TRUSTED® COMPUTING GROUP

SPECIFICATION

Cyber Resilient Module and Building Block Requirements

Version 1.0
Revision 0.08
October 19, 2020

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

Contributors

Name	Organization
Matthew Areno	Intel Corp.
David Challener	John Hopkins University, Applied Physics Lab
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

CONTENTS

1 Scope

This specification defines capabilities that provide a foundation for Device Vendors to build Cyber Resilient Devices. Self-recoverable Cyber Resilient Devices are architected to be protected from network-based adversaries and, even if compromised, can be recovered without manual intervention. Symbiotic Cyber Resilient Devices require external assistance (inclusive of manual actions) to initiate recovery actions. Devices can be made up 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. To make the servicing capabilities general enough to be applicable across many different components and layers, this specification defines an abstract Cyber Resilient Module that has (1) an engine assumed to be uncompromised that does recovery and (2) a target that may be compromised and can be recovered by the engine. The requirements and recommendations in this specification can be used to protect the bottommost layer and support its ability to do recovery for the layer above it. The Cyber Resilient module concept can be applied repeatedly for each successive layer and for components to support recovery for every layer and every component in a device (except the bottommost layer). The engine and recoverable target architecture can protect and recover successive layers inductively.

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 compromises. To implement cyber resilient capabilities, the specification combines the building blocks together into an abstract computing component called a Cyber Resilient Module. 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)/SoC that is part of a component that is incorporated into a larger complex device. If the SoC or MCUs have numerous layers of code and configuration, the first and second layer could comprise a Cyber Resilient Module and the second and third layers could be another module and so on. A single Cyber Resilient Device may consist of one or many Cyber Resilient Modules.

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, like a 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 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 to perform local, remote servicing or both.

The mechanisms in this specification are well suited for helping to construct 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 device and use case specific functionality and optionally networked services to complete a finished device.

This specification contains a set of building blocks that illustrate how to achieve cyber resilient capabilities in an extensible framework. This specification anticipates that additional building blocks, including alternatives to the building blocks in this version of this specification, may be added later.

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 does 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.

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

1.1 Document Structure

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), what performs the recovery, and how recovery actions are authorized. Section 5 describes the relationship between what performs the servicing and what is serviced by grouping them together as a Cyber Resilient Module. It introduces the concept of "resetting" the Cyber Resilient Module so servicing actions can occur safely even if the portion of the module that can be serviced is uncooperative or compromised by malware. Resetting a Cyber Resilient Module could have implications for other components of a device, power management, data loss or other vendor specific concerns. A Resilience Orchestrator architectural element has the responsibility to make sure the integration between all the elements internal and external to a Cyber Resilient Module are well coordinated.

Section 6 introduces Cyber Resilient Building Blocks internal to a Cyber Resilient Module that support the relationship between what will be serviced and what does the servicing. It explains how building blocks can be used for their primary role in resilience and also provide other benefits. The section includes design guidance and consideration for using the building blocks.

Section 7 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 8 formally describes the building blocks and details their normative requirements.

For the scope of Cyber Resilient Modules defined in this specification (e.g. using "early boot" as a "safe" environment) Sections 9 and 10 provide context on how the primary architectural elements are intended to be used by a complete solution. The different flavors of Cyber Resilient Modules (self-recoverable versus symbiotic) have separate requirements. Section 10 includes additional recommendations and requirements for attestation and secure boot.

Section 11 defines solution profile nomenclature intended to be useful for Device Vendors to concisely describe concisely what capabilities their solution has implemented.

The Appendix has five informative sections. Section 12.1 provides a description of how Protection Latches can also be helpful to implement Roots of Trust for attestation and secure boot. Sections 12.2 and 12.3 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 12.4 discusses how the building blocks in this specification could be applied in different IoT scenarios.

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. [Accessed 31 July 2020].

[2] Trusted Computing Group, "TCG Glossary Version 1.1, Revision 1.0," 11 May 2017. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-glossary/. [Accessed 31 July 2020].

[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/. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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/. [Accessed 21 September 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

[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. [Accessed 19 October 2020].

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 (for example, a Watchdog Counter elapsing or the press of a physical button).
Authenticated Watchdog Counter	A Watchdog Counter that can only be deferred by providing a Deferral Ticket satisfying its Deferral Policy or using a protected capability.
Basic Watchdog Counter	A Watchdog Counter that can be cancelled or indefinitely deferred without special 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	A building block useful for supporting Cyber Resilient capabilities in a Cyber Resilient Module. (Examples: A Watchdog Counter or a Protection Latch)
Cyber Resilient Device	A device containing one or more Cyber Resilient Modules.
Cyber Resilient Module	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 Ticket	A cryptographically protected single use message that authorizes reconfiguration and/or restart of an Authenticated Watchdog Counter.
Deferral Policy	A policy specifying the requirements for generating and validating a Deferral Ticket.
Detection	A process of discovering when a device becomes compromised or begins manifesting aberrant behavior. Detection can be performed locally or remotely (for example, 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	A Watchdog Counter that cannot be deferred after it is Enabled without using a protected capability.
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 of 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	A Protection Latch that prevents reading when activated.
Read-Write-Protection Latch	A Protection Latch that prevents reading and writing when activated.
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 can authorize 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 Cyber Resilient Building Blocks to form a Cyber Resilient Module. For example, by resetting all the building blocks at the same time.
Resilience Target (RT)	A mutable engine that is serviceable by one or more Resilience Engines. The RT cannot include any of its REs.
Secure Execution Environment	A Cyber Resilient Building Block that ensures that the same program code with the same configuration data produces the same result (see section 8.1 for more details).
Self-Recoverable Cyber Resilient Module	A Cyber Resilient Module that has no external dependencies to initiate recovery actions.
Symbiotic Cyber Resilient Module	A Cyber Resilient Module that has external dependencies that need to assist in initiating 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	A Latchable Watchdog Counter or an Authenticated Watchdog Counter that is unaffected by power management performed by the RTs in the same Cyber Resilient Module.
Watchdog Counter (WDC)	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, Authenticated or Wakeup Watchdog Counter.
configuring	Setting of a WDC's Enable/Disable state, expiration value and/or other parameters.
deferral	A reconfiguration of a WDC that delays the expiration event.
expiration value	The period of time or number of events that will elapse before a WDC generates an Attention Signal.
Write-Protection Latch	A Protection Latch that prevents writing when activated.

3.2 Abbreviations

Acronym	TERM
ATA	Advanced Technology Attachment
AWDC	Authenticated Watchdog Counter
BMC	Baseboard Management Controller
BWDC	Basic Watchdog Counter
CDI	Compound Device Identity
CPU	Central Processing Unit
CRBB	Cyber Resilient Building Block
DoS	Denial of Service
IoT	Internet of Things
KDF	Key Derivation Function
LWDC	Latchable 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
TPM	Trusted Platform Module
UDS	Unique Device Secret
WDC	Watchdog Counter
WPL	Write-Protection Latch

4 Introduction

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. 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 IoT revolution will increase the number of computing devices by orders of magnitude. These devices will be built from the same imperfect software in use today, but manual remediation will be much less practical or even unfeasible 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.

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.

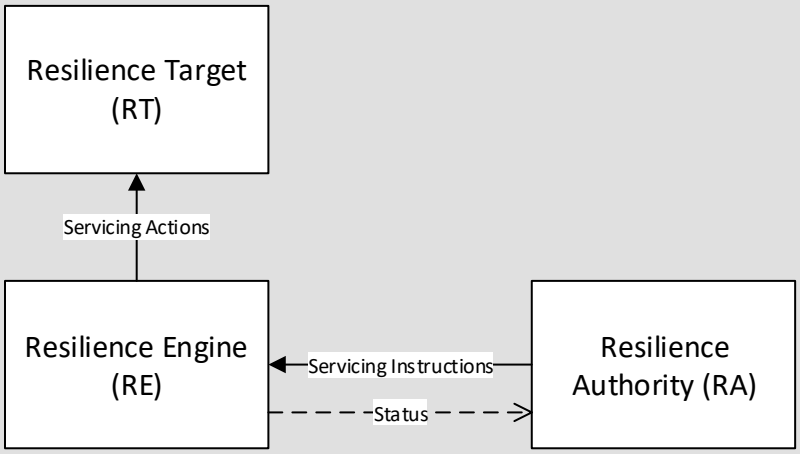

Figure 1: Relationships between the Resilience Target, Resilience Engine, and Resilience Authority

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 an 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 an 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 on an RT, but it does include normative requirements for support for attestation of the RT in section 10.3.1. 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 an RT is resilient by providing the capability to reliably service and restart an RT due to:

- An actual loss of integrity of the persistent image of the RT, including code and configuration.
- Possible 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.

The RT servicing actions could result in small or large changes to the persistent stored image of the RT (for example, 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 machines, 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 an 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 Cyber Resilient Module, the RE requires direct access to the RT(s) to allow servicing of the RT(s). 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 possible 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.

An 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 Cyber Resilient Module 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) –Isolates the environment the RE uses from interference from its RTs such that it is not subverted by any previous or concurrent RT actions on the device whether direct or indirect (see section 8.1 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 an 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, 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 the Root of Trust for Reporting (RTR) to attest the RT locally or remotely.
- Follow other instructions from an RA: e.g., quarantine the RT module
- Take defensive measures to reduce potential damage if the RT is suspected of being infected
- Implement defensive policies if communication (direct or indirect) with its RAs is not available
- Update the RE's own code, configuration and policy

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. Though not explicitly discussed elsewhere in this specification, it is conceivable a layered architecture could repeat the paradigm by designating the RE of an upper layer RT as the RT for a lower layer RE.

This specification does not mandate Detection capabilities on an RE, but it does include normative requirements to support for attestation of the RE.

End of informative comment

5.3 Resilience Authority (RA)

Start of informative comment

Attack techniques are always improving, and solutions 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 solutions, multiple approvers may need to agree to servicing actions.

This specification abstracts the complex set of activities that can occur before the final decision to conduct servicing actions by defining a Resilience Authority (RA) as an entity (e.g., Device Vendor or owner-operated cloud management service, IT administrator, owner, 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 an 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 an 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 abstracts the details by having one or more RAs associated with a RE with each RA independently able to authorize servicing actions.

The RA is expected to continue functioning and approve recovery strategies when an 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's credentials 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, an 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 an RE depends, at least partially, on an 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 possible RT compromise.

End of informative comment

5.4 Cyber Resilient Module and Module Reset

Start of informative comment

The primary interactions between the RA and RE and between the RE and RT are shown above in Figure 1. 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 Resilience Authority.

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

- The RE is device firmware and the RT is device software (both including additional hardware resources)
- The RE is a software layer before the RT in the boot sequence (both including additional hardware resources)
- The RE is an OS and the RT is an application (both including additional hardware resources)
- 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) protection of the RE execution environment from the RT. 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 Cyber Resilient Module as all of the components needed to enable a Device Vendor to construct an RE and 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. An illustration is shown in Figure 2.

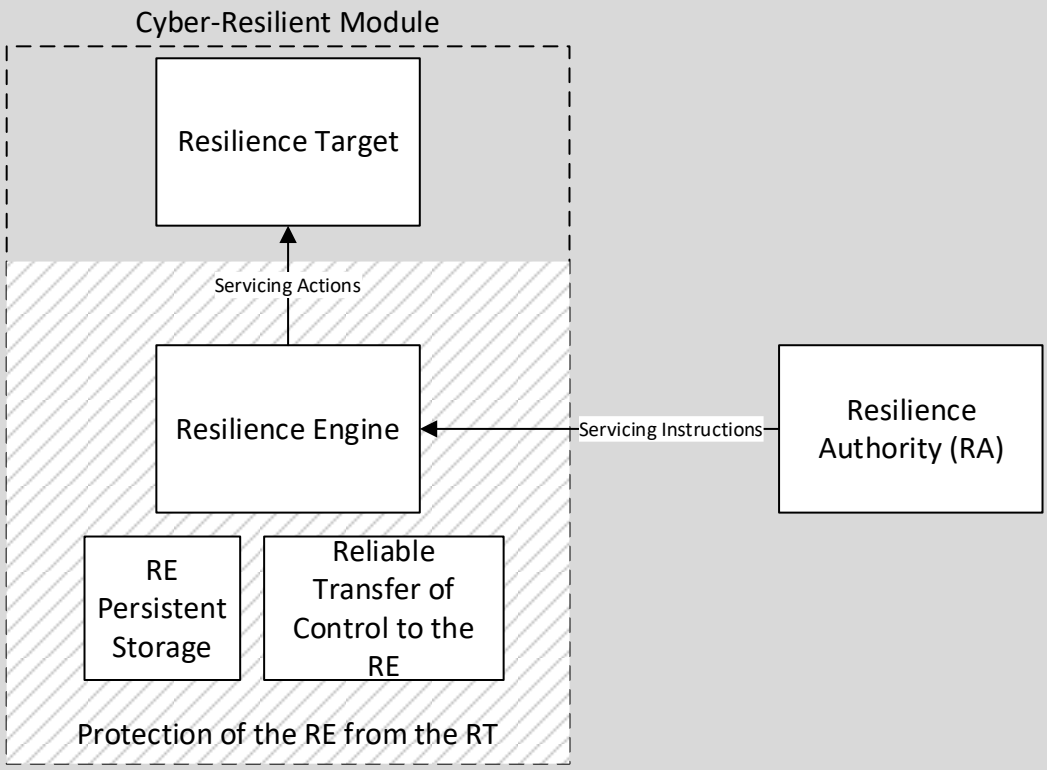

Figure 2: Cyber Resilient Module

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

A reset of a Cyber Resilient Module does not necessarily correspond to a reset of an entire device, although it could for some implementations. The reset of a Cyber Resilient Module means resetting all components internal to the module. Later in this specification in sections 6 and 8, each of the capabilities supporting the Cyber Resilient Module are defined as individual building blocks, each receiving an Initialize Signal that resets them. The term Module Reset is defined as the reset of a Cyber Resilient Module that sends an Initialize Signal to all of its internal Cyber Resilient Building Blocks and transfers control to the RE before the RT.

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

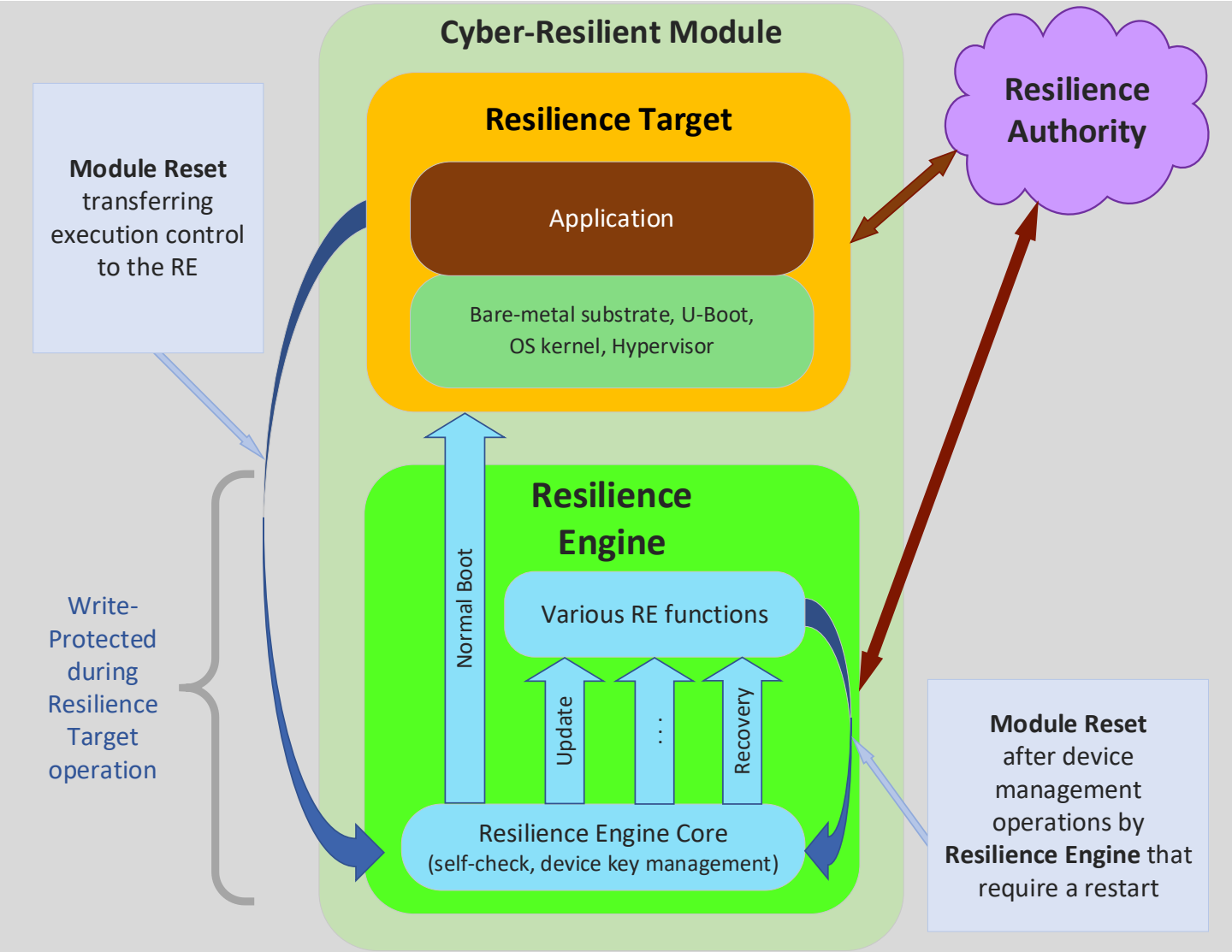

Figure 3: Example Cyber Resilient Module using the building blocks described in this specification

A complex device may include multiple Cyber Resilient Modules some of which can be reset without resetting the others. For example, a PC Client device could have a first Cyber Resilient Module 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 with the OS software and host CPU being the RT. Additional Cyber Resilient Modules 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 both including the subcomponent's microprocessor and other computing resources). While it could be possible to reset subcomponent Cyber Resilient Modules independently, if the host CPU Cyber Resilient Module is reset, it would likely require resetting all of the Cyber Resilient Modules in the device.

A simpler device having only a single CPU could also have multiple Cyber Resilient Modules. The first Cyber Resilient Module 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 Cyber Resilient Module might be designed with the second stage boot loader used by its RE with any software that runs later used by its RT. In this example, it would not be possible to perform Module Reset on either Cyber Resilient Module individually.

End of informative comment

5.5 Resilience Orchestrator (RO)

Start of informative comment

The primary elements above (RT, RE, RA and the Cyber Resilient Module) 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 primary component called the Resilience Orchestrator (RO) is defined as the component that coordinates the RT, RE, Cyber Resilient Building Blocks and integrates with the environment the Cyber Resilient Module lives in. It receives and processes notifications when the RE needs to execute and acts by resetting building blocks, performing Module Reset, and invoking the Resilience Engine. 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 Cyber Resilient 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 a Watchdog Counter reaching its expiration value or other initiator.

End of informative comment

6 Cyber Resilient Building Blocks Introduction

Start of informative comment

This specification includes requirements for basic Cyber Resilient Building Blocks and guidance intended to support Device Vendors or other specifications that provide precise requirements to recover RTs on specific device types. Though building blocks defined in this specification can be utilized individually, they are most effective when combined to deliver complete cyber resilient scenarios.

End of informative comment

6.1 Secure Execution Environment for Resilience Engine

Start of informative comment

A *Secure Execution Environment (SEE)* is a capability that supports the execution of a program in such a manner that any other program that has been executed prior to or is being executed concurrently with program execution on the same or different component of the Cyber Resilient Device cannot affect initial conditions and operations of the program executed in the SEE.

This specification defines and is limited to capabilities that support Temporal Protection of the Resilience Engine, with the Module Reset event initializing the SEE and passing control to the RE before the RT. The RE code is executed (possibly using the same hardware resources as the Resilience Target) and completes all its resilience related activities, applies all the necessary protections to its sensitive code and data, and only then initializes the RT and passes control to it.

In order to establish a SEE of this kind when the RE and RT potentially share major hardware capabilities (such as a CPU, persistent storage and RAM) the Module Reset functionality has to satisfy a particular set of requirements (see Section 8.1.1).

End of informative comment

6.2 Signals and Programmatic Interfaces

Start of informative comment

The Cyber Resilient Building Blocks 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 Resilience Engine and/or the Resilience Target. 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 cryptographic 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 Resilience Engine 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 Resilience Engine may be implemented in the form of code and associated configuration data that have to be kept in device’s persistent storage. Since an untrusted Resilience Target may share the same hardware resources as the RE, special precautions may be required to protect the RE and its associated data.

A Resilience Engine may need storage protection capabilities for two primary purposes. First, read protections are needed to prevent secret values used by the RE from being read by the RT. Second, write protections for the code and critical configuration data of the RE are necessary to prevent modification after the RE passes control to the RT. This section describes capabilities that the RE can use to protect itself and its data/configuration from a potentially adversarial 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 is a protected capability that allows read access to one or more ranges 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 Read-Protection Latch 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 (i.e. a backup RT image)

When the RE receives control after Module Reset, it should be able to perform an update of the Resilience Target, and possibly its own resources (e.g. firmware, software, configuration, etc.). The RE may need to be updated in order 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~~ configuration~~s~~ 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 – 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 Write-Protection Latches are activated.

End of informative comment

6.4 Attention Signal Generators

Start of informative comment

While the Resilience Engine 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, getting control may be impossible in some cases. For instance, when the Resilience Target is hung because of a software bug, or when it is compromised with malware preventing 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 – a mechanism that results in an unconditional transfer of execution to a Resilience Engine. Attention Signals can be generated remotely (e.g. by a Resilience Authority), 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

6.4.1 Watchdog Counters

Start of informative comment

Watchdog Counter (WDC) 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 (BWDC)
- Latchable Watchdog Counter (LWDC)
- Authenticated Watchdog Counter (AWDC)
- Wake-up Watchdog Counter (WWDC)

If a WDC is implemented in such a way that once configured and enabled, it cannot be disabled or reconfigured by the RT or malware, it is considered cyber resilient.

End of informative comment

6.4.2 Basic Watchdog Counter

Start of informative comment

A Basic Watchdog Counter (BWDC) is a WDC that can be serviced at any time. It may be indefinitely deferred. A BWDC is useful for increasing reliability but malware may interfere with it. Because a BWDC that is accessible to the RT can be indefinitely deferred by malware, a BWDC cannot be relied on to start a recovery process if malware did compromise the RT. The BWDC has fewer security features than other Watchdog Counters in this specification.

End of informative comment

6.4.3 Latchable Watchdog Counter

Start of informative comment

A Latchable Watchdog Counter (LWDC) is a WDC that, once enabled, cannot be disabled, or deferred. Thus, a Resilience Engine can use an LWDC to ensure that control returns to it periodically even in the presence of malware. In contrast to a BWDC that only fires when the device is hung, the LWDC triggers during normal device execution, which may interfere with the device's normal operation.

End of informative comment

6.4.4 Authenticated Watchdog Counter

Start of informative comment

An Authenticated Watchdog Counter (AWDC) is a WDC that can be deferred programmatically only if a cryptographically protected single use Deferral Ticket is presented to it. In contrast to a BWDC that can be indefinitely deferred by malware, an AWDC 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 issuing Deferral Tickets. In the absence of the Deferral Ticket, the AWDC generates an Attention Signal which can cause a Module Reset and give the RE an opportunity to perform any necessary remediation.

End of informative comment

6.4.5 False Alarms

Start of informative comment

Cyber resilient WDCs may themselves disrupt device availability by triggering false alarms occasionally (in case of an AWDC) or periodically (in case of an LWDC). In light of these considerations, the Cyber Resilient Watchdog Counters described in this specification will not be appropriate in all use cases. When they are used, their expiration values should be picked so as to balance the recovery latency with the operational requirements of the device (tolerance to its service interruptions). If Cyber Resilient Watchdog Counters are *not* used, then Device Vendors may consider employing alternative solutions to generate Attention Signals beyond those described in this specification or may rely on human interaction.

Another consequence of cyber resilient WDCs producing false alarms is that one of the primary responsibilities of the RE is to make an authoritative determination of whether remediation is indeed required, rather than immediately taking corrective actions. A well-protected RE can perform relatively sophisticated health assessments. Such assessments may involve both local firmware code and configuration analysis, as well as communication with an RA. At the same time, the RE code is of critical importance for the Cyber Resilient Device and is expected by the threat model to be bug-free (section 7). Therefore, a prudent balance between the RE sophistication and simplicity must always be preserved.

End of informative comment

6.4.6 Loss of Connectivity

Start of informative comment

Reliance on cloud control by an 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, in this case, instructions or assistance from the RA are unavailable to the RE.

A simple way for an RE to react to such situations 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 solutions are possible. For example, in a design where occasional loss of connectivity is anticipated, the solution could allow a physically present person to press a button that delays Watchdog Counters from generating Attention Signals by sending an Initialize Signal that results in deferral.

End of informative comment

6.4.7 Wakeup Watchdog Counter 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, it could attempt a denial of service attack by putting a Cyber Resilient Module or an entire device in a power state that is unresponsive. If an RE relies on a Latchable Watchdog Counter or an Authenticated Watchdog Counter that has a dependency on the power state of a device or its Cyber Resilient Module 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 on a human operator of the device to notice the device is stuck in a low power state and use manual actions (for example, 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 Cyber Resilient Module 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 Cyber Resilient Module persistently unresponsive.

A third solution is to make the functioning of a Latchable Watchdog Counter or an Authenticated Watchdog Counter independent of power states the RT controls. For example, a Latchable Watchdog Counter 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. A Latchable Watchdog Counter or an Authenticated Watchdog Counter whose functioning cannot be interfered with by the RT's ability to do power management is called a Wakeup Watchdog Counter (WWDC).

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

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

Depending on the architecture of a device, it may have multiple independent Cyber Resilient Modules, each with their own power states and power management. Simpler solutions may consist of multiple Cyber Resilient Modules 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 Cyber Resilient Modules might need their own individual mitigation against denial of service attack attempts if their RT is compromised.

End of informative comment

7 Threat Model

Start of informative comment

The threat model of this architecture is built around four entities: The Resilience Engine, Resilience Target, Resilience Authority, and Resilience Orchestrator. 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 an RT fault 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 an RT fault caused by a bug in its implementation.

With regard to intentional attacks, 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, powering off a subcomponent or the entire device
- A compromised RT generating an Initialize Signal to Watchdog Counters or Protection Latches internal to the Cyber Resilient Module 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

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 of their hardware. The RT is considered vulnerable to attacks and thus unreliable at any time. The Secure Execution Environment and the functioning of the RE are required to be isolated from any interference by the (potentially compromised) RT.
- Between an RA and REs 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 using properly secured channels between the RE and the RA, (b) using untrusted channels (for example, through the RT) with some other mechanism to validate authenticity (for example, data signed by the RA with rollback protection), or other schemes.
- The generation of Initialize Signals for Watchdog Counters and Protection Latches internal to the Cyber Resilient Module.
- The propagation of Initialize Signals from source to destination.

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

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 (Section 6.4.6).

End of informative comment

8 Cyber Resilient Building Block Requirements

Start of informative comment

This section contains normative requirements for Cyber Resilient Building Blocks (CRBBs). It adds to the informative introduction to cyber resilient building block capabilities in Section 6.

End of informative comment

8.1 Secure Execution Environment

Start of informative comment

A *Secure Execution Environment* (SEE) ensures that the same program code with the same configuration data produces the same result disregarding the state of the device caused by previously executed code and concurrent device component activities.

General examples of a Secure Execution Environment are: (1) running Read Only Memory (ROM) code just after a power cycling of 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.

End of informative comment

8.1.1 Module Reset Established Secure Execution Environment

Start of informative comment

This specification uses a Secure Execution Environment (SEE) to protect the Resilience Engine. The RE runs in the SEE. In this specification, the SEE is established in early boot as the result of Module Reset, but other options may be added in the future.

The approach described by this specification is the *Temporal Protection* of the Resilience Engine. This method ensures protection of the RE by requiring that the Secure Execution Environment is initialized, and the RE is executed and completes execution, before the Resilience Target is allowed to run.

This is in contrast to *Spatial Protection* in which a SEE provides physical isolation of the RE from the RT and any other concurrent device component activities. Though only the Temporal Protection has normative requirements in this specification, both variants of a SEE can provide adequate protection for the RE.

For Cyber Resilient Modules implementing Temporal Protection of the RE, it is assumed that the device will (1) perform Module Reset as part of the power on sequence and/or (2) have at least one Attention Signal the result of which is Module Reset.

This specification does not address devices not using Temporal Protection. For example, devices that instead rely on physical isolation of the RE from the RT and other concurrent device component activities typically have Attention Signals that result in guaranteed execution of the RE without triggering a reset.

Module Reset establishes a Secure Execution Environment for the Resilience Engine regardless of the prior execution state of the device. Establishment of a SEE will typically involve resetting internal registers and caches. If the device incorporates independent programmable processors or devices that can affect memory or the processor complex (i.e., bus-mastering devices), then these devices must also be reset during Module Reset or quiesced until brought back online by the Resilience Engine.

Actions constituting Module Reset are performed during power-on, and Module Reset can be initiated both voluntarily (e.g. by device hardware, software, etc.) and involuntarily (e.g. when a Watchdog Counter expires).

Device Vendors may optionally allow RAM contents to survive Module Reset. This allows the RE to work cooperatively with the Resilience Target 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 and resume the suspended RT, thereby avoiding longer boot process.

Device designs need to prevent any indirect way for a RT to interfere with a RE. For example, it cannot be possible for an RT to reprogram another component that subverts the 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.) In such a situation a solution could be preventing the RT from communicating with the other component or, alternatively, implementing Module Reset to force bus mastering devices to be inactive until the RE enables them.

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.

To ensure the RE is executed without interference from the RT, the Module Reset event initializes the SEE and passes control to the RE. 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 sensitive code and data, and only then passes control to the RT.

In order to establish a SEE of this kind when the RE and RT share major hardware capabilities (such as a CPU, persistent storage and RAM) the Module Reset functionality has to satisfy a particular set of requirements (see Section 8.1).

Because the RE may consume data and parameters from untrusted sources (e.g., persistent storage accessible to the RT), the SEE cannot prevent all possible attacks. Thus, the RE is always responsible for careful validation of any such inputs, disregarding the type of its SEE implementation.

Some implementations may rely on the RT to cooperatively transfer patches from the RA to the RE when everything is functioning normally. This helps to minimize disruptive recovery scenarios to only situations where the RT is actually compromised or some other failure occurs. To support the RT passing (untrusted) information to the RE, it is desirable for there to be a way for the RT to pass input data to the RE. For example, by placing the data in volatile memory and resetting the device during a scheduled downtime window.

End of informative comment

Requirements for establishing a Temporal Secure Execution Environment for Resilience Engine execution using Module Reset:

1) As a result of Module Reset:
   a) All CRBBs internal to the Cyber Resilient Module SHALL receive an Initialize Signal
   b) The Secure Execution Environment SHALL be initialized so that none of the prior execution state, except code and data explicitly validated by the Resilience Engine, can affect the RE behavior
   c) Any code that executes before the RE SHALL be validated or loaded from locations able to be protected from the RT
   d) Sufficient Cyber Resilient Module state SHALL be initialized
2) The RE SHALL be validated or loaded from storage locations able to be protected from the RT
3) The RE SHALL be invoked before the RT
4) Establishment of the SEE SHOULD provide a way to pass input data to the RE
5) The cause of a Module Reset (power-on, a WDC Attention Signal, or programmatically initiated) SHOULD be recorded and accessible to the RE.
6) Module Reset SHALL be the only way to send Initialize Signals to any of the CRBBs internal to the Cyber Resilient Module

8.2 Signals

Start of informative comment

This section contains normative requirements for Signals.

There are no normative requirements for Programmatic Interfaces.

End of informative comment

8.2.1 Attention Signals

Start of informative comment

An Attention Signal is an output signal for attention generated by a component (for example, a Watchdog Counter elapsing or the press of a physical button). In the context of how an Attention Signal is used, there may not be a need for a destination to act on the signal. For example, a destination asleep in a low power state or busy dealing with a higher priority 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. Note: There are requirements in section 9.4.2 that describe how an RO must act upon attention signals it receives.

End of informative comment

Normative requirements for Attention Signals:

1) An Attention Signal generated by a source SHALL be transmitted to its destination.

8.2.2 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 Cyber Resilient Building Block.

An Initialize Signal received by a destination may result in a lower security configuration (for example, 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 is provided in section 9.4.1.

End of informative comment

Normative requirements for Initialize Signals:

1) Only sources designated by the Device Vendor SHALL be capable of generating an Initialize Signal.
2) An Initialize Signal generated by a source SHALL be transmitted to its destination.

8.3 Protection Latches

Start of informative comment

Section 6.2 provides an overview of storage protection mechanisms applicable to Cyber Resilient Module designs and discusses the use of Protection Latches. This section contains normative requirements for an abstract basic building block called a Protection Latch which prevents storage read and write actions when active. For readability, this section uses the word “latch” interchangeably to refer to the term Protection Latch. The latch prevents read or write actions to provide protection for stored data.

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

The 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 scope of the 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

8.3.1 Protections for Storage

Start of informative comment

When active, the 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 the latch is active and does not occur when the latch is inactive.

End of informative comment

8.3.2 Initial State

Start of informative comment

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

Note: The initial state for the 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

8.3.3 Active State and Protections

Start of informative comment

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

End of informative comment

8.3.4 Latch Reset

Start of informative comment

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

End of informative comment

8.3.5 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 range of storage protected against write actions is an untrusted environment appending an error log. After writes to storage that append a new log entry, an untrusted environment could dynamically increase the area 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 ranges which could create device issues such as a denial of service.

End of informative comment

8.3.6 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

8.3.7 Indication of Blocked Actions

Start of informative comment

When blocking occurs, the result is Device Vendor defined. The result of a blocked action could (for example) 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 the 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

8.3.8 Protection Latch Abstract Diagram Example

Start of informative comment

Figure 4 illustrates Protection Latch implementation choices that could be helpful for a reader to better visualize the Protection Latch interfaces and state transitions. Optional items are indicated in italics text followed by an asterisk. Table 1 illustrates the meanings of the different Latch States. Table 2 illustrates the meaning of the different actions associated with the Protection Latch. Table 3 illustrates configuration parameters and values that an implementation might use to represent a Protection Latch configuration.

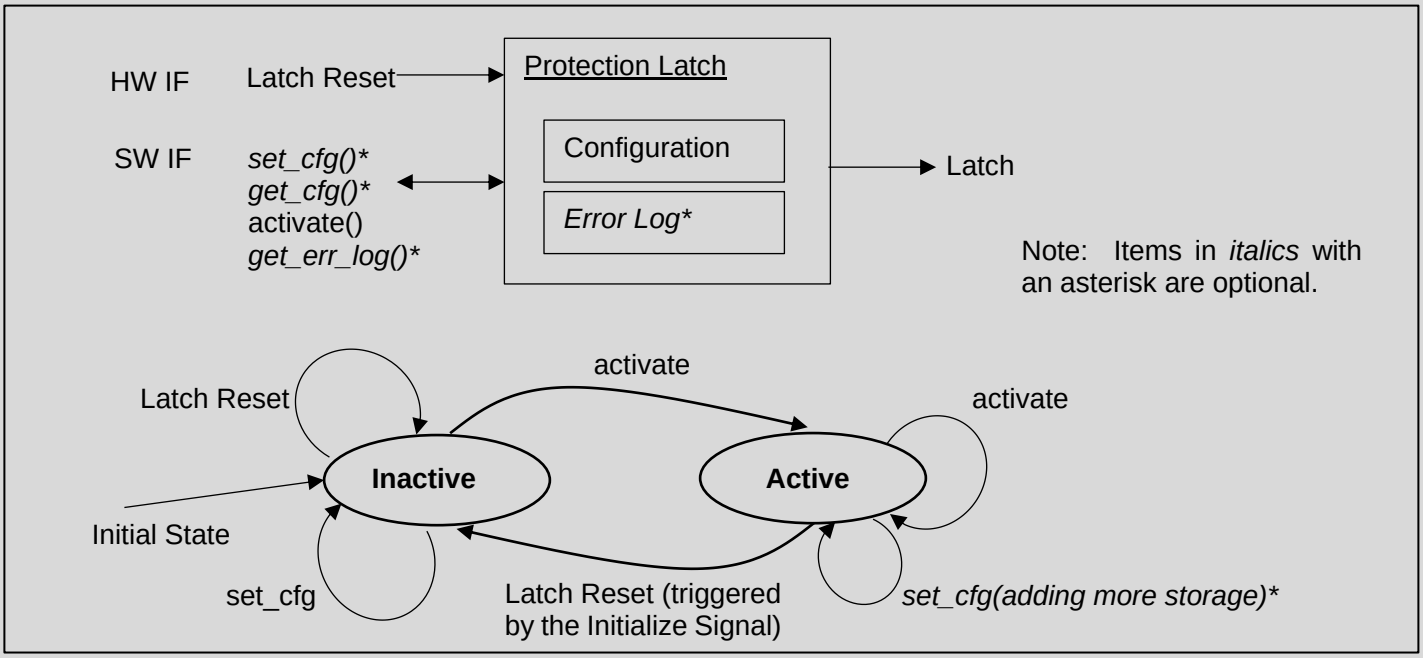

Figure 4: Protection Latch Block and State Diagram

State	Description
Inactive	Latch does not block, All APIs are enabled, Set configuration
Active	Latch does block, Set_cfg() is limited

Table 1: Protection Latch States

Action	Description
Initialize Signal	Reset Latch and transition state to inactive
Latch	If active access is blocked to the storage regions associated with the latch If inactive access is allowed to the storage regions associated with the latch

Table 2: Protection Latch Signal Interface

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

8.3.9 Protection Latch Requirements

A Protection Latch is an access-control mechanism for actions on a storage resource that meets the following normative 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 be fixed when the latch is in the Active State with the exception that the bound storage range 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 storage metadata about location(s) it protects.
13) If a Protection Latch supports reconfigurability then it SHALL provide an interface for querying storage metadata about 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 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.

8.4 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 assumed by this specification, see Section 6.4.1. This section describes the various WDC types and provides requirements for each.

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

End of informative comment

8.4.1 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 evocation of a Resilience Engine. To provide resilience, a Resilience Engine needs to be invoked when:

- a device malfunctions (for example, “freezes” or stops responding to a Resilience Authority)
- malware gains control of a Resilience Target and tries to
  - delay or prevent servicing actions through denial of service attacks, including putting the Cyber Resilient Module into low power states
  - mimic an uninfected device
- preventative patching or software updates need to be deployed
- power loss occurs during an update process
- et cetera

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

8.4.2 Basic Watchdog Counter

Start of informative comment

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

When enabled, the BWDC generates an Attention Signal when its current count reaches its expiration value. The default state for a BWDC can be enabled or disabled. While disabled, it can be configured with an expiration value and enabled. After a BWDC is enabled, it can be cancelled or reconfigured with a new expiration value. A

reconfiguration of the BWDC that increases the time or number of events remaining to reach the expiration value is called a “deferral”.

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

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

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

A BWDC could use either a time-based clock signal or a non-time-based mechanism, e.g. a BWDC 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 8.4.3 for an example diagram which illustrates a possible design for a BWDC implementation.

End if informative comment

Normative requirements for the BWDC:

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

8.4.3 Basic Watchdog Counter (BWDC) Abstract Diagram Example

Start of informative comment

This section provides an abstract diagram of a Basic Watchdog Counter (BWDC) to help visualize how the normative requirements might fit together to implement various types of cyber resilient Watchdog Counters. This is just one example of an abstract diagram; other abstract diagrams based on the normative requirements are possible. The diagrams provided here are merely illustrative and do not imply any specific required implementation. Device Vendors are free to build Watchdog Counters in whatever way best fits their architecture. Implementations could be based in hardware, firmware, software, or any combination.

Figure 5 is an abstract diagram of a BWDC based on the normative requirements in Section 8.4.2. Numbered circles in the diagram refer to the corresponding numbered normative requirement, e.g. number circle 1 refers to normative requirement #1 in Section 8.4.2. Solid lines and black numbered circles refer to SHALL requirements.

Dashed lines and the gray numbered circle refer to SHOULD or MAY requirements or where the Device Vendor has implementation discretion (for example, using an internal or an external clock). The blue rectangle encloses resources which are part of the BWDC. Lines going into or out of the blue rectangle are external Signals, events or Programmatic Interfaces.

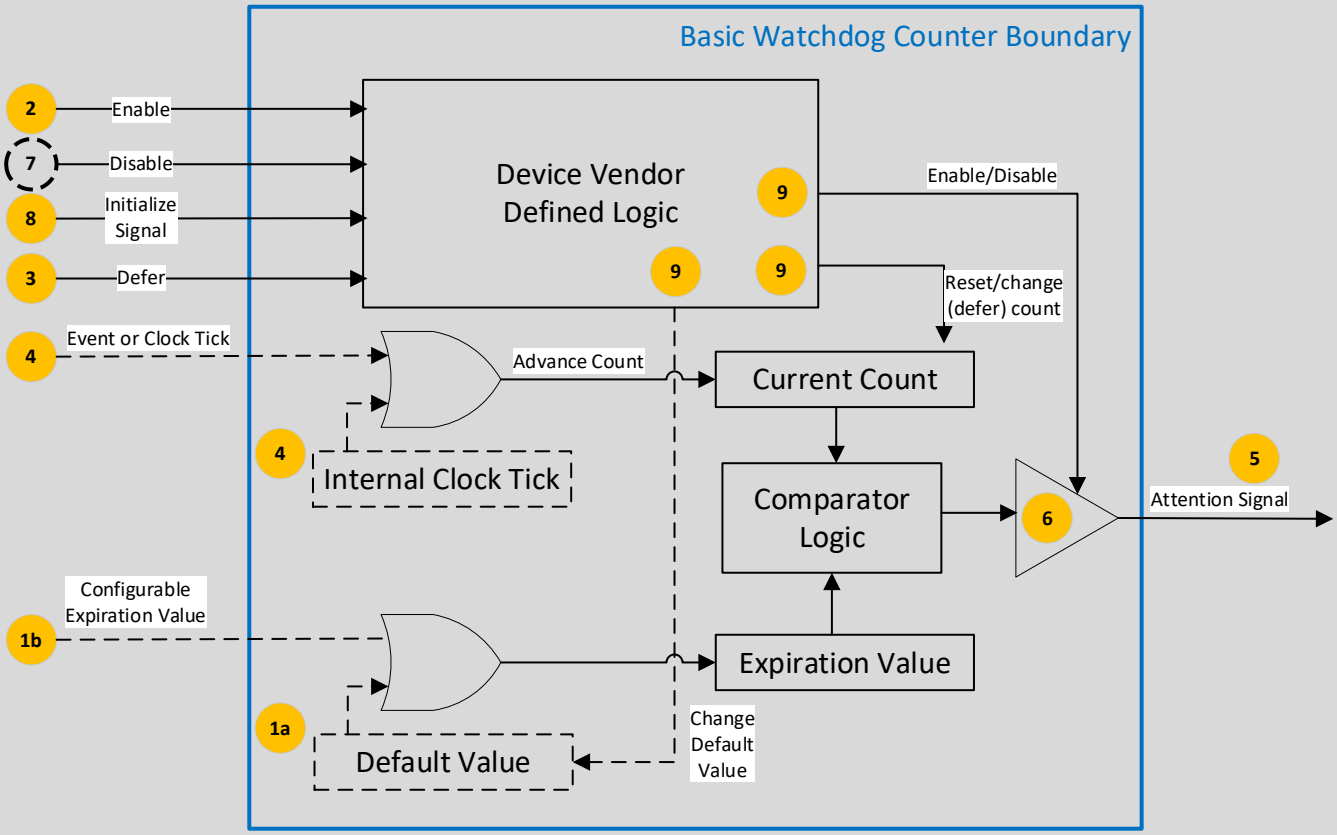

Figure 5: Basic Watchdog Counter Block Diagram

Figure 6 is a state diagram for a Basic Watchdog Counter. For simplicity, the transitions for the Initialize Signal are not shown. If shown, the Initialize Signal transition could start and end at either state.

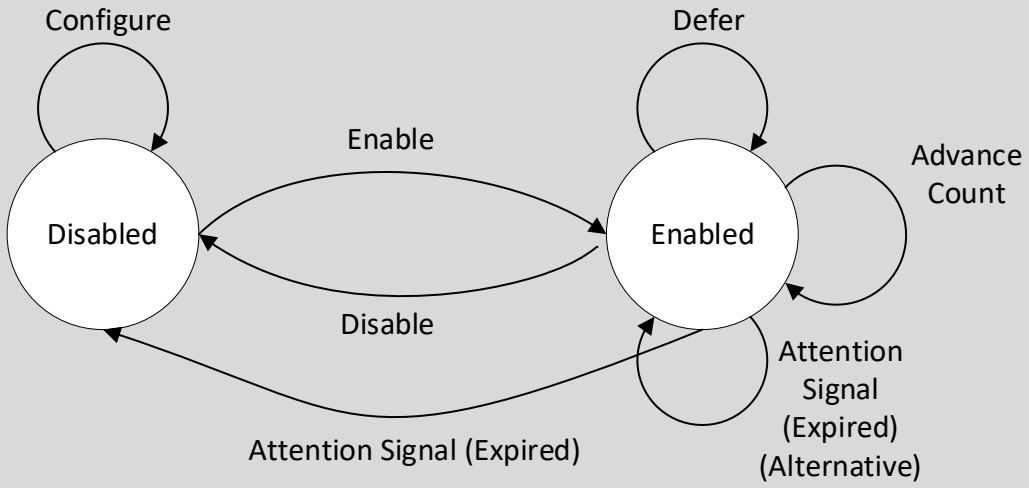

Figure 6: Basic Watchdog Counter State Diagram

End of informative comment

8.4.4 Latchable Watchdog Counter

Start of informative comment

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

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

The Initialize Signal has the effect of disabling the LWDC. The ability to generate the Initialize Signal needs careful protection and needs to be distinct from how Programmatic Interfaces interact with the LWDC.

This specification does not support changing the expiration value after the LWDC is enabled.

As with the BWDC, a LWDC 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 a LWDC is based on so adopters can determine whether the protections are adequate for their use case.

See Section 8.4.5 for an example diagram that illustrates a possible design for a LWDC implementation.

End of informative comment

Normative requirements for the LWDC:

1) A LWDC SHALL have one or both of:
   a) Default expiration value
   b) A configurable expiration value
2) There SHALL be a way to enable the LWDC.
3) When enabled, the only ways to disable the LWDC SHALL be via an Initialize Signal and (optionally) generation of its Attention Signal.
4) When enabled, internal or external events or clock ticks SHALL advance the current count towards the expiration value.
5) An enabled LWDC SHALL generate an Attention Signal when expiration occurs.
6) A disabled LWDC SHALL NOT generate an Attention Signal.
7) A LWDC SHALL accept an Initialize Signal.
8) The Device Vendor SHALL define the LWDC states resulting from its receipt of an Initialize Signal and after its generation of an Attention Signal.

8.4.5 Latchable Watchdog Counter (LWDC) Abstract Diagram Example

Start of informative comment

This section provides an abstract diagram of a Latchable Watchdog Counter (LWDC) to help visualize how the normative requirements might fit together to implement various types of cyber resilient Watchdog Counters. This is just one example of an abstract diagram; other abstract diagrams based on the normative requirements are possible. The diagrams provided here are merely illustrative and do not imply any specific required implementation.

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

Figure 7 is an abstract diagram of a LWDC based on the normative requirements for a LWDC. Numbered circles in the diagram refer to the corresponding numbered normative requirement, e.g. number circle 1 refers to normative requirement #1 in section 8.4.4. Solid lines and numbered circles refer to SHALL requirements. Dotted lines indicate the Device Vendor has implementation discretion. The blue rectangle encloses resources which are part of the LWDC. Lines going into or out of the blue rectangle are external Signals, events or Programmatic Interfaces.

.

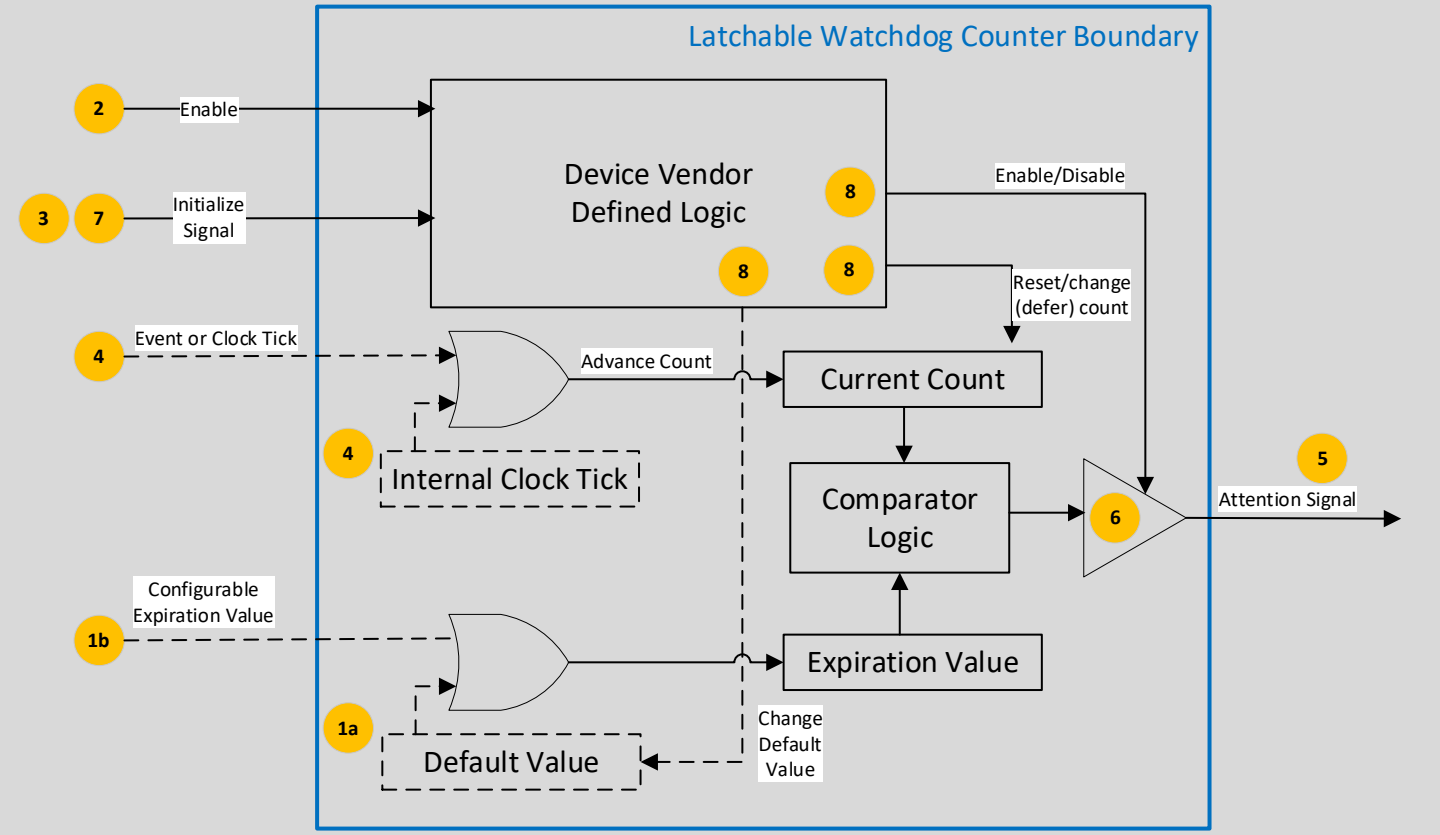

Figure 7: Latchable Watchdog Counter block diagram

Figure 8 is a state diagram for a Latchable Watchdog Counter. For simplicity, the transitions for the Initialize Signal are not shown. If shown, the Initialize Signal transition could start and end at either state.

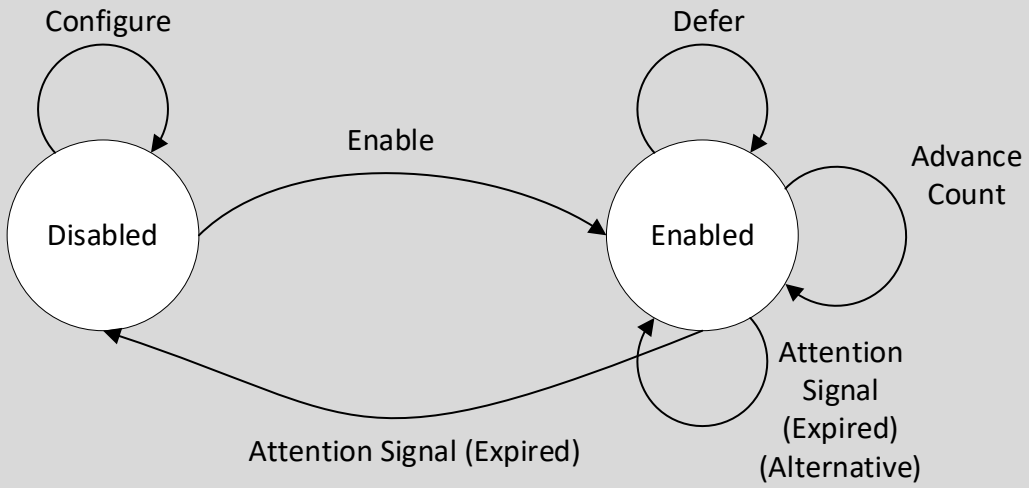

Figure 8: Latchable Watchdog Counter state diagram

End if informative comment

8.4.6 Authenticated Watchdog Counter

Start of informative comment

An Authenticated Watchdog Counter (AWDC) is a WDC that requires authenticated access in order to be deferred or reconfigured.

This specification requires any programmatic deferral and other configuration changes to be cryptographically authorized when the AWDC is enabled. The strength of function of the cryptographic algorithms and protocols is left up to the Device Vendor.

An AWDC can support non-programmatic authorization by using one or more Initialize Signals. The AWDC allows a Device Vendor to define more than one Initialize Signal, each with their own result. For example, an AWDC could receive one Initialize Signal for power cycle events that results in the AWDC returning to its default state and a second Initialize Signal for the user pressing a physical presence authorization button that results in deferral of the AWDC by sending an electrical signal.

The most common activity for an AWDC is programmatic deferral. A Deferral Policy can be configured, and deferral can happen repeatedly each time a valid cryptographically secured single use message, called a Deferral Ticket, is programmatically provided to the AWDC.

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 AWDC.

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

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

An AWDC can be implemented without persistent storage for its configuration, state and/or Deferral and Configuration Policies. For example, the values could need to be set explicitly each time the AWDC receives an Initialize Signal.

An AWDC can be implemented with persistent storage for its configuration, state and/or Deferral and Configuration Policies. For implementations that always require authorization to change the configuration, care should be taken to consider how it will be set initially and to avoid situations where the authorization is lost, unless that is appropriate for the use case.

An AWDC can have a default expiration value. An AWDC is required to have a configurable expiration value so it is flexible enough to use for different use cases.

This specification does not specify any specific protocols or cryptographic solutions, but the schemes used for deferral (and optional re-configuration) must be resistant to replay and man-in-the-middle attacks.

End of informative comment

Normative requirements for the AWDC:

1) An AWDC SHALL have one or both of:
   a) Default expiration value
   b) A configurable expiration value
2) There SHALL be a way to enable the AWDC

3) When enabled, the only way to disable an AWDC SHALL be:
   a) An authorization that satisfies the AWDC's Reconfiguration Policy
   b) Receipt of an Initialize Signal
   c) Generation of its Attention Signal
4) The only ways to defer an AWDC SHALL be:
   a) An authorization that satisfies the AWDC's Deferral Policy
   b) An authorization that satisfied the AWDC's Reconfiguration Policy
   c) Receipt of an Initialize Signal
5) The cryptographic authorization policy called a Deferral Policy SHOULD be configurable
6) The cryptographic authorization policy called a Reconfiguration Policy MAY be configurable
7) The Deferral Policy SHALL NOT be configurable without authorization when the AWDC is enabled or the Reconfiguration Policy is set
8) The Reconfiguration Policy SHALL NOT be configurable without authorization when the AWDC is enabled or the Reconfiguration Policy is set
9) If an AWDC's Reconfiguration Policy is set, the AWDC SHALL require authorization against the Reconfiguration Policy for all of the following:
   a) Disabling the AWDC
   b) Setting/Updating the Deferral Policy
   c) Setting/Updating the expiration value
   d) Updating the Reconfiguration Policy itself
10) When enabled, internal or external events or clock ticks SHALL advance the current count towards the expiration value.
11) An enabled AWDC SHALL generate an Attention Signal when expiration occurs
12) A disabled AWDC SHALL NOT generate an Attention Signal.
13) An AWDC SHALL accept an Initialize Signal.
14) The Device Vendor SHALL define the AWDC states resulting from its receipt of an Initialize Signal and after its generation of an Attention Signal.
15) The cryptographic protocols associated with the Deferral Policy and Reconfiguration Policies SHALL be safe from:
   a) Replay attacks
   b) Man-in-the-middle attacks
16) When enabled, the only ways to reconfigure the AWDC SHALL be:
   a) An authorization that satisfies the AWDC's Reconfiguration Policy
   b) Receipt of an Initialize Signal
   c) Deferral (per #4 above)

8.4.7 Wakeup Watchdog Counter

Start of informative comment

A Wakeup Watchdog Counter (WWDC) needs to function independent of any power state an RT may place a Device or Cyber Resilient Module into. Complete loss of external power is more likely to be an indication of change in the environment, not malware local to the device.

The interaction of external power loss and WWDC should be considered carefully by implementers. A WWDC 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 WWDC still functions when the external power source for a device is unavailable, but the Resilience Orchestrator that normally processes the Attention Signal output from the WWDC is offline. Some implementers may consider the loss of external power out of scope for their design. Other implementations may take care to ensure the Attention Signal output from a WWDC is received and acted on by a Resilience Orchestrator before the Initialize Signal is sent to the WWDC.

Some implementations of a WWDC may depend on an external power source. A WWDC implementer should consider whether their solution 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 WWDC for the duration external power was lost or have the WWDC 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 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 WWDC. For example, a WWDC might divide the remaining time in half each time the WWDC external power is lost. This guarantees that the WWDC will eventually expire, irrespective of the duration of time periods between clock ticks or events counted by the WWDC.

End of informative comment

Normative requirements for the Wakeup Watchdog Counter:

1) A Wakeup Watchdog Counter SHALL meet the requirements of one of the following:
   a) A Latchable Watchdog Counter
   b) An Authenticated Watchdog Counter
2) A Wakeup Watchdog Counter SHALL continue to function independent of Device or Cyber Resilient Module power states the RT may control.

9 Primary Element Requirements

9.1 Resilience Engine Requirements

Start of informative comment

Resilience Engine requirements are not within the scope of this specification.

End of informative comment

9.2 Resilience Target Requirements

Start of informative comment

Resilience Target requirements are not within the scope of this specification.

End of informative comment

9.3 Resilience Authority Requirements

Start of informative comment

Resilience Authority requirements are not within the scope of this specification.

End of informative comment

9.4 Resettable Resilience Capabilities

Start of informative comment

This section describes how the input and output Signals of the Cyber Resilient Building Blocks can be used to build devices with enhanced resilience capabilities.

This specification stipulates requirements for Cyber Resilient Building Blocks (CRBB) that can be used to build Cyber Resilient Modules. The resilience capabilities provided by the CRBBs described in this specification have the property that once enabled (or activated) they cannot be disabled (or deactivated) nor, in some cases, reconfigured via their Programmatic Interface until they receive an Initialize Signal. Such resilience capabilities are called Resettable Resilience Capabilities (RRC). The Initialize Signal returns a Resettable Resilience 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 Resilience Capabilities should be coordinated by a Resilience Orchestrator (RO) to build a Cyber Resilient Module.

The Cyber Resilient Building Blocks (CRBBs) are:

- Protection Latches (Read, Write and Read-Write)
- Watchdog Counters (BWDC, LWDC, AWDC, WWDC)
- A Secure Execution Environment (SEE)

The CRBBs are described as standalone elements with input and output Signals. The normative requirements of a Protection Latch specify an input Initialize Signal. The normative 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 implement additional Programmatic Interfaces that can be used, for example, to activate a Protection Latch or to configure or service a Watchdog Counter. The Programmatic Interfaces may be accessible from both inside and outside the SEE (for example, to software executing in the RE or in the RT)

End of informative comment

9.4.1 Cyber Resilient Modules and the Resilience Orchestrator

Start of informative comment

For the Cyber Resilient Building Blocks to deliver useful resilience capabilities, their interconnections must be protected with additional security considerations. For example, if arbitrary software were able to directly generate Initialize Signals, then malware might be able to inactivate 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 Cyber Resilient Module. This specification refers to the entity that manages the signals as the *Resilience Orchestrator (RO)*.

A building block is internal to a Cyber Resilient Module 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 8 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. Those building blocks are not internal to the Cyber Resilient Module (despite being in the same physical device) and this specification does not have requirements about when those building blocks receive Initialize Signals or what happens when they generate Attention Signals.

If a WDC (that is internal to a Cyber Resilient Module), a Resilience Target, a Resilience Engine, or an external entity generates an Attention Signal, the main function of the RO is to generate the Initialize Signals of all the Cyber Resilient Building Blocks that it is managing and perform Module Reset.

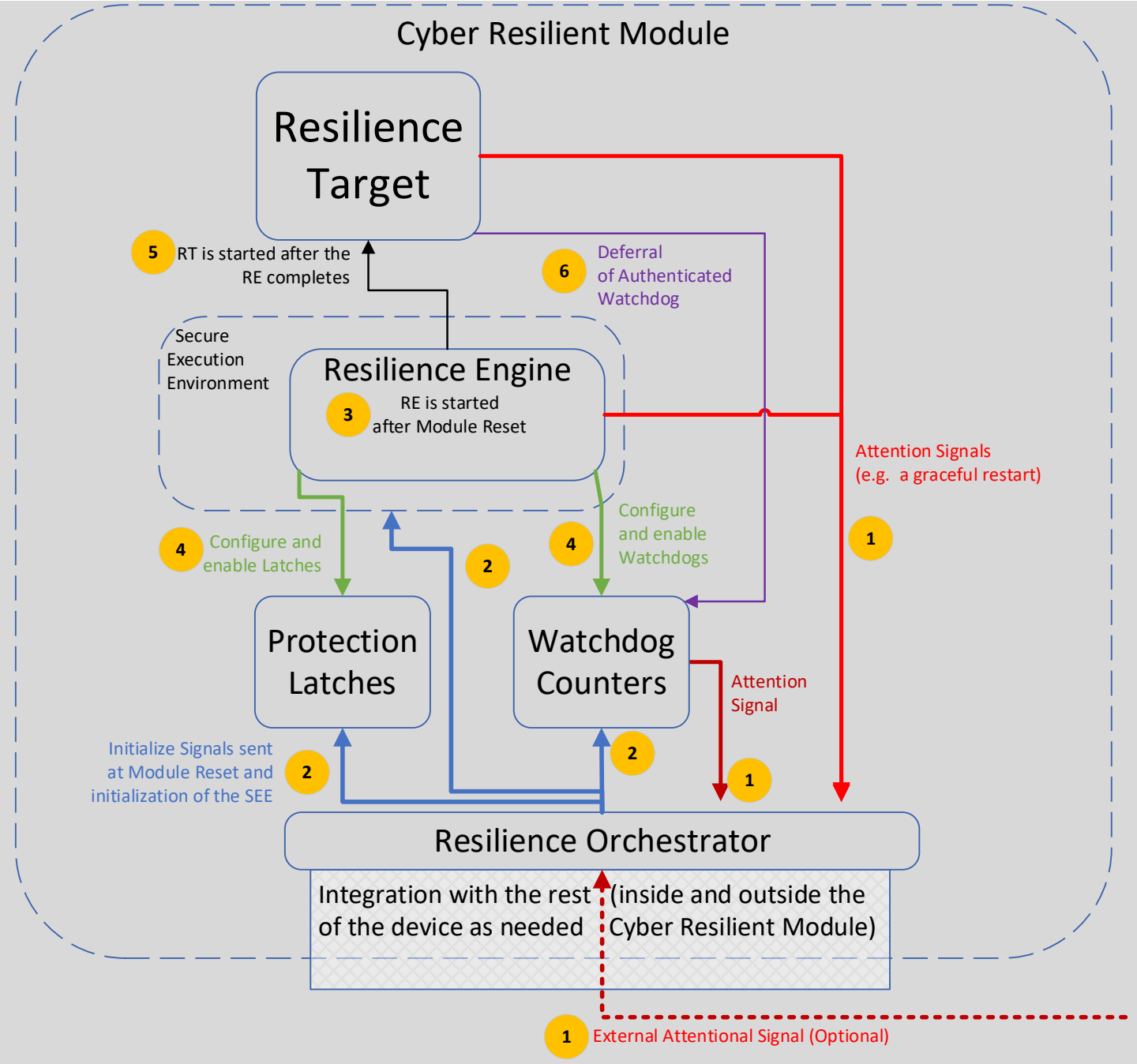

Figure 9: Example RO Illustration

Figure 9 is a schematic illustration of the interconnections between a RO and the Cyber Resilient Building Blocks in a Cyber Resilient Module.

Figure 9 illustrates that the Initialize Signals to the building blocks internal to the Cyber Resilient 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 Cyber Resilient Module, but a device could have any number of Cyber Resilient Modules, 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 exernal to the Cyber Resilient Module.

An Attention Signal directed to a Cyber Resilient Module may be produced by a facility external to the module. The RO may receive Attention Signals from device components external to the Cyber Resilient Module. The RO may also accept attention requests from components internal to the Cyber Resilient Module, for example from software in the RE or RT.

Figure 9 also illustrates (see circle labeled 4 in the diagram) that the Cyber Resilient Building Blocks may present Programmatic Interfaces to software running on the Cyber Resilient Module. Software running in the SEE (for example, 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 Cyber Resilient Module. The numbered list below corresponds to the number items in Figure 9.

1) A Watchdog Counter, RE, RT, or something external to the Cyber Resilient Module 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 (for example, 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 the module will be reset, or park rotational media.
2) The RO performs Module Reset and sends an Initialize Signal to all Cyber Resilient Building Blocks internal to the Cyber Resilient Module. Note: There could be many other Cyber Resilient Building blocks on the same device that are external to this module that are not sent an Initialize Signal. Figure 9 shows a Module Reset established Secure Execution Environment, a Protection Latch and a Watchdog Counter as receiving an Initialize Signal because they are all internal to the module for this example.
3) Sometime after the Secure Execution Environment is reset, it starts running the Resilience Engine. 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 Authenticated Watchdog Counter, the RT may interact with an 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 AWDC. If the RT cannot obtain a Deferral Ticket or if it decides to restart gracefully the sequence jumps back to step 1.

End of informative comment

9.4.2 Requirements for the Resilience Orchestrator (RO)

Start of informative comment

The Resilience Orchestrator is a trusted primary element that coordinates the reset of the Cyber Resilient Module.

This specification does not dictate a power management architecture for Cyber Resilient Modules. 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 contained subcomponents. Depending on its precise behavior, this functionality may meet the requirement for a compliant Resilience Orchestrator or may be modified to serve as a Resilience Orchestrator.

A Resilience Orchestrator for a temporal SEE 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 Cyber Resilient Module and any other implementation specific components (if needed) are reset. Performing Module Reset ensures that control is transferred to the boot-time SEE.

Implementers should ensure that additional capabilities that might interfere with the proper operation of an early boot SEE are also reset.

End of informative comment

Requirements for a Cyber Resilient Module comprised of one or more Cyber Resilient Building Blocks, using a Module Reset Established SEE:

1) The RO SHALL receive all Attention Signals from all Cyber Resilient Building Blocks internal to the Cyber Resilient Module
2) The RO SHALL receive all Attention Signals intended for the Cyber Resilient Module from all sources external to the Cyber Resilient Module
3) Only the RO SHALL send Initialize Signals to the Cyber Resilient Building Blocks (Storage Protection Latches, Watchdog Counters, Secure Execution Environment) internal to the Cyber Resilient Module.
4) Following reception of any Attention Signal the RO SHALL perform a Module Reset (see 8.1.1)
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.

10 Cyber Resilient Module Requirements

10.1 Requirements

Start of informative comment

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 Cyber Resilient Module. This specification focuses on providing requirements for capabilities that support resilient solutions without dictating precisely how solutions must use the capabilities. The requirements for a Cyber Resilient Module 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 normative requirements for solutions that use Module Reset combined with early boot as a SEE for the RE. Requirements for other ways to establish a SEE for the RE are out of scope. This section lists the normative requirements for Cyber Resilient Modules that use Module Reset to establish their SEE.

Section 10 goes on to define two kinds of Cyber Resilient Modules and their corresponding requirements. The first is a Self-Recoverable Cyber Resilient Module that can initiate recovery without relying on external Attention Signals or manual actions. The second is a Symbiotic Cyber Resilient Module that may have dependences on external Attention Signals or manual interaction to initiate recovery.

End of informative comment

Normative requirements for Cyber Resilient Modules that use a Module Reset Established SEE:

1) There SHALL be at least one Protection Latch as defined in section 8.3.9 or Watchdog Counter as defined in section 8.4.
2) There SHALL be a Resilience Orchestrator satisfying the requirements of section 9.4.2.
3) The module SHALL meet the requirements for establishing a temporal Secure Execution Environment for RE execution using Module Reset from section 8.1.1.

10.2 Mitigations Against Denial of Service Attacks Through Low Power States

Start of informative comment

To be resilient, a Cyber Resilient Module 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. Please see section 8.4.7 for additional background.

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

In some cases, Cyber Resilient Module 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 on a daily basis 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 Cyber Resilient Module 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 Cyber Resilient Module is in a low power state.

End of informative comment

Normative requirements to prevent denial of service attacks through low power states for a Self-Recoverable Cyber Resilient Module:

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

Normative requirements to prevent denial of service attacks through low power states for a Symbiotic Cyber Resilient Module:

2) If a Symbiotic Cyber Resilient Module 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 once enabled.

10.3 Roots of Trust

10.3.1 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 Cyber Resilient Module 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 important for secure and reliable remote management of some classes of device. For example, an 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 AWDC Deferral Tickets to non-compliant devices to force them to remediate.

End of informative comment

Normative 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

Start of informative comment

While a Root of Trust supporting attestation is a requirement of this specification, no restrictions are placed on how it is implemented (e.g., TPM, DICE, or other). Note also that storage Read-Write-Protection Latches may be used to satisfy this attestation requirement. One possible implementation that mirrors DICE technologies is described in appendix 12.1.1 of this specification.

This specification does not favor dedicated TPM or DICE protected attestation over an implementation that uses storage Read-Write-Protection Latches. Device Vendors should implement technologies that are appropriate for their customer use-cases and threat models.

Additional notes on Roots of Trust and Cyber Resilient Module attestation:

- The Roots of Trust supporting attestation may reside within a Cyber Resilient Module or elsewhere on the device.
- For those Cyber Resilient Modules 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 (for example, TPM [3] and DICE [4]) containing guidance on Roots of Trust and attestation.

End of informative comment

10.3.2 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 Write-Protection Latches is described in an appendix to this document in section 12.1.2.

End of informative comment

11 Cyber Resilient Module Profiles

Start of informative comment

Each of the Cyber Resilient Building Blocks mitigates different threats, and not all threats are relevant to all scenarios. Some Cyber Resilient Modules implementations may include more than the minimum required CRBBs. Considering this, this specification defines a normative way for a Device Vendor and suppliers to Device Vendors to denote the CRBBs and other relevant trusted computing capabilities a Cyber Resilient Module or Device contains.

Appendix 12.2 describes some of the specific threats that can be mitigated using the CRBBs defined in this specification which illustrates why nomenclature for Cyber Resilient Module profiles is useful.

End of informative comment

CRBB Implemented	Description
WPL	One or more Write-Protection Latches that can be used to protect the RE and its configuration data.
RPL	One or more Read-Protection Latches that can be used to protect a device secret.
RWPL	One or more Read-Write-Protection Latches that can be used to protect a device secret, and an RE along with its configuration data.
BWDC	One or more Basic Watchdog Counters
LWDC	One or more Latchable Watchdog Counters
AWDC	One or more Authenticated Watchdog Counters
LWWDC	One or more Wakeup Watchdog Counters satisfying the requirements of a Latchable Watchdog Counter
AWWDC	One or more Wakeup Watchdog Counters satisfying the requirements of an Authenticated Watchdog Counter

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

1) The nomenclature below SHALL be used to indicate the Cyber Resilient Building Blocks a Cyber Resilient Module contains:

   CRM (<comma separated list of implemented CRBBs>)

   Examples:

   CRM (WPL)

   CRM (WPL, RWPL)

   CRM (RWPL, BWDC)

   CRM (WPL, LWDC)

CRM (WPL, AWDC)

CRM (RWPL, BWDC, LWWDC)

CRM (RWPL, BWDC, AWWDC)

The nomenclature below SHOULD be used to indicate which platform capabilities are available on the platform to support the security of the Cyber Resilient Module:

CRM (<comma separated list of implemented CRBBs>) <plus sign separated list of implemented trusted computing components>

Examples:

CRM (WPL, AWDC) + TPM

CRM (WPL, LWDC) + SB + DICE

12 Appendices (Informative)

12.1 Examples of Implementing Attestation and Secure Boot using Protection Latches

Read-Write-Protection Latches can be used to support attestation and Secure Boot of Cyber Resilient Devices. This section provides a brief description of each, respectively.

12.1.1 Attestation

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 Read-Write-Protection Latches. The UDS is read from a storage location and then its storage location is immediately protected using a Read-Write-Protection Latch. 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 area that contains the engine's image. This prevents later code from modifying the DICE actions in the earliest boot code.

12.1.2 Secure Boot

Storage Write-Protection Latches 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 load 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 a detection capability for the first stage boot loader.

12.2 A Device Resilience Scenario

This specification describes the following Cyber Resilient Building Blocks (CRBBs):

1) A Protection Latch blocking Write-Access to storage (Write-Protection Latch, or WPL)
2) A Protection Latch blocking Read- and Write-Access to storage (Read-Write-Protection Latch, or RWPL)
3) Several classes of Cyber Resilient Watchdog Counter (BWDC, LWDC, AWDC, WWDC)

This specification also describes a Module Reset established Secure Execution Environment and a Resilience Orchestrator (RO) that coordinates security-relevant events such as external and watchdog-initiated resets. These capabilities are required for the CRBBs in the numbered list above to function correctly and are not elaborated on in this section. The RO is assumed to be implemented when CRBBs are part of the device described in this section. Likewise, the RE is assumed to run in the SEE.

This non-normative 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 solution.

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.

12.2.1 Scenario-Based Device Security and Management Requirements

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
- 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 – i.e. an Attention Signal to start the RE
- Storage for device settings, including the Secure Boot policy that identifies authorized firmware
- 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.

12.2.2 Software/Firmware Architecture to Support the Scenario

In this section, a prototypical software/firmware architecture to support the scenario is described. No special security or resiliency features are assumed in this section (CRBB, or other). In the sections 12.2.3 and 12.2.4 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 Resilience Engine (RE) integrated into early boot firmware, for example, the device bootloader
2) A Resilience Target (RT) executing an OS/application package that is started by the RE

The device is managed by a Resilience Authority (RA). In the IoT case, the RA is likely to be a cloud or on-premise device management service. Component devices may be managed by an 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 Resilience Authority to perform management and servicing actions. For example, the RT periodically contacts the RA to determine if 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 of 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.

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

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 harm 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.

12.2.4 Selecting CRBBs to Mitigate Threats

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 4 illustrates how the security threats identified in the software-only implementation are mitigated by the Cyber Resilient Building Blocks 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**	**LWDC**	**AWDC**	**WWDC**
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 4: Mapping of CRBB (and other security technologies) to resilience scenarios

Notes:

1) Assuming DICE is implemented using a Read-Write-Protection Latch, for example, as described in section 12.1.2
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 Authenticated Watchdog Counter or a Latchable Watchdog Counter.
4) If a Latchable Watchdog Counter 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 AWDC or LWDC (or equivalently, switch off the module in such a way that the BWDC, LWDC or AWDC cannot restart the device). If the WDCs 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 AWDC and LWDC can also serve as a WWDC 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 when the Protection Latch is activated, and it will also prevent code from retrieving configuration data stored in a location protected against reads. Both of these 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 Wakeup Watchdog Counter
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.

12.2.4.1 Profile CRM (WPL)

CRBBs assumed:

1) One Write-Protection Latch that can be used to protect the RE

CRBB Support for the Scenario

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 Write-Protection Latch 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.

Write-Protection Latches 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 Write-Protection Latch 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. Solution architects can structure the RE to further mitigate risks: for example, RE network access opens up 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 Write-Protection Latch, 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.

Limitations of this Profile

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 – for example, a TPM.
2) A management controller (RA) cannot force updates or reboots if the Resilience Target is unresponsive.[1] If this functionality is required, then additional support logic must be included: for example, 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.

12.2.4.2 Profile CRM (WPL, RWPL)

CRBBs Assumed:

1) One Write-Protection Latch that can protect the RE stored image
2) One Read-Write-Protection Latch to protect a device secret

[1] Note that a boot-time-isolated RE cannot reliably perform functions once the RT has been started because the code that constitutes the boot-time RE will only run if it is invoked by the RT once the RT has control.

CRBB Support for the Scenario

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 Read-Write-Protection Latch to protect the storage area containing the secrets so that later code – e.g. 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 Cyber Resilient Module Technologies in Appendix 12.1.

This scenario assumes both a Write-Protection Latch (to protect boot code) *and* a Read-Write-Protection Latch (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 Write-Protection Latch.

Limitations

1) A management controller (RA) cannot force updates or reboots if the Resilience Target is unresponsive. If this functionality is required, then additional support logic must be included: for example, 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.

12.2.4.3 Profile CRM (WPL, RWPL, AWDC or LWDC)

CRBBs Assumed:

1) One Write-Protection Latch
2) One Read-Write-Protection Latch
3) An AWDC or LWDC

CRBB Support for the Scenario

In addition to the resilience support described in the profile CRM (WPL, RWPL), devices that satisfy this profile 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 AWDC, the RA forces reset by withholding Deferral Tickets. This means that a device that uses an AWDC 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 AWDC-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 a Latchable Watchdog Counter, 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 (AWDC) or because of LWDC Attention Signals - are not acceptable, then alternative solutions should be used. For example, the device might provide a physical button that allows a physically present operator to manually defer the AWDC.

A WWDC 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.

Limitations

1) If the WDC 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.

12.2.4.4 Profile CRM (WPL, RWPL, WWDC)

CRBBs Assumed:

1) One Write-Protection Latch
2) One Read-Write-Protection Latch
3) A Wakeup Watchdog Counter

CRBB Support for the Scenario

In addition to the resilience support described in the profile CRM (WPL, RWPL, AWDC or LWDC), devices that satisfy this profile provide the RE with a Wakeup Watchdog Counter 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.

12.3 A Subcomponent Resilience Scenario

12.3.1 Subcomponent Resilience Scenario Background

In this profile a subcomponent of a larger platform is used to describe how CRBB's could be used to provide resiliency 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 resiliency 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.

12.3.2 Subcomponent Resilience Scenario Requirements

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 only concerned 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. Meaning, 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 Resilience Engine (RE) integrated into early boot firmware executing on hardware, for example, the device bootloader (which could be contained in immutable ROM).

2) The Resilience Target (RT) firmware executing on hardware that is started by the RE/bootloader. This RT is the firmware which 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 Resilience Authority (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.

12.3.3 Selecting CRBBs to Mitigate Threats

12.3.3.1 Profile CRM (WPL)

No identified changes from Section 12.2.4.1.

12.3.3.2 Profile CRM (WPL, RWPL)

No identified changes from Section 12.2.4.2.

12.3.3.3 Profile CRM (WPL, RWPL, AWDC or LWDC)

In some scenarios it may be applicable/useful to consider using an Authenticated Watchdog Counter (AWDC) to prevent malware from creating a denial of service situation by modification of the RT, disabling the storage device interface, etc. In such a situation, use of an AWDC through lack of authenticated Deferral Tickets reaching the AWDC 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 WDC 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.

12.3.3.4 Profile CRM (WPL, RWPL, WWDC)

A WWDC 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 Denial of Service (DoS) attack. However, similar to section 12.3.3.3, the use of a WWDC (or any other WDC) 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.

12.3.3.5 Host-Symbiont Relationship

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 resilience 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 AWDC 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 Resilience Orchestrator (RO) for the storage controller is able to 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).

12.4 IoT Scenarios

This non-normative 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.

12.4.1 Smart Home

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

12.4.1.1 Exterior Door Lock in Smart Home

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 (Resilience Authority) may not be available so CRBBs that require remote monitoring are impractical (for example, an AWDC). Without remote monitoring, an LWDC is a simple measure that can enable recovery from run-time compromise. A WWDC provides some incremental value by preventing malware from sleeping the device but additional cost is incurred. An LWDC can be used by the RE to remove malware periodically so a WWDC may not be needed in cases the risk is acceptable. Alternatively, the solution could provide a way for a user to manually wake up the device and initiate recovery.

12.4.1.2 Interior Door Lock in Smart Home

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 AWDC, which requires remote monitoring, is impractical. In the absence of remote monitoring, an LWDC is a simple measure that can enable recovery from run-time compromise. A WWDC involves more complexity and cost so it might not be practical in an interior door lock.

12.4.1.3 Smart Smoke Detector in Smart Home

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 WWDC 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 LWDC to ensure reliable functionality. AWDC with a RA can also help detect both functional safety and security anomalies.

12.4.1.4 Light Bulb in Smart Home

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 provide safety or security at all times, 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 LWDC to ensure it is continuously functioning. Failure of the normal light bulb is inconvenient but typically neither safety nor security critical.

12.4.2 Smart Building

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

12.4.2.1 Exterior Door Lock in Smart Building

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 (AWDC). A WWDC may be useful in addition to the AWDC to protect against malware that sleeps the lock and thus prevents the AWDC from working.

12.4.2.2 Interior Door Lock in Smart Building

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 (AWDC).

12.4.3 Industrial Systems

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.

12.4.3.1 Industrial PC

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 Authenticated Watchdog Counter can be used for IPCs that are remotely managed and a Wakeup Watchdog Counter can be used for IPCs deployed in a remote location with unreliable communication. Generally, a Latchable Watchdog Counter 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.

12.4.3.2 Programmable Logic Controller (PLC)

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 (AWDC). While restarting the PLC is not desirable as this may interfere with proper operation, a forced and unexpected restart is generally preferable to compromised operation.

12.4.3.3 Fail Safe Containment Alarm

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 Authenticated Watchdog Counter 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 Wakeup Watchdog Counter could help ensure that the containment alarm system is properly reactivated as soon as possible after unauthorized shutdown. A Latchable Watchdog Counter seems inappropriate because a containment alarm system should never be unnecessarily automatically deactivated.

12.4.4 Summary

Table 5 has the scenarios in section 12.4 as rows. The columns are shaded for building blocks that are relevant for each scenario. The table summarizes the usefulness of Cyber Resilient Building Blocks for IoT scenarios.

IoT Scenario	Write-Protection Latch (WPL)	Read-Write-Protection Latch (RWPL)	Latchable Watchdog Counter (LWDC)	Authenticated Watchdog Counter (AWDC)	Wakeup Watchdog Counter (WWDC)
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 5 Examples of Cyber Resilient Building Blocks in IoT Devices

