status.

When a BWC receives an Initialize Signal it can change to its default state or change its configuration (current count and/or expiration value) and state (enabled/disabled) in a Device Vendor defined way.

A BWC is useful for increasing reliability, but malware may interfere with it. Malware could disable a BWC or make it irrelevant by setting the maximum possible expiration value. It is also possible the external clock or event upon which a BWC is based can be changed, adjusted, or powered off.

A compromised RT may disable a BWC, so a BWC is not considered a CRBB. The BWC is included in this section for reference and to help explain and contrast the properties of cyber resilient Watchdog Counters. All the other types of Watchdog Counters defined in this section are CRBBs.

This specification does not normatively preclude a BWC being enabled without an expiration value set. It is expected such a condition will be prevented by Device Vendors, avoided by users, etc.

A BWC could use either a time-based clock signal or a non-time-based mechanism. For example, a BWC might trigger after a device sends 500 network packets or experiences a fixed number of sleep/wake cycles. This might be useful in situations where counting of events to trigger an Attention Signal is better protected or more meaningful than an absolute time.

See section 9.4.3 for an example diagram that illustrates a possible design for a BWC implementation.

End if informative comment

Requirements for a BWC:

1) A BWC SHALL have one or both of:
   a) a default expiration value, or
   b) a configurable expiration value.
2) There SHALL be a way to enable a BWC.
3) There SHALL be a way to defer the expiration without disabling it.
4) While enabled, internal or external events or clock ticks SHALL advance the current count towards the expiration value.
5) An enabled BWC SHALL generate an Attention Signal when expiration occurs.
6) A disabled BWC SHALL NOT generate an Attention Signal.
7) There MAY be a way to disable a BWC.
8) A BWC SHALL accept an Initialize Signal.
9) The BWC state and configuration SHALL be deterministic and discoverable
   a) resulting from its receipt of an Initialize Signal and
   b) after its generation of an Attention Signal.

9.4.3 Basic Watchdog Counter Abstract Diagram Example

Start of informative comment.

This section provides abstract diagrams of a Basic Watchdog Counter (BWC) to help visualize how its requirements might fit in an implementation. These abstract diagrams are just examples. Other abstract diagrams based on the requirements in section 9.4.2 are possible. The diagrams provided here are merely illustrative and do not imply

any specific required implementation. Device Vendors are free to build BWCs in whatever way best fits their architecture. Implementations could be based in hardware, firmware, software, or any combination.

Figure 8 is an abstract diagram of a BWC based on the requirements in section 9.4.2. Numbered circles in the diagram refer to the corresponding numbered requirement. For example, number circle 1 refers to requirement #1 in section 9.4.2. Solid lines and yellow numbered circles refer to SHALL requirements. Dotted lines and yellow numbered circles refer to SHALL requirements that have more than one choice for their implementation (e.g., using an internal or an external clock). Dashed lines and the gray numbered circles refer to MAY statements. The blue rectangle encloses resources which are part of a BWC. Lines going into or out of the blue rectangle are external Signals, events, or Programmatic Interfaces.

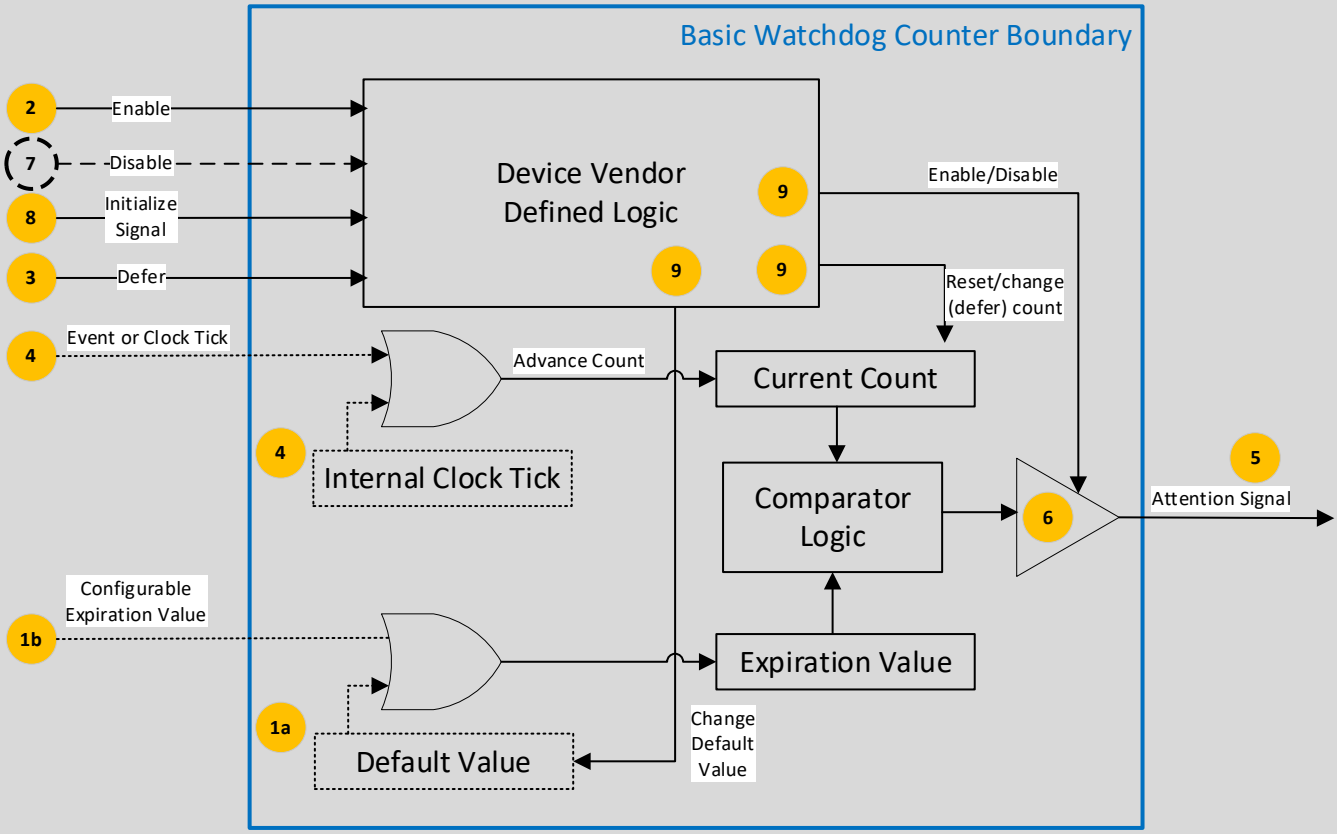

Figure 8: BWC block diagram

Figure 9 is a state diagram for a BWC. For simplicity, the transitions for the Initialize Signal are not shown. The Initialize Signal transition could start and end at either state because the result is defined by the Device Vendor.

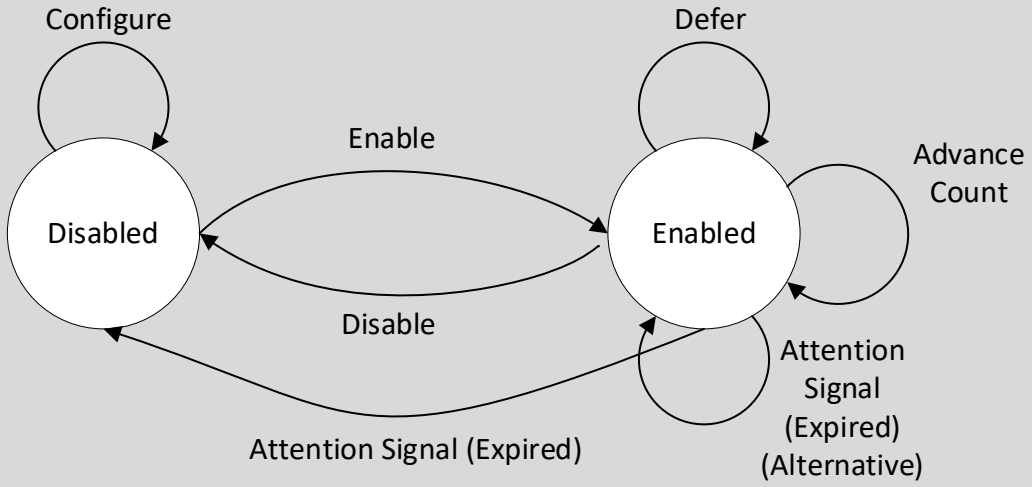

Figure 9: BWC state diagram

End of informative comment

9.4.4 Latchable Watchdog Counter

Start of informative comment.

A Latchable Watchdog Counter (LWC) is a Watchdog Counter that, while enabled, cannot be disabled, or changed until it generates an Attention Signal, or it receives an Initialize Signal.

When an LWC receives an Initialize Signal or generates an Attention Signal, its default state, configuration (expiration value or current count) and/or state can change in a Device Vendor defined way.

The ability to generate the Initialize Signal needs careful protection and needs to be distinct from how Programmatic Interfaces interact with an LWC.

This specification does not support changing the expiration value while an LWC is enabled.

As with a BWC, an LWC could also use a non-time-based mechanism by counting events.

Device Vendors need to provide transparency regarding the protections for the clock or the non-time-based mechanism an LWC is based on, so adopters can determine whether the protections are adequate for their use case.

An LWC can have a default expiration value. An LWC should have a configurable expiration value, so it is flexible enough to use for different use cases. A configurable expiration value is recommended.

See section 9.4.5 for an example diagram that illustrates a possible design for an LWC implementation.

End of informative comment

Requirements for the LWC:

1) An LWC SHALL have one or both of:
   a) a default expiration value, or
   b) a configurable expiration value.
2) There SHALL be a way to enable an LWC.
3) While enabled, the only ways to disable an LWC SHALL be via an Initialize Signal and (optionally) generation of its Attention Signal.

4) While enabled, internal or external events or clock ticks SHALL advance the current count towards the expiration value.
5) An enabled LWC SHALL generate an Attention Signal when expiration occurs.
6) A disabled LWC SHALL NOT generate an Attention Signal.
7) An LWC SHALL accept an Initialize Signal.
8) The LWC state and configuration SHALL be deterministic and discoverable
   a) resulting from its receipt of an Initialize Signal and
   b) after its generation of an Attention Signal
9) While enabled, there SHALL NOT be a way to change (e.g., defer) the expiration (except via receipt of an Initialize Signal or by generating an Attention Signal).

9.4.5 Latchable Watchdog Counter Abstract Diagram Example

Start of informative comment.

This section provides abstract diagrams of a Latchable Watchdog Counter (LWC) to help visualize how its requirements might fit in an implementation. These abstract diagrams are just examples. Other abstract diagrams based on the requirements in section 9.4.4 are possible. The diagrams provided here are merely illustrative and do not imply any specific required implementation. Device Vendors are free to build LWCs in whatever way best fits their architecture. Implementations could be based in hardware, firmware, software, or any combination.

Figure 10 is an abstract diagram of an LWC based on the requirements in section 9.4.4. Numbered circles in the diagram refer to the corresponding numbered requirement. For example, number circle 1 refers to requirement #1 in section 9.4.4. Solid lines and yellow numbered circles refer to SHALL requirements. Dotted lines and yellow numbered circles refer to SHALL requirements that have more than one choice for their implementation (e.g., using an internal or external clock). The blue rectangle encloses resources which are part of the LWC. Lines going into or out of the blue rectangle are external Signals, events, or Programmatic Interfaces. (Note that normative requirement #9 from section 9.4.4 is not shown.)

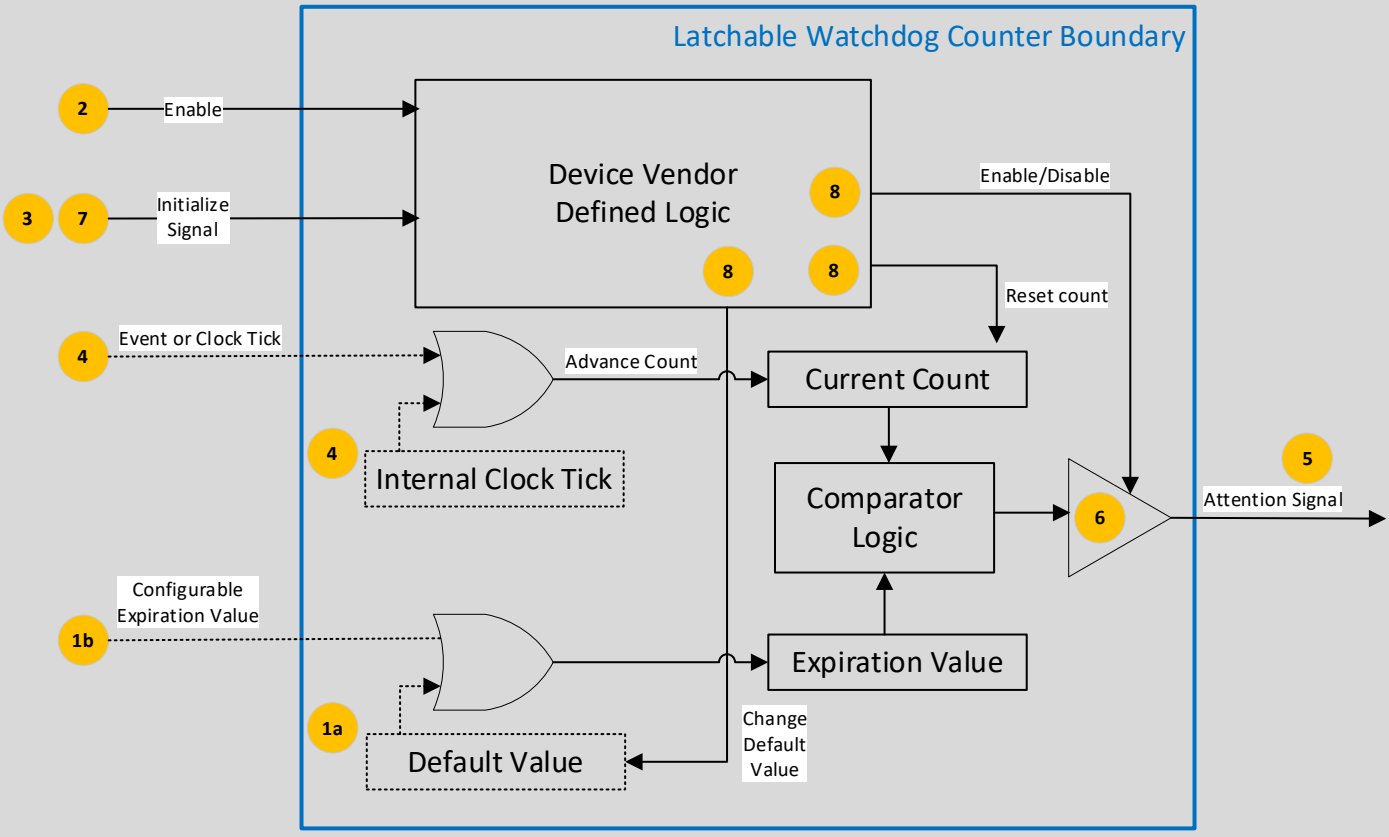

Figure 10: LWC block diagram

Figure 11 is a state diagram for an LWC. For simplicity, the transitions for the Initialize Signal are not shown. The Initialize Signal transition could start and end at either state.

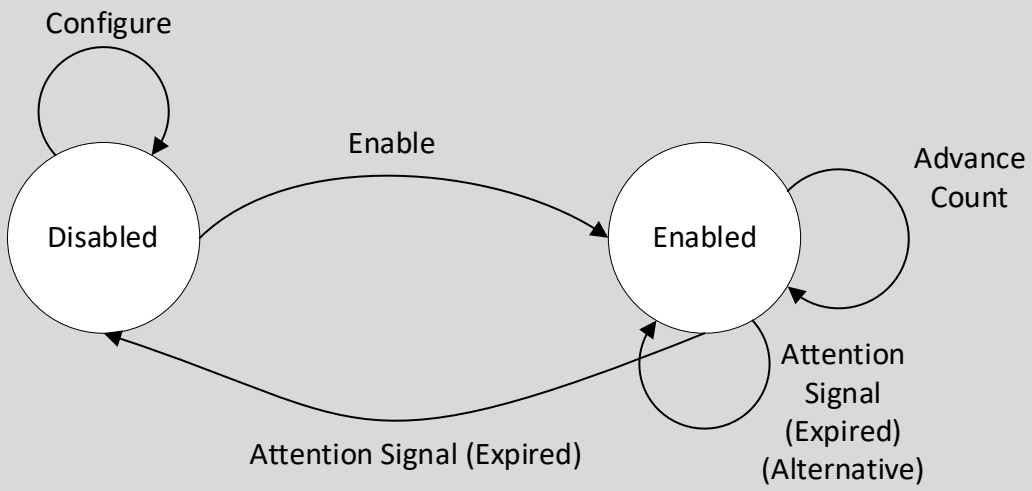

Figure 11: LWC state diagram

End if informative comment

9.4.6 Authorized Deferral Watchdog Counter

Start of informative comment.

An Authorized Deferral Watchdog Counter (ADWC) is a Watchdog Counter that requires authorization to be deferred or reconfigured.

This specification requires any deferral and other configuration changes to be authorized while the ADWC is enabled. The strength of function of cryptographic algorithms and protocols is left up to the Device Vendor, although protection against some common attacks is essential.

The most common usage model for an ADWC is programmatic deferral, whereby the generation of an Attention Signal by an ADWC is repeatedly postponed using a software API and a cryptographic authorization ticket until recovery or servicing actions need to occur. The ADWC verifies that programmatic deferral requests are authorized. The criteria the ADWC uses for deferral authorization is defined in a Deferral Policy. A Deferral Policy specifies the requirements for generating and validating a Deferral Ticket. Depending on the implementation, a Deferral Policy can be predefined or configurable. Deferral can happen repeatedly, each time an authorized deferral request is received by the ADWC.

An ADWC can also support non-programmatic authorizations (e.g., with signals or protected capabilities). For example, an ADWC could receive an input hardware signal authorized by a physically present user who presses a hardware button. The button could generate a hardware signal that authorizes deferral of the ADWC by sending an electrical signal. This could be useful in a scenario where network connectivity is unavailable, but a physically present user wants to postpone a module reset and maintain availability of the device despite the lack of network connectivity.

An example of a Deferral Policy is information that indicates the public key needed to validate a Deferral Ticket. A remote entity possessing the corresponding private key could repeatedly generate Deferral Tickets that postpone the generation of an Attention Signal from the ADWC.

A nonce based design could make the Deferral Tickets single use.

Some applications may want to configure other information about the ADWC using authorization. A more general use policy called the Reconfiguration Policy may be implemented to authorize any other configuration changes including disabling, changing the expiration value, and changes to the Deferral or Reconfiguration Policies. An ADWC can support multiple Deferral Policies and Reconfiguration Policies.

An ADWC can be implemented with or without persistent storage for its configuration, state, and/or Deferral and Configuration Policies. For example, if the ADWC has no persistent storage, the configuration, state, or values may need to be set explicitly each time after the ADWC receives an Initialize Signal.

For implementations that always require authorization to change the configuration, care should be taken to consider how Deferral and Reconfiguration policies will be set initially and how to avoid situations where the authorization credentials are lost, unless that is appropriate for the use case.

An ADWC can have a default expiration value, which may be configurable or not. A configurable expiration value is more flexible and can be altered for different use cases. A configurable expiration value is recommended.

This specification does not specify any specific protocols or cryptographic designs, but the schemes used for deferral and reconfiguration (if implemented) need to be resistant to spoofing, replay and man-in-the-middle attacks. In some scenarios, authentication values for deferral and reconfiguration may pass through a RT that is compromised with a zero-day vulnerability. Implementations need to prevent a compromised RT from stockpiling and reusing Deferral Tickets in the future.

The TPM 2.0 Library Specification [3], Revision 1.59 and later includes a richly configurable Authenticated Countdown Timer which is an example of an ADWC.

End of informative comment

Requirements for the ADWC:

1) Initialization: An ADWC SHALL accept an Initialize Signal.
2) Initialization: The ADWC state and configuration SHALL be deterministic and discoverable
   a) resulting from its receipt of an Initialize Signal and
   b) after its generation of an Attention Signal.
3) Policy: The policy authorizing deferral behavior is called a Deferral Policy, and it SHOULD be configurable.
4) Policy: The policy authorizing reconfiguration behavior is called a Reconfiguration Policy, and it MAY be configurable.
5) Policy: The authorization protocols associated with Deferral and Reconfiguration Policies SHALL be safe from:
   a) spoofing,
   b) replay attacks, and
   c) man-in-the-middle attacks.
6) Policy: The Deferral and Reconfiguration Policies SHALL NOT be configurable without authorization while an ADWC is enabled, or while a Reconfiguration Policy is set. While an ADWC's Reconfiguration Policy is set, the ADWC SHALL require an authorization that satisfies the Reconfiguration Policy for all the following:
   a) disabling the ADWC,
   b) setting/updating the Deferral Policy,
   c) setting/updating the expiration value, or
   d) updating the Reconfiguration Policy itself
7) Enabling/Expiration: There SHALL be a way to enable an ADWC.
8) Enabling/Expiration: While enabled, internal or external events or clock ticks SHALL advance the current count towards the expiration value.
9) Enabling/Expiration: An enabled ADWC SHALL generate an Attention Signal when expiration occurs.
10) Enabling/Expiration: An ADWC SHALL have one or both of:
    a) a default expiration value, or
    b) a configurable expiration value.
11) Deferral: While enabled, the only ways to defer an ADWC SHALL be:
    a) an authorization that satisfies the Deferral Policy,
    b) receipt of an Initialize Signal.
12) Reconfiguration: While enabled, the only ways to reconfigure the ADWC SHALL be:
    a) an authorization that satisfies the Reconfiguration Policy,
    b) receipt of an Initialize Signal.
13) Disabling: While enabled, the only ways to disable an ADWC SHALL be:
    a) an authorization that satisfies the Reconfiguration Policy,
    b) receipt of an Initialize Signal, or
    c) generation of its Attention Signal.
14) Disabled: A disabled ADWC SHALL NOT generate an Attention Signal.

9.4.7 Authorized Deferral Watchdog Counter Abstract Diagram Example

Start of informative comment.

This section provides an abstract diagram of an ADWC to help visualize how its requirements might fit in an implementation. This abstract diagram is just an example. Other abstract diagrams based on the requirements in section 9.4.6 are possible. The diagram provided here is merely illustrative and does not imply any specific required implementation. Device Vendors are free to build ADWCs in whatever way best fits their architecture. Implementations could be based in hardware, firmware, software, or any combination.

Figure 12 is an abstract diagram of a ADWC based on the requirements for an ADWC. Numbered circles in the diagram refer to the corresponding numbered requirement. For example, number circle 1 refers to requirement #1 in section 9.4.6. Solid lines and yellow numbered circles refer to SHALL requirements. Dotted lines and yellow

numbered circles refer to SHALL requirements that have more than one choice for their implementation (e.g., using an internal or an external clock). Dashed lines and gray numbered circles refer to MAY statements. The red rectangle enforces authorization policies. The blue rectangle encloses additional resources which are part of the ADWC. Lines going into or out of the red and blue rectangles are external Signals, events, or Programmatic Interfaces (some of which require authorization).

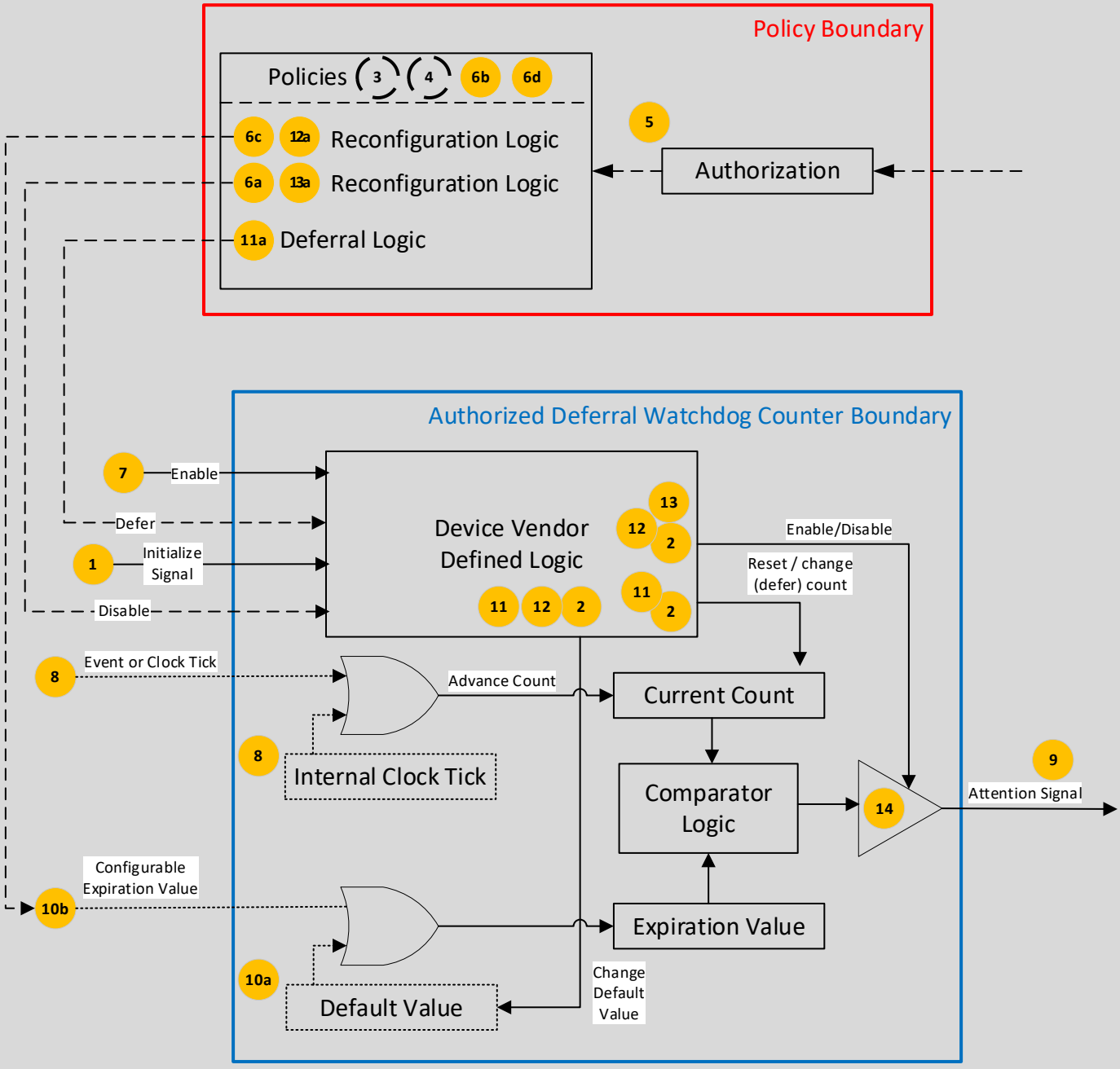

Figure 12: ADWC block diagram

End of informative comment

9.4.8 Wakeup Watchdog Counter

Start of informative comment.

A Wakeup Watchdog Counter (WWC) needs to function independent of any power state a RT may place a Device or CRM into. Note that complete loss of external power is more likely to be an indication of change in the environment, not malware local to the device.

The effect of external power loss on a WWC should be considered carefully by WWC implementers. A WWC is required to function independently of power states the RT controls, but loss of a device's external power supply could put the device into even lower power states. Consider the result if a WWC still functions when the external power source for a device is unavailable, but the RO that normally processes the Attention Signal output from the WWC is offline. Some WWC implementers may consider the loss of external power out of scope for their design. Other WWC implementations may take care to ensure the Attention Signal output from a WWC is received and acted on by a RO after external power loss and before the Initialize Signal is sent to the WWC.

Some implementations of a WWC may depend on an external power source. A WWC implementer should consider whether their design should wake up after an external power source is restored, continue onward in whatever power state the device had prior to the power loss or perform some other scenario appropriate action. An implementer may choose to have time (or events) advance for a WWC for the duration external power was lost or have the WWC ignore the passage of time (or events) while external power is off. Some implementations may provide an indication to the RE that external power was interrupted.

Implementations could rely on the RO generating separate Initialize Signals to indicate complete external power loss, entry or exit to/from low power states, or other power relevant events. Each Initialize Signal could result in a different Device Vendor specific behavior for the WWC. For example, a WWC might divide the remaining time before expiration in half each time the WWC external power is lost. This guarantees that the WWC will eventually expire, irrespective of the duration of time periods between clock ticks or events counted by the WWC.

End of informative comment

9.4.8.1 Latchable Wakeup Watchdog Counter (LWWC)

Start of informative comment.

A Latchable Wakeup Watchdog Counter (LWWC) is a Watchdog Counter that is both an LWC and a WWC.

End of informative comment

Requirements for an LWWC:

1) An LWWC SHALL meet the requirements for an LWC.
2) An LWWC SHALL continue to function independent of Device or CRM power states the RT may control.

9.4.8.2 Authorized Deferral Wakeup Watchdog Counter (ADWWC)

Start of informative comment.

An Authorized Deferral Wakeup Watchdog Counter (ADWWC) is a Watchdog Counter that is both a ADWC and a WWC.

End of informative comment

Requirements for a ADWWC:

1) An ADWWC SHALL meet the requirements for an ADWC.
2) An ADWWC SHALL continue to function independent of Device or CRM power states the RT may control.

9.4.8.3 Wakeup Watchdog Counter (WWC)

Start of informative comment.

A Wakeup Watchdog Counter (WWC) is a Watchdog Counter that is either an LWWC or an ADWWC.

End of informative comment

Requirements for the WWC:

1) A WWC SHALL meet the requirements of one of the following:
   a) an LWWC, or
   b) an ADWWC.

10 Architectural Element Requirements

10.1 Resilience Engine Requirements

Start of informative comment.

By definition the RE needs to be able to service its RTs.

End of informative comment

Requirements for the RE:

1) The RE SHALL be able to service its RT(s).

10.2 Resilience Target Requirements

Start of informative comment.

RT requirements are not within the scope of this specification.

End of informative comment

10.3 Resilience Authority Requirements

Start of informative comment.

RA requirements are not within the scope of this specification.

End of informative comment

10.4 Requirements for the Resilience Orchestrator

Start of informative comment.

This section describes how the input and output Signals of the CRBBs can be used to build devices with enhanced resilient capabilities.

This specification stipulates requirements for CRBBs that can be used to build CRMs. The resilient capabilities provided by the CRBBs described in this specification have the property that when enabled (or activated) they cannot be disabled (or deactivated) nor, in some cases, reconfigured via their Programmatic Interface until they receive an Initialize Signal. (Note: An exception is an ADWC that may be reconfigured or deferred with authorization.) Such resilient capabilities are called resettable resilient capabilities. The Initialize Signal returns a resettable resilient capability into an initial state (or some other Device Vendor defined state), in which it can be (re)configured and activated again. This section describes how CRBBs with resettable resilient capabilities are coordinated by a RO to build a CRM.

The CRBBs with resettable resilient capabilities are:

- Protection Latches (Read, Write and Read-Write)
- Watchdog Counters (LWC, ADWC, WWC, LWWC and ADWWC)

The CRBBs are described as standalone elements with input and output Signals. The requirements of a Protection Latch specify an input Initialize Signal. The requirements of a Watchdog Counter specify both an input Initialize Signal, and an output Attention Signal.

In addition to the Signals, Protection Latches and Watchdog Counters can implement additional Programmatic Interfaces that can be used, for example, to attempt to activate a Protection Latch or to configure or defer a

Watchdog Counter. The Programmatic Interfaces may be accessible from both inside and outside the SEE (e.g., to software executing in the RE or in the RT)

For the CRBBs to deliver useful resilient capabilities, their signals must be protected by the Device Vendor mechanisms. For example, if arbitrary software (e.g., the RT) were able to directly generate Initialize Signals, then malware might be able to deactivate Protection Latches or disable Watchdog Counters. To mitigate such threats, some trusted entity must be in ultimate control of generating the Initialize Signals and responding to the Attention Signals within the scope of a CRM. This specification refers to the entity that manages the signals as the RO.

A building block is internal to a CRM if it is needed by the RE to protect read or write access to persistent storage from the RT, or if it is relied upon by the RE to regain control from the RT. However, the building blocks defined in sections 6 and 9 are useful generally and not all uses will be to protect the RE from the RT or to help the RE regain control from the RT. A device could have many applications for the building blocks unrelated to the relationship between the RE and RT and the RO does not need to control the signals generated or received by building blocks in those applications.

If a Watchdog Counter (that is internal to a CRM), a RT, a RE, or an external entity generates an Attention Signal, the main function of the RO is to generate the Initialize Signals of all the CRBBs that it is managing and perform Module Reset.

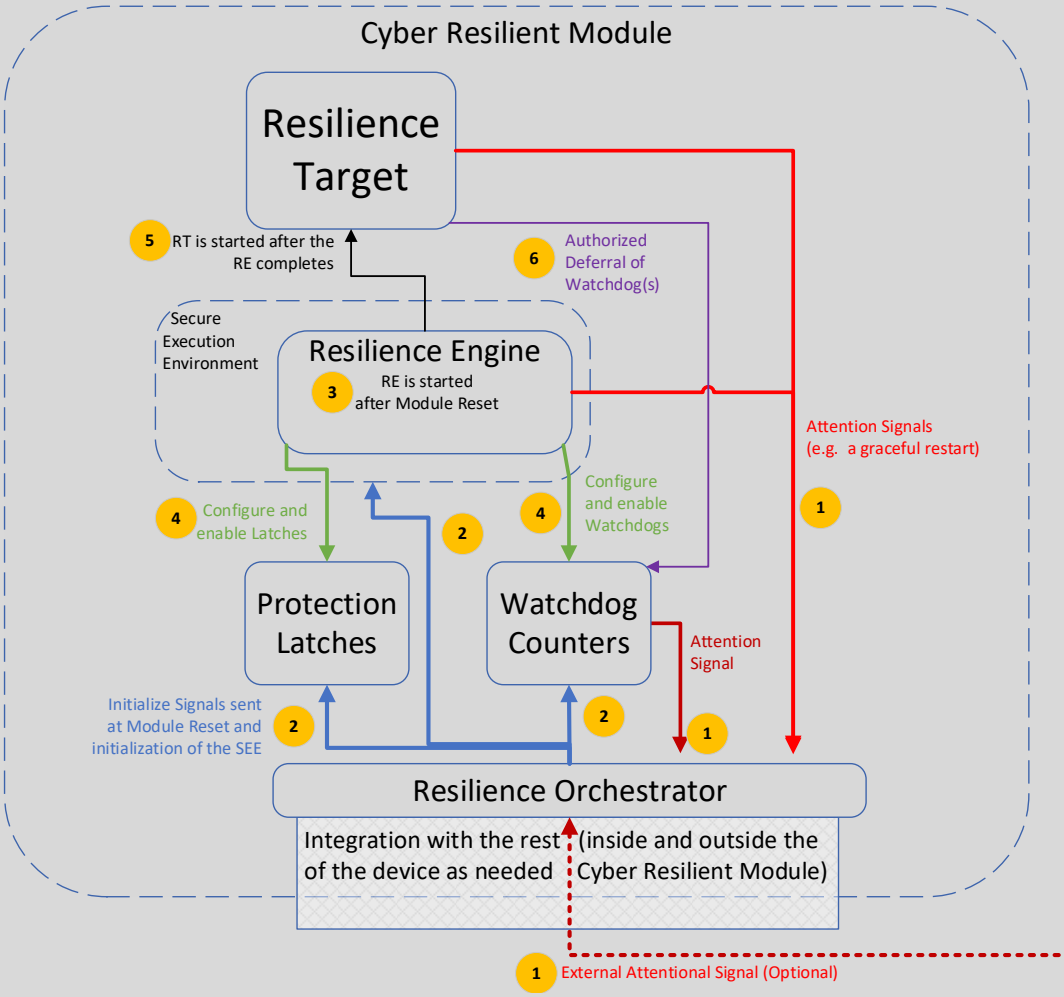

Figure 13: Example RO

Figure 13 is a schematic illustration of the interconnections between a RO and the CRBBs in a CRM.

Figure 13 illustrates that the Initialize Signals to the building blocks internal to the CRM are exclusively controlled by the RO, and the Attention Signals from internal Watchdog Counters, the RT, RE or an external source serve as inputs to the RO. The figure only shows a single CRM, but a device could have any number of CRMs, some of which might be able to undergo Module Reset independently. Note: The figure does not show any Watchdog Counters or Protection Latches that are external to the CRM.

An Attention Signal directed to a CRM may be produced external to the CRM. The RO may receive Attention Signals from device components external to the CRM. The RO may also accept attention requests from components internal to the CRM (e.g., from software in the RE or RT).

Figure 13 also illustrates (see circle labeled 4 in the diagram) that the CRBBs may present Programmatic Interfaces to software running on the CRM. Software running in the SEE (e.g., early boot code when the SEE is established by Module Reset) may use the Programmatic Interfaces presented by the other building blocks.

The following is a description of a typical sequence of activities anticipated in a CRM. The numbered list below corresponds to the numbered items in Figure 13.

1) A Watchdog Counter, RE, RT, or something external to the CRM sends an Attention Signal to the RO. After the RO receives the Attention Signal it will perform a Module Reset, but first the RO may take vendor specific policy actions to prepare for a Module Reset to occur. This could, for example, give software a small amount of time to save volatile data, address safety related concerns (e.g., a traffic light in the United States could transition to flashing red, indicating for drivers to safely stop at the light), notify other components on the same device that the module will be reset, or park rotational media.
2) The RO performs Module Reset and sends an Initialize Signal to all CRBBs internal to the CRM. (Note: There could be many other CRBBs on the same device that are external to this CRM that are not sent an Initialize Signal.) Figure 13 shows the RO causing a Module Reset to establish the SEE, and sending an Initialize Signal to a Protection Latch and a Watchdog Counter because they are all internal to the module for this example.
3) Sometime after the SEE is reset, it starts running the RE. There could be any number of activities that occur prior to the RE starting. The critical condition is that the RE starts execution prior to the RT and the RE controls when the RT execution starts.
4) The RE can perform servicing actions and then it will probably configure and activate a Protection Latch to protect the RE code and configuration information (assuming they are mutable). The RE may also configure and enable a Watchdog Counter to make sure the RE gets control back from the RT within a maximum threshold of time.
5) When its actions are completed, the RE transfers control to the RT.
6) If the Watchdog Counter is an ADWC, the RT may interact with a RA to obtain a Deferral Ticket it passes to the Watchdog Counter. If the RA believes the RE doesn't need to be executed, it can continue to repeatedly issue Deferral Tickets to the RT for forwarding to the ADWC. If the RT cannot obtain a Deferral Ticket or if it decides to restart gracefully the sequence jumps back to step 1.

The RO is an Architectural Element (as described in section 5) that coordinates the reset of the CRM.

This specification does not dictate a power management architecture for CRMs. An implementer may choose to define various module or device power state change events (such as power on, or power loss) as external Attention Signals.

Most devices of non-negligible complexity will include functionality to coordinate reset and power-on of its constituent subcomponents. Depending on its precise behavior, this functionality may meet the requirement for a compliant RO or it may need to be modified to serve as a RO.

A RO for a CRM that uses Module Reset to establish a SEE for Temporal Protection of the RE from the RT coordinates the reset of the CRBBs and any other contained capabilities. If a Watchdog Counter, RE, RT, or an external entity generates an Attention Signal, then the RO will ensure that all CRBBs internal to the CRM and any other implementation specific components (if needed) are reset. Performing Module Reset ensures that control is transferred to the SEE with Temporal Protection of the RE from the RT.

Implementers should ensure that any other functionality that might interfere with the proper operation of an early boot SEE is also reset.

End of informative comment

RO requirements for a CRM comprised of one or more CRBBs, using a Module Reset Established SEE:

1) The RO SHALL receive all Attention Signals from all CRBBs internal to the CRM.
2) The RO SHALL receive all Attention Signals intended for the CRM from all sources external to the CRM.
3) Only the RO SHALL send Initialize Signals to the CRBBs (Storage Protection Latches, Watchdog Counters, SEE) internal to the CRM.
4) Following reception of any Attention Signal the RO SHALL perform a Module Reset (see section 9.1.1)

5) Following reception of any Attention Signal the RO MAY implement a policy that allows it to delay or perform sequencing of CRBB initialization steps as needed to complete a Module Reset.

11 Cyber Resilient Module Requirements

11.1 CRM using a Module Reset to establish a SEE with Temporal Protection

Start of informative comment.

This section includes requirements for a CRM that uses Module Reset to establish a SEE with Temporal Protection of the RE from the RT.

The ability for a RE to gain control and perform servicing actions on behalf of its RA on the RT is what provides many of the resilience benefits contained in this specification. The relationship between the RE and RT forms the basis for the definition of a CRM. This specification focuses on providing requirements for capabilities that support resilient devices without dictating precisely how devices must use the capabilities. The requirements for a CRM provide flexibility for implementers to decide what to implement in hardware versus software. It is anticipated that Device Vendors will use the following steps to implement the cyber resilient capabilities described in this specification:

1) Designate some hardware and software resources as the RE
2) Designate some mutable hardware and software resources as the RT.
3) Provide a way for the RE to service the RT.
4) Prevent the RT from modifying any mutable portions of the RE.
5) Provide a way for the RE to regain control in a SEE after passing control to the RT, even if the RT does not cooperate.

As mentioned in section 1, this specification's scope includes requirements for implementations that use Module Reset to establish a SEE with Temporal Protection of the RE from the RT. Requirements for other ways to establish a SEE for the RE are out of scope for this specification.

A CRM that uses Module Reset needs to have at least one CRBB internal to the module so there is something to be reset. There also needs to be a RO to perform activities such as send the Initialize Signal to the CRBBs in the module. If the CRM relies on a SEE to provide Temporal Protection of the RE, then the SEE needs to meet the requirements in section 9.1.1.

There is no explicit requirement the CRM can recover from low power states. Requirements associated with low power states are covered later in this section.

End of informative comment

Requirements for CRMs that use a Module Reset Established SEE to protect the RE from the RT

1) The CRM SHALL include at least one Protection Latch (as defined in section 9.3.9), LWC, ADWC, LWWC or ADWWC (as defined in section 9.4).
2) The CRM SHALL include a RO satisfying the requirements of section 10.4.
3) The CRM SHALL meet the requirements for Module Reset to establish a SEE with Temporal Protection of the RE from the RT in section 9.1.1.

11.2 Mitigations Against DoS Attacks Through Low Power States

Start of informative comment.

To be resilient, a CRM needs mitigations or protections against the RT being compromised and placing the module into a lower state and potentially being unavailable for an unacceptable duration. See section 9.4.8 for additional background.

This requirement could be implemented at the device level if a single WWC or other design is sufficient to prevent unacceptably long durations of low power states for all CRMs in a device. However, a single WWC may be insufficient for architectures with CRMs that can undergo Module Reset independently.

In some cases, CRM implementations may rely on non-technical means to mitigate the risk that the module may be placed into a prolonged low power state. For example, a handheld device could be designed to have its battery replaced daily and the process of replacing the battery could cause the device to perform a full boot that passes control to a RE.

Some devices may not implement low power states at all.

Mechanisms to force a Module Reset after a CRM has entered a low power state could be manual, internal, or external. A manual example is a device designed to be turned on and off regularly by an interactive user. An external example is a device implemented to wake up on receipt of a network packet, whereby an external entity could cause a Module Reset even if the CRM is in a low power state.

The following subsections describe two kinds of CRMs and their corresponding requirements. The first is a Self-Recoverable CRM that can initiate recovery without relying on external Attention Signals or manual actions. This is a good choice for devices that need to be recovered without physical interaction or may be deployed in physically inaccessible locations or may lack a user interface to initiate recovery. The second is a Symbiotic CRM that may have dependences on external Attention Signals or manual interaction to initiate recovery. To be recoverable, a CRM must have some mechanism to force recovery to occur.

End of informative comment

11.2.1 Self-Recoverable Cyber Resilient Module

Start of informative comment.

A Self-Recoverable CRM does not need assistance to initiate recovery actions.

End of informative comment

Requirements to prevent DoS attacks through low power states for a Self-Recoverable CRM:

1) If a Self-Recoverable CRM implementation supports low power states, it SHALL include an internal WWC or provide some other internal mechanism to force a Module Reset based on time or events that cannot be cancelled without authorization while enabled.

11.2.2 Symbiotic Cyber Resilient Module

Start of informative comment.

A Symbiotic CRM that needs assistance to initiate recovery actions.

End of informative comment

Requirements to prevent DoS attacks through low power states for a Symbiotic CRM:

1) If a Symbiotic CRM implementation supports low power states, it SHALL support at least one mechanism (internal, external, or manual) to force a Module Reset based on time or events that cannot be cancelled without authorization while enabled.

11.3 Roots of Trust

11.3.1 Attestation

Start of informative comment.

Attestation is the act of cryptographically authenticating a device and the software that it booted or is running. A variety of technologies that provide attestation capabilities are available. Current TCG technologies include the Trusted Platform Module (TPM) [3] and the Device Identifier Composition Engine (DICE) [4].

Attestation is included in TCG requirements for a CRM to provide a mechanism to verify a RE can be trusted to perform its functions and to verify a RT has been serviced. Attestation is recommended for the RO, however, some

implementations may need to implicitly trust the RO, especially if some or all the RO is loaded before or as part of the core root of trust for measurement.

Attestation is important for secure and reliable remote management of some classes of device. For example, a RA that initiates firmware update for a family of devices may then demand that the devices provide attestation evidence that they are booting up-to-date firmware. If devices attest out-of-date configurations or do not provide attestation evidence, then the RA may refuse to issue ADWC Deferral Tickets to non-compliant devices to force them to remediate.

End of informative comment

Requirements related to Roots of Trust and attestation:

1) One or more Roots of Trust SHALL be implemented to support attestation of the RE and RT.
2) One or more Roots of Trust SHOULD be implemented to support attestation of the RO.

Start of informative comment.

While a Root of Trust supporting attestation is a requirement of this specification, no restrictions are placed on how a Root of Trust is implemented (e.g., TPM, DICE, or other).

Note that a storage RWPL may be used to help implement a root of trust that provides attestation. One possible implementation that mirrors DICE technologies is described in appendix 13.1.1 of this specification.

This specification does not favor a dedicated TPM or DICE over a storage RWPL to implement support for attestation. Device Vendors should implement technologies that are appropriate for their customer use cases and threat models.

Additional notes on Roots of Trust and CRM attestation:

- The Roots of Trust supporting attestation may reside within a CRM or elsewhere on the device.
- For those CRMs in which the RE is implemented in ROM, attestation of the RE may be implicit.
- This specification assumes the RA role includes verification of attestation statements.
- Either the RE or the RT may be the actor communicating with a verifier.
- The chain of trust for measurement can stop at an engine that is trusted to provide sufficient security policy enforcement for a verifier to infer trust in later components.

The Trusted Computing Group provides further specifications (e.g., TPM [3] and DICE [4]) containing guidance on Roots of Trust and attestation.

End of informative comment

11.3.2 Secure Boot

Start of informative comment.

Secure Boot is a family of technologies for ensuring that a device can run only authorized code, for example, by checking that code has a particular hash, or checking that code is signed using an authorized key. In the context of this specification the term Secure Boot is not to be confused with the Secure Boot feature of the Unified Extensible Firmware Interface (UEFI) specification.

Many devices implement Secure Boot in stages: a first stage bootloader loads and authenticates a second-stage bootloader before passing control to the second stage. The second stage bootloader then loads and authenticates the OS (or possibly the third-stage bootloader), and so on.

Secure Boot implementations generally require hardware-support to ensure that the first stage boot loader is itself authorized. One possible implementation is CPU internal microcode that hashes the first stage bootloader and compares the hash to a value stored in e-fuses. If the first stage boot loader is authorized (i.e., the hashes match),

then control is passed to the first-stage boot loader. If the hashes do not match, then some recovery action is performed.

It is recommended that Cyber Resilient Devices implement Secure Boot.

One possible implementation of Secure Boot using WPLs is described in an appendix to this document in section 13.1.2.

End of informative comment

12 Cyber Resilient Module Profiles

Start of informative comment.

This section contains recommended descriptors which implementers (such as Device or Module manufacturers) are encouraged to use to denote the CRBBs and other relevant trusted computing capabilities a Cyber Resilient Module or Device contains.

A CRM may be implemented inside a component or device that has additional security capabilities or trusted computing components. This section also contains informative language for conveying what capabilities and Trusted Computing components are implemented in a component or a device to help support the security of the CRM.

Table 4 is an inventory of building blocks defined in sections 9.3 and 9.4. The entries in the left column list the acronyms defined earlier in this specification. The normative language (defined below) is used to list the types of building blocks the CRM contains. The notation does not communicate how many of each type of building block are in the CRM. The notation is written out as “CRM” followed by each item listed in parenthesis.

Building Block Implemented	Description
WPL	One or more Write-Protection Latches (e.g., that can be used to protect the RE and its configuration data)
RPL	One or more Read-Protection Latches (e.g., that can be used to protect a device secret)
RWPL	One or more Read-Write-Protection Latches (e.g., that can be used to protect a device secret and/or a RE along with its configuration data)
BWC	One or more Basic Watchdog Counters
LWC	One or more Latchable Watchdog Counters
ADWC	One or more Authorized Deferral Watchdog Counters
LWWC	One or more Latchable Wakeup Watchdog Counters
ADWWC	One or more Authorized Deferral Wakeup Watchdog Counters

Table 4: CRBB Nomenclature

Some examples are:

CRM (WPL)
CRM (RPL)
CRM (WPL, RWPL)
CRM (RWPL, BWC)
CRM (WPL, LWC)
CRM (WPL, ADWWC)
CRM (WPL, ADWC, ADWWC)
CRM (RWPL, BWC, LWWC)
CRM (RWPL, BWC, ADWWC)

This nomenclature is encouraged to indicate the types of building blocks from sections 9.3 and 9.4 a CRM contains:

CRM (<comma separated list of acronyms of the implemented building blocks>)

There is potential for ambiguity when a CRM contains a wakeup form of watchdog counters, such as an LWWC or ADWWC. When the term “LWWC,” is included, it can be assumed that the LWWC is also a type of LWC, so it is recommended the implementer lists only “LWWC,” as “LWC” is implied.

The same pattern applies for ADWWC, which is a type of AWDC. When a CRM contains an ADWWC, it is recommended "ADWWC" is listed alone, as "ADWC" is implied.

If an implementation contains independent wakeup and non-wakeup versions of a counter, the implementer can decide to list each to denote that the implementation contains both a wakeup and non-wakeup counter.

Table 5 contains a list of security capabilities and trusted computing components that may contribute to the security of a CRM.

Capability or Trusted Computing Component Implemented	**Description**
TPM	A Trusted Platform Module
DICE	Device Identifier Composition Engine
SB	Secure Boot

Table 5: Capability or Trusted Computing Component

When a CRM is implemented inside a component or device that has additional security capabilities or trusted computing components, it is recommended the following nomenclature is used to indicate which security capabilities and/or trusted computing components are available to support the security of a CRM:

> CRM (<comma separated list of implemented CRBBs>) <plus sign separated list of implemented security capabilities and/or trusted computing components>
>
> Examples:
>
> > CRM (WPL, ADWC) + TPM
> >
> > CRM (WPL, LWC) + SB + DICE

Appendix 13.2 describes some of the specific threats that can be mitigated using the CRBBs defined in this specification using the nomenclature in this section.

End of informative comment

13 Appendices (Informative)

13.1 Examples of Implementing Attestation and Secure Boot using Protection Latches

Start of informative comment.

RWPLs can be used to support attestation and Secure Boot of Cyber Resilient Devices. This section provides a brief description of each.

End of informative comment

13.1.1 Attestation

Start of informative comment.

DICE-compatible devices implement an engine that provides a secret value called the Compound Device Identity (CDI) to early boot code. The CDI is derived from a device unique secret value called the Unique Device Secret (UDS) and the hash of early boot code (layer 0) using a construction similar to the following:

$$d = hash\ of\ layer\ 0$$

$$CDI = KDF(UDS, d)$$

A Cyber Resilient Device can implement the engine specified in [4] in its earliest boot code by using RWPLs. The UDS is read from a storage location and then its storage location is immediately protected using a RWPL. This prevents later boot code from accessing the stored UDS. In addition to preventing read access to the UDS storage location, the engine must also use the Protection Latch to protect the storage region that contains the engine's image. This prevents later code from modifying the DICE actions in the earliest boot code.

End of informative comment

13.1.2 Secure Boot

Start of informative comment.

Storage WPLs can be used as a foundation for Secure Boot implementations.

For example, a Device Vendor could prepare a first-stage boot loader that checks for the presence and validity of a certificate that authorizes a second-stage boot loader. Alternatively, the first stage bootloader could include a "compiled in" key or certificate that authorizes a second stage bootloader. On their own, neither of these examples are sufficient to implement Secure Boot because the first stage bootloader is not protected and can be replaced by malware. However, if the first stage bootloader write protects its persistent stored image using a Protection Latch before passing control to the next stage, then (later) malware cannot modify the first stage boot loader. To the second stage loader, the first stage loaded is essentially ROM. The Device Vendor may have a mechanism whereby the first stage loader can update itself after verifying an update package. However, there would not be Secure Boot verification for the first stage boot loader.

End of informative comment

13.2 A Device Resilience Scenario

Start of informative comment.

This specification describes the following CRBBs:

1) A Protection Latch blocking Write-Access to storage (WPL)
2) A Protection Latch blocking Read- and Write-Access to storage (RWPL)
3) Several classes of Watchdog Counter (BWC, LWC, ADWC, WWC, LWWC, ADWWC)

The scenario in this section assumes the CRBBs are implemented internal to a CRM and satisfy the requirements for a CRM (e.g., an RO, a Module Reset established SEE, an RE that is executed in the SEE, etc.). This section does not go into detail about how all parts of the CRM are implemented. This section focuses on how the building blocks are used.

This informative appendix describes a Cyber Resilient Device scenario and explains how the CRBBs defined in this specification can be used to support the scenario. This section also sketches how other TCG technologies (TPM, DICE, etc.) can be used to mitigate additional threats to the resilience and security of a design.

This section is not exhaustive: the CRBBs themselves support a greater range of scenarios than are described here, and the resilience scenarios described can be implemented using technologies other than those defined in this specification.

End of informative comment

13.2.1 Scenario-Based Device Security and Management Requirements

Start of informative comment.

In this scenario, a device requires secure and reliable remote management, including the situation where the main device firmware (e.g., the OS) has been compromised and/or is non-cooperative.

Required management actions include:

- firmware updates and
- changes to device configuration data held in non-volatile storage.

The management server requires the device to have the following capabilities in order to enable secure and reliable management:

- cryptographic device identity, to guard against device impersonation,
- attestation, to ensure that the device is up-to-date and properly configured,
- a means to force the device to perform a management action (e.g., an Attention Signal to start the RE)
- storage for device settings, including the Secure Boot policy that identifies authorized firmware, and
- a network connection to the management server. Note that this need not be a direct IP/Internet connection

The device also requires one or more symmetric keys or sealed storage capabilities to enable data at rest protection/encryption.

This scenario assumes that device hardware includes power management capabilities that allow the RT to switch the device off or put the device into a non-operating low power state.

The resilience scenario is not scenario specific: the techniques are largely applicable to IoT devices, subcomponents that are part of a larger device, or any other field-updatable standalone equipment.

End of informative comment

13.2.2 Software/Firmware Architecture to Support the Scenario

Start of informative comment.

This section describes an exemplary software/firmware architecture to support the scenario. No special security or resiliency features are assumed in this section (CRBB, or other). In the sections 13.2.3 and 13.2.4 that follow, security threats inherent in software-only implementations are described, as well as how the threats can be mitigated using the CRBBs defined in this specification.

The device software and hardware are structured as:

1) A RE integrated into early boot firmware (e.g., the device bootloader)
2) A RT executing an OS/application package that is started by the RE

The device is managed by a RA. In the IoT case, the RA is likely to be a cloud or an on-premises device management service. Component devices may be managed by a RA that is distributed across a network service and a more powerful local device such as a host CPU.

The RE has access to enough communication or networking functionality to communicate with the RA. Networking functions may be integrated into the RE execution environment (e.g., u-boot, UEFI), or could be provided by an external network component such as a Wi-Fi module or a separate host processor. Networking may also be implemented in a dedicated stripped-down OS environment. In this case, the stripped-down OS becomes part of the RE.

The device coordinates with the RA to perform management and servicing actions. For example, the RT periodically contacts the RA to determine whether management actions need to be performed. When needed, the RA can instruct the device to download firmware updates (patches) and other configuration changes. When the RT is operating correctly, the RT will perform the requested function. If the RT is un-responsive or uncooperative, the RE can perform these functions following a device reset.

Patches and changes to persistent configuration data are always signed by the RA. The signatures are always checked by the RE (if the RE is being updated, or if the RT is being updated by the RE) or the RT (if the RT is updating itself) before any changes are made.

During normal operation, the RT will normally be responsible for servicing itself without the involvement of the RE. However, the RE can also service the RT if the RA demands it (e.g., if an attestation claim is out-of-policy), or if the RT is non-functional. However, the RT is *not* allowed to service the RE directly: changes to the RE are only ever applied by the RE itself.

The architecture supports both A/B updates [5] (for both the RE and/or the RT) or in-place updates. This is not discussed further here.

The RE enables attestation and device identity by the RE measuring some or all the RT at boot time. How this is performed varies depending on the security technology utilized (e.g., TPM, DICE) and is not discussed further here. The RE also provides one or more symmetric keys to the RT that the RT uses for encryption and decryption to protect data at rest.

The RE will normally require both code and configuration settings in order to perform its function. For the purposes of this example, changes to persistent configuration settings are handled in the same way as patches: specifically, requested changes are signed by the RA and the signatures are verified by the RE before the changes are made.

The RE validates that all or part of the RT is authorized before starting the RT - i.e., the RE Secure Boots the RT. If the RT is out of policy, the RE will attempt remediation: perhaps using a local golden copy of the RT, or perhaps by contacting the RA to obtain a patch or new RT package.

End of informative comment

13.2.3 Risks Associated with a Software-Only Implementation of the Architecture

Start of informative comment.

If the module provides no resilience functions, then protection for all functions must be implemented in software alone. It is beyond the scope of this section to describe best practice architectures and software engineering, but it is likely that the best-effort Trusted Computing Base (TCB) for device servicing and recovery is much larger than a TCB that utilizes CRBBs.

In the discussion that follows, this scenario assumes that no dedicated security hardware exists. This is likely the situation for a low-end microcontroller. More powerful microprocessors will probably (at least) provide privilege

levels, which can provide protection for resilience functions. This reduces the risk of RE or RT compromise, but does not alter the fact that compromise may still occur.

If the RT or RE is compromised, the following may occur:

RT Runtime Compromise:

The device may refuse RA management instructions, and instead of performing its intended function, the device may perform unwanted actions under the control of an adversary.

RT compromise may lead to the stored image or critical settings of the RE or RT being compromised.

RE or RT Stored Image Compromise

The device remains compromised even after a reset or power cycle. Attestation claims are no longer trustworthy.

RE Configuration Settings Compromise

Deletion or modification of security critical settings (e.g., the public key or certificate that the RE uses to verify patches came from the RA) can lead to an adversary taking over the device.

Device Identity Compromise

Since there is no hardware protection for device identity keys, compromise of the RT can easily result in device identity key compromise. If the device identity key is compromised, the device can be freely impersonated, and attestations are unreliable.

Encryption Key Compromise

Devices may use an encryption key to encrypt data at rest for confidentiality of data. Since there is no hardware protection for device encryption keys, compromise of the RT can result in compromise of encryption keys.

Unresponsive or Uncooperative RT:

If the RT is compromised, it may refuse to perform management and servicing functions. This need not be a compromise of the stored image of the RT: runtime compromise can still lead to a device that can no longer be controlled by the RA.

Programmatic Power Down of the Device

Some devices can be programmatically switched off or put into deep sleep states with no means to restore functionality without external interaction. Malware in the RT may be able to deny service by switching off the device using these facilities.

Some of these eventualities can result in significant harm to the device. For example, if the stored image of the RT is compromised but the RE remains intact, then the device can be recovered using the RE. However, If the stored image of the RE itself is compromised, it is unlikely that a device can be remediated without manual (or possibly factory) repair. Similarly, if the device identity key is compromised, then it is unlikely that new keys can be securely provisioned without manual or factory repair to securely re-key the device.

The CRBB profiles described in the next section provide capabilities to reduce the likelihood of these unrecoverable compromises and allow for automated device recovery in most circumstances.

End of informative comment

13.2.4 Selecting CRBBs to Mitigate Threats

Start of informative comment.

This section describes how the CRBBs can be utilized to mitigate the security threats described in the previous section. The mapping of CRBBs (and other TCG security technologies) to threats is summarized in tabular form, and then described in more detail in the subsections that follow.

Table 6 illustrates how the security threats identified in the software-only implementation are mitigated by the CRBBs defined in this specification. A green-shaded cell means that the threat is mitigated by the CRBB. A yellow shaded cell means that the threat is mitigated if the device requirements in the note are satisfied.

Note that TPM 2.0 Revision 1.59 and later includes a richly configurable Authenticated Countdown Timer. DICE is included because it is another TCG technology that enables secure and reliable device identity and attestation.

Security or Resilience Threat	**Mitigation**						
	WPL	**RWPL** Note 7	**DICE**	**TPM**	**LWC**	**ADWC**	**WWC**
RE or RT Stored Image Compromise		Note 6					
RE Configuration Settings Compromise		Note 6		Note 9			
Device Identity Compromise		Note 1					
Device Identity Compromise (Attestation)		Note 1					
Encryption Key Compromise		Note 1 Note 2	Note 2				
Unresponsive or Uncooperative RT				Note 3	Note 4		
Programmatic Power Down of the Device				Note 8	Note 5	Note 5	

Table 6: Mapping of CRBB (and other security technologies) to resilience scenarios

Notes:

1) Assuming DICE is implemented using a RWPL, for example, as described in section 13.1.2
2) The DICE specification [4] describes how device identity and attestation keys can be derived. Similar techniques can be used to derive keys for data encryption or sealing.
3) If the TPM implements the Authenticated Countdown Timer feature and the device uses it as an ADWC or an LWC.
4) If an LWC is used, then control-transfer to the RE is unconditional (i.e., the device must occasionally reboot).
5) This specification does not place requirements on whether the RT can power-manage/switch off the ADWC or LWC (or equivalently, switch off the module in such a way that the BWC, LWC or ADWC cannot restart the device). If the Watchdog Counters are integrated into the device in such a way that they can reliably function (and wake up the device) regardless of power management actions performed by the RT, then the ADWC and LWC can also serve as a WWC in the event that the device is placed into a low power state.
6) A RWPL can be used to protect the RE and configuration data as an alternative to a WPL. Note, however, that using a RWPL in place of a WPL will prevent the RE from executing in place while the Protection Latch is activated, and it will also prevent code from retrieving configuration data

stored in a location protected against reads. Both drawbacks can be overcome by copying data from persistent storage to RAM before activating the RPL.

7) This column is relevant to a dedicated RWPL or a separate RPL and WPL if the RPL and WPL are used together to protect data.
8) If the TPM implements the Authenticated Countdown Timer feature and the device uses it as a WWC
9) An example of how a TPM could prevent the compromise of RE configuration settings is by storing them in TPM NV storage.

Hardware-based attacks, or attacks that require physical presence, are not considered.

End of informative comment

13.2.4.1 Profile CRM (WPL)

13.2.4.1.1 CRBBs Assumed

Start of informative comment.

This profile assumes the following CRBB is available for its use:

1) One WPL that can be used to protect the RE

End of informative comment

13.2.4.1.1.1 CRBB Support for the Scenario

Start of informative comment.

The RE can only safely and reliably perform its function if the stored image of the RE cannot be changed by an adversary.

The risk of stored image compromise is greatly reduced if the RE uses a WPL to protect its code and critical state before passing control to the RT. In this case, RT compromise may lead to a malfunctioning device, but power-cycling or resetting the device will return the device to an unmodified RE, which can assess the state of the device, and if necessary, perform remediation.

WPLs can also be used to protect critical device state. Examples of state that may be protected include the Secure Boot policy for the device (e.g., the hash of the RT) and the URL of a cloud management controller (RA) together with its associated certificate.

If the RE maintains a recent RT “golden image,” then the storage containing the golden image can also be write-protected. In the event of compromise of the stored image of the active RT, then the stored golden copy can be used to repair the active copy.

Note that RE malfunction and compromise prior to activating the WPL and protecting storage can still lead to an unrecoverable compromise. The risk of RE compromise will usually be far less than the risk of RT compromise because the RE performs fewer functions and typically presents a far less complex interface through which it can be attacked. Designs can structure the RE to further mitigate risks: for example, RE network access opens a large potential attack surface, so write-locking storage before the RE enables networking can lessen the risk that a network compromise can lead to persistent image compromise. Of course, if this strategy is employed, then downloaded patches must be staged to non-write-protected storage and patches can only be applied after a device reset.

Note also that if the device provides only one WPL, then all data that needs to be write-protected (e.g., the RE, RE-settings, and the golden image) must be laid out in storage in such a way that it can all be latched together by the RE.

End of informative comment

13.2.4.1.2 Limitations

Start of informative comment.

This profile has the following limitations:

1) This profile provides no hardware support for key protection for device identity, attestation, and data encryption keys. If these functions are required, then they can be implemented in software (resulting in low assurance protection for keys), or an additional external security device can be used (e.g., a TPM).
2) A management controller (RA) cannot force updates or reboots if the RT is unresponsive.[1] If this functionality is required, then additional support logic must be included (e.g., a management Service Processor or BMC). Alternatively, a local user may reset or power-cycle the device.
3) A misbehaving RT can switch the device off, requiring external intervention to restore operation.

End of informative comment

13.2.4.2 Profile CRM (WPL, RWPL)

13.2.4.2.1 CRBBs Assumed

Start of informative comment.

This profile assumes the following CRBBs are available for its use:

1) One WPL that can protect the RE stored image, and
2) One RWPL to protect a device secret.

End of informative comment

CRBB Support for the Scenario

Start of informative comment.

In addition to the resilience benefits described for the CRM (WPL) profile, devices that satisfy this profile also enable enhanced protection for device identity keys, encryption keys, and attestation using DICE technologies.

In a nutshell, the RE can read one or more device-specific secrets from storage early in boot, and then enable a RWPL to protect the storage region containing the secrets so that later code, for example, the RT, cannot read or modify the device-specific secrets. Note that an adversary overwriting device-specific secrets could be damaging, so write protection is also needed for the device-specific secrets. Of course, if the key is still held in RAM or registers, then the secrets are still vulnerable. DICE specifications [4] describe the use of one-way functions and key-derivation functions to create derived identity keys. Similar techniques can be used to derive keys for bulk data encryption. Disclosure of a DICE-derived key does not lead to compromise of the underlying hardware-protected key. For more details, see the DICE Layering Architecture specification [4], and the discussion of implementing Roots of Trust using Protection Latches in Appendix 13.1.

This scenario assumes both a WPL (to protect boot code) *and* a RWPL (to protect a device secret) because adversarial modification of the early boot code can lead to disclosure of the device specific secret. If early boot code is in ROM, or if the device has other mechanisms to prevent unauthorized modification of early boot code, then the profile CRM (RWPL) to protect a device secret is useful, without the need for the WPL.

End of informative comment

13.2.4.2.2 Limitations

Start of informative comment.

[1] Note that a boot-time-isolated RE cannot reliably perform functions once the RT has been started because the RT has control of execution.

This profile has the following limitations:

1) A management controller (RA) cannot force updates or reboots if the RT is unresponsive. If this functionality is required, then additional support logic must be included (e.g., a management Service Processor or BMC). Alternatively, a local user may reset or power-cycle the device.
2) A misbehaving RT can switch the device off, requiring external intervention to restore operation.

End of informative comment

13.2.4.3 Profiles CRM (WPL, RWPL, ADWC) or CRM (WPL, RWPL, LWC)

13.2.4.3.1 CRBBs Assumed

Start of informative comment.

This profile assumes the following CRBBs are available for its use:

1) One WPL
2) One RWPL
3) An ADWC or LWC

End of informative comment

13.2.4.3.2 CRBB Support for the Scenario

Start of informative comment.

In addition to the resilience support described in the profile CRM (WPL, RWPL), devices that satisfy these profiles allow the RA to also force the device to reset itself, resulting in control being transferred to the RE for device health assessment and remediation.

The means by which the RA forces the device to reset itself varies depending on the Watchdog Counter utilized.

In the case of an ADWC, the RA forces reset by withholding Deferral Tickets. This means that a device that uses an ADWC may reset itself if network connectivity is lost. Device Vendors should trade off the maximum time a device should run before forcibly returning control the RE for assessment and recovery, with the risk of unwanted ADWC-initiated service interruptions. For “IoT-style” devices that mostly perform functions when online, an expiration value of hours to days may be suitable.

If the device employs an LWC, then control is unconditionally transferred to the RE at a time chosen by the RE: perhaps in the middle of the night. Note that if the RT is operating correctly, and the device allows it, the RT may voluntarily suspend execution, reset itself, and the RE can resume operation with less downtime than a full reboot.

If unwanted resets - either caused by service or network downtime (ADWC) or because of LWC Attention Signals - are not acceptable, then alternative designs should be used. For example, the device might provide a physical button that allows a physically present operator to manually defer the ADWC.

A WWC should be employed if the device supports low power states, and the RT can prevent the other counters from working by altering the device power state.

End of informative comment

13.2.4.3.3 Limitations

Start of informative comment.

This profile has the following limitation:

1) If the Watchdog Counter cannot reliably function in the face of adversarial power management by the RT, then a misbehaving RT can switch the device off, requiring external intervention to restore operation.

End of informative comment

13.2.4.4 Profiles CRM (WPL, RWPL, LWWC) or CRM (WPL, RWPL, ADWWC)

13.2.4.4.1 CRBBs Assumed

Start of informative comment.

This profile assumes the following CRBBs are available for its use:

1) One WPL
2) One RWPL
3) A WWC

End of informative comment

13.2.4.4.2 CRBB Support for the Scenario

Start of informative comment.

In addition to the resilience support described in the profiles CRM (WPL, RWPL, ADWC) or CRM (WPL, RWPL, LWC), devices that satisfy this profile provide the RE with a WWC that can be used to wake up the device and transfer control to the RE, even if an adversarial RT has put the device into a low power/non-operative state.

End of informative comment

13.3 A Subcomponent Resilience Scenario

13.3.1 Subcomponent Resilience Scenario Background

Start of informative comment.

In this profile a subcomponent of a larger platform is used to describe how CRBB's could be used to provide resilience of the subcomponent. The description here is one of many possible different implementations which could vary by vendor, architecture, goals, and subcomponent type.

In many larger platforms (e.g., PC, server) a mass storage device (e.g., hard disk drive (HDD), solid state drive (SDD)) is used to storage large quantities of non-volatile code and data. Traditionally such storage devices have been removable/replaceable, but in many newer platforms, especially smaller platforms like notebooks, these devices may be soldered to the main printed circuit board (PCB). These storage devices perform an important function by storing the main OS, applications, and user data. As such, the ongoing functioning and reliability of these devices is critical in the platform's overall resilience goals.

These types of storage devices are often complex standalone "embedded devices" which contain one or more microcontrollers, possibly some immutable non-volatile memory (e.g., ROM), mutable non-volatile memory (e.g., flash ROM), volatile memory (e.g., DRAM), and an interface to communicate with the host platform. The hardware and firmware contained in the microcontroller and various ROM/flash components are what control the proper functioning of the storage device and its ability to communicate with the host platform. These storage devices also contain the mass storage media (e.g., magnetic spinning disk, or non-volatile mutable flash), and any associated servo controllers and read/write heads, in the case of a magnetic spinning disk-based device.

End of informative comment

13.3.2 Subcomponent Resilience Scenario Requirements

Start of informative comment.

In this scenario the storage device requires the ability to maintain the integrity of its firmware resources and critical data stored in mutable non-volatile memory (flash ROM) to ensure the proper functioning of the storage device, the ability to securely update firmware, and optionally the ability for the device to attest to its state.

One of the core functions of a mass storage device is to provide reliable access to user data stored on the device. However, this scenario is concerned only with the resilience of the firmware and its associated critical data under the assumption that if this is maintained then the device hardware and firmware will provide the necessary functions for reliable access to user data. This means that this scenario is not concerned with how the user data is secured and accessed – this is left to the storage Device Vendors to implement as part of their hardware and firmware.

The device firmware is structured as:

1) A RE integrated into early boot firmware executing on hardware, for example, the device bootloader (which could be contained in immutable ROM).

2) The RT firmware executing on hardware that is started by the RE/bootloader. This RT is the firmware that controls the functioning of the storage device, including access to the mass storage media and interface to the host platform.

The storage device communicates with the host platform over its interface (e.g., Serial ATA, NVMe, SCSI, etc.) to transfer user data, to perform firmware updates, to communicate device health information, to control power modes, etc. In this scenario the host platform can either serve as the RA itself or serve as a conduit (communication path) for another "remote" RA (e.g., the storage vendor, the platform vendor) to/from the storage device, as would be the case when performing a firmware update. Optionally, if supported by the storage device, either the host platform or a remote entity RA could request attestation information from the storage device.

End of informative comment

13.3.3 Selecting CRBBs to Mitigate Threats

13.3.3.1 Profile CRM (WPL)

Start of informative comment.
No identified changes from section 13.2.4.1.

End of informative comment

13.3.3.2 Profile CRM (WPL, RWPL)

Start of informative comment.
No identified changes from section 13.2.4.2.
End of informative comment

13.3.3.3 Profiles CRM (WPL, RWPL, ADWC) or CRM (WPL, RWPL, LWC)

Start of informative comment.
In some scenarios it may be applicable/useful to consider using an ADWC to prevent malware from creating a DoS situation by modification of the RT, disabling the storage device interface, etc. In such a situation, use of an ADWC through lack of authorized Deferral Tickets reaching the ADWC would trigger a reset of the storage device to return it back to the storage device RE. Such abrupt resets of storage devices can have very disruptive effects on platforms, so care must be taken in considering whether use of a Watchdog Counter (of any type) on storage devices is merited. However, it may be appropriate, especially in cases where immediate or timely physical access to the platform is impractical in order to recover access to the storage device, or where a full reset/reboot of the platform is undesirable.

End of informative comment

13.3.3.4 Profile CRM (WPL, RWPL, LWWC) or CRM (WPL, RWPL, ADWWC)

Start of informative comment.
A WWC could be useful in the case where malware puts a storage device into a low power state and won't allow it wake back up, effectively performing a DoS attack. However, similar to section 13.3.3.3, the use of a WWC (or any other Watchdog Counter) should be very carefully considered for subcomponents, especially one such as a storage device which plays a critical role in the functioning of the larger platform.

End of informative comment

13.3.3.5 Host-Symbiont Relationship

Start of informative comment.
In the case of a subcomponent, it may be the case that there are insufficient resources to fully implement some of the CRBB's or other resilient capabilities on the subcomponent itself. In such cases, the subcomponent may unavoidably need to rely on some other, more capable, element in the platform to "assist" the subcomponent in satisfying the resilience goals of the platform. NIST SP 800-193 refers to this situation as a "host-symbiont" relationship, whereby the subcomponent is the "symbiont" (it relies on something else) and the helper element is the "host".

As an example, in this storage device scenario, it could be that the subcomponent does not have an ADWC so the host could implement a function which periodically "pings" the storage device to make sure it is still responsive. If the storage device does not reply in a specified time period or the host decides the storage device is corrupted, then the host could generate an Attention Signal which would cause the storage controller to reset the storage device. If the storage device does reply in a specified time period and isn't suspected of being corrupted, then the periodic "pinging" function would continue.

In this case, the RO for the storage controller can receive an Attention Signal when it is generated by the host. The RO performs a Module Reset, eventually resulting in starting the RE so the RE can perform servicing actions (if needed).

End of informative comment

13.4 IoT Scenarios

Start of informative comment.

This informative appendix section describes several IoT device scenarios and explains how the CRBBs defined in this specification can be used to support these scenarios. A summary chart is included at the end of this section.

This section is intended to guide IoT Device Vendors by providing hints about the Protection Latches and Watchdog Counters that may be suitable for a spectrum of IoT examples.

Latches are useful to protect confidentiality and integrity of data on the device. IoT devices require well protected identity secrets that can be protected using RWPL if DICE or TPM are unavailable. In addition, WPLs can help IoT devices protect their firmware from malware.

End of informative comment

13.4.1 Smart Home

Start of informative comment.

Various smart home devices have different security requirements and therefore different needs for CRBBs.

End of informative comment

13.4.1.1 Exterior Door Lock in Smart Home

Start of informative comment.

The lock on an exterior door in a smart home is critical to the safety of the homeowner because it keeps out intruders. Therefore, strong security is needed against most kinds of compromise. However, a remote monitoring service (e.g., a RA) may not be available so CRBBs that require remote monitoring are impractical (e.g., an ADWC). Without remote monitoring, an LWC is a simple measure that can enable recovery from run-time compromise. A WWC provides some incremental value by preventing malware from sleeping the device, but additional cost is incurred. An LWC can be used by the RE to remove malware periodically, albeit there will remain risks associated with low power states. For example, malware might find a way to perform a DoS attack by putting the device to

sleep after a successful compromise of the RT. Alternatively, the design could provide a way for a user to manually wake up the device and initiate recovery.

End of informative comment

13.4.1.2 Interior Door Lock in Smart Home

Start of informative comment.

The lock on an interior door typically does not need the same level of security as an exterior door lock because it mainly protects the privacy of one resident against intrusions by another resident. Therefore, the security of this lock can be weaker but basic functionality should be preserved. Again, a remote monitoring service may not be available so an ADWC, which requires remote monitoring, is impractical. In the absence of remote monitoring, an LWC is a simple measure that can enable recovery from run-time compromise. A WWC involves more complexity and cost so it might not be practical in an interior door lock.

End of informative comment

13.4.1.3 Smart Smoke Detector in Smart Home

Start of informative comment.

A smoke detector in a smart home is expected to be both resilient to a possible malicious attacker as well as being functionally secure. Once provisioned during initial setup it must never be interrupted in its continuous detection operation. This would need a WWC to monitor any power loss, a way to identify type of power loss, plus a way to differentiate between a dead battery, planned battery replacement, and a main power rail glitch or abnormal power disturbance.

Smoke detectors may require regularly or randomly pinging other detectors for typical setups where there is more than one detector in a home. Smart Smoke Detectors may therefore benefit from LWC to ensure reliable functionality. ADWC with a RA can also help detect both functional safety and security anomalies.

End of informative comment

13.4.1.4 Light Bulb in Smart Home

Start of informative comment.

Light bulbs can be divided into two categories: critical light bulbs and normal light bulbs. The difference between a critical light bulb and a normal light bulb is that the critical light bulb must function continuously and always provide safety or security, post provisioning.

A critical light bulb might be needed in a home to provide visibility for a security camera.

A critical bulb might be used with a photo sensor to trigger a chain of events that are critical to security or functional safety. So, a critical light bulb will need the same CRBB's as a Smoke Detector.

A normal light bulb in a smart home might not need the same CRBB's as a Smoke Detector but might need LWC to ensure it is continuously functioning. Failure of the normal light bulb is inconvenient but typically neither safety nor security critical.

End of informative comment

13.4.2 Smart Building

Start of informative comment.

Commercial and managed residential buildings have a higher set of security requirements than for a Smart Home. Fortunately, they have greater resources available for security.

End of informative comment

13.4.2.1 Exterior Door Lock in Smart Building

Start of informative comment.

The lock on an exterior door in a smart building is critical to the safety of tenants and the protection of their possessions. Therefore, strong security is needed against most kinds of compromise. A remote monitoring service is probably available so CRBBs that require remote monitoring can be used (ADWC). A WWC may be useful in addition to the ADWC to protect against malware that sleeps the lock and thus prevents the ADWC from working.

End of informative comment

13.4.2.2 Interior Door Lock in Smart Building

Start of informative comment.

The lock on an interior door in a smart building may be separating mutually distrustful tenants. When this is true, the security of this lock must be as strong as it would be for an exterior door. A remote monitoring service is probably available so CRBBs that require remote monitoring can be used (ADWC).

End of informative comment

13.4.3 Industrial Systems

Start of informative comment.

The CRBBs mentioned in this section augment and do not replace industrial security standards, such as the IEC 62443 series [6] [7] [8] [9] [10] [11] [12] [13] [14], which specifies security capabilities for control systems.

End of informative comment

13.4.3.1 Industrial PC

Start of informative comment.

Industrial PCs (IPCs) are computer systems for factory and industrial workloads which typically include process control, data acquisition, and/or human machine interface (HMI). The current trend is to consolidate a wide variety of industrial functions into a single IPC often with virtualization, increasing the criticality of IPCs. IPCs are rugged computer systems, which come in a variety of form factors, must provide higher dependability, wider range of operating conditions (e.g., extended temperature ranges) and safety. As IPCs control many critical infrastructure systems, strong security to withstand and recover from malicious attacks is important.

IPCs can be deployed with a central management system (such as a factory automation), but they can be also deployed in a remote location with intermittent connectivity or no connectivity to a central management system (such as oil & gas). In general, IPCs downtime should be minimized. When IPCs are deployed in a remote location with unreliable communication, they must be able to protect and recover themselves. Self-protection and self-recovery can reduce operating cost by avoiding manual intervention at this remote location.

An ADWC can be used for IPCs that are remotely managed and a WWC can be used for IPCs deployed in a remote location with unreliable communication. Generally, an LWC is undesirable for IPCs as it disrupts normal IPC functionalities which may not meet the high uptime requirements of IPCs.

In the future, ordinary PCs may contain the CyRes functionality and could be the basis of IPCs.

End of informative comment

13.4.3.2 Programmable Logic Controller (PLC)

Start of informative comment.

Programmable Logic Controllers (PLCs) control industrial systems, such as increasing or decreasing heating for a chemical reaction to maintain the proper temperature. Their proper functioning is essential to the proper operation of these systems. Deliberate sabotage of PLC operation can cause halted production, damage to systems, and even safety issues. Therefore, strong security is needed against compromise of these systems.

PLCs are usually centrally managed so CRBBs that require remote monitoring can be used (ADWC). Restarting the PLC is not desirable, as this may interfere with proper operation, though a forced and unexpected restart is generally preferable to compromised operation.

End of informative comment

13.4.3.3 Fail Safe Containment Alarm

Start of informative comment.
A containment alarm system might be designed to fail-safe in the event that leaks would cause injury or death. Such warning systems continually or frequently use audible or visual means to attract attention during normal operation, instead of doing nothing until/unless a leak occurs. These warning systems indicate an alarm condition by ceasing to attract attention, in order that any fault in the warning system itself is indistinguishable from an actual emergency. A malware attack could either cause disruption by inappropriately turning off the audible or visual signals or cause significant harm by continuing to produce the signals when a leak happens.

An ADWC may be suitable for a fail-safe containment alarm system. This is because the alarm system would principally be backup for remote monitoring of both containment and the containment alarm system. A WWC could help ensure that the containment alarm system is properly reactivated as soon as possible after unauthorized shutdown. An LWC seems inappropriate because a containment alarm system should never be unnecessarily automatically deactivated.

End of informative comment

13.4.4 Summary

Start of informative comment.

Table 7 has the scenarios in section 13.4 as rows. The columns are shaded green for building blocks that are relevant for each scenario. The table summarizes the usefulness of CRBBs for IoT scenarios.

IoT Scenario	Write-Protection Latch (WPL)	Read-Write-Protection Latch (RWPL)	Latchable Watchdog Counter (LWC)	Authorized Deferral Watchdog Counter (ADWC)	Wakeup Watchdog Counter (WWC)
Exterior Door Lock in Smart Home					
Interior Door Lock in Smart Home					
Smart Smoke Detector in Smart Home					
Normal Light bulb in Smart Home					
Critical Light bulb in Smart Home					
Exterior Door Lock in Smart Building					
Interior Door Lock in Smart Building					
Industrial PC (IPC)					
Programmable Logic Controller (PLC)					
Fail Safe Containment Alarm in Industrial Systems					

Table 7: Examples of CRBBs in IoT Devices

End of informative comment

