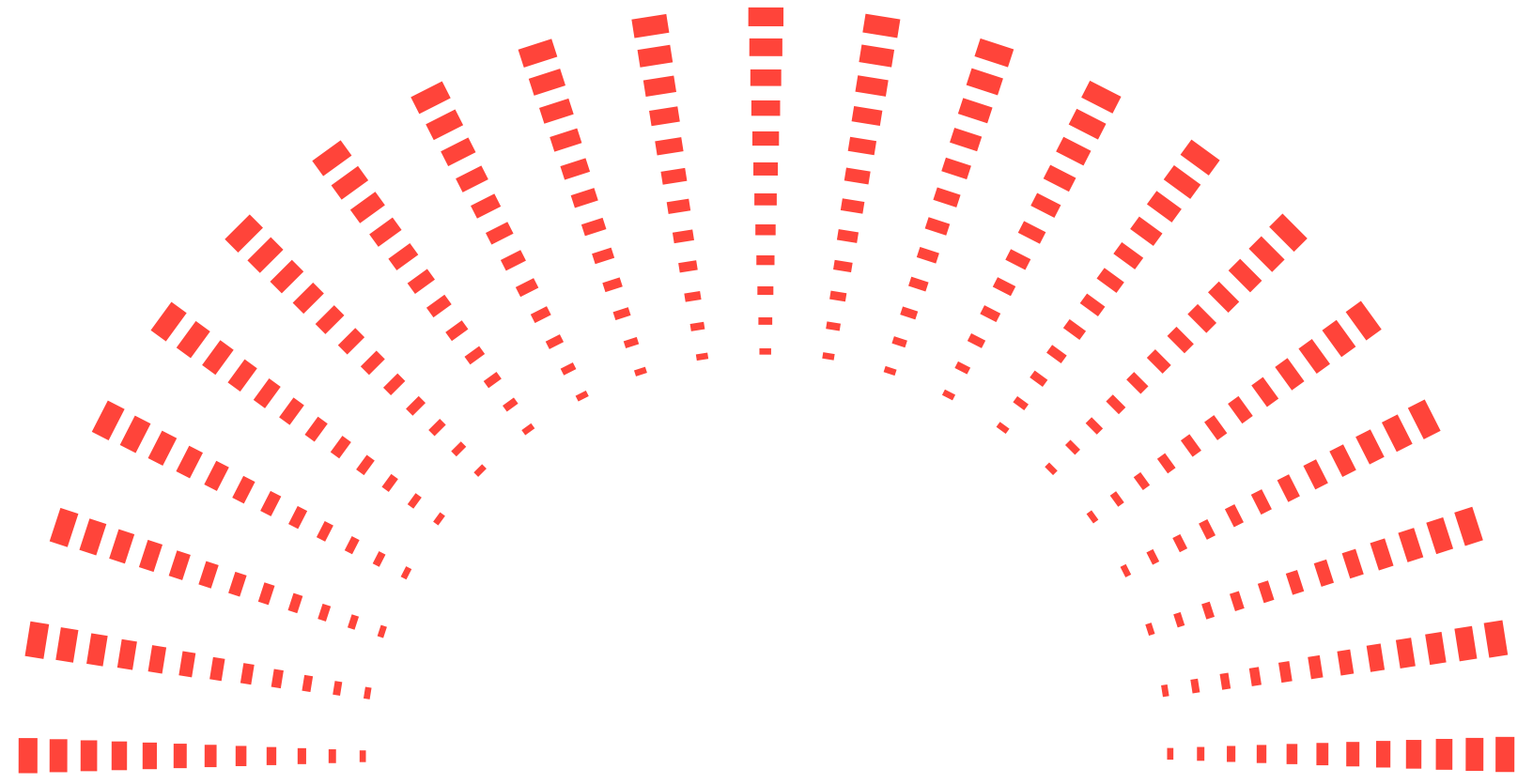
QNX

Solution Guide

Accelerating Software-Defined Embedded Systems Development

QNX General Embedded Development Platform



Overview

In the era of the software-defined world, trends such as urbanization, green energy, digitalization, and autonomous technologies are driving the demand for modern and smart connected embedded systems. The rapid pace of this technological innovation has transformed the landscape of software-defined systems, introducing new opportunities and challenges across various industries. From medical devices and robotics to energy systems and beyond, the development of these systems requires a robust and adaptable software stack. Additionally, some of these next-generation smart devices and industrial automation systems have to be certified to stringent safety and cybersecurity standards. This document explores the essential components and frameworks needed to build and maintain software-defined systems, highlighting the critical role of the QNX General Embedded Development Platform in reducing complexity in designing high-performance software systems.

The convergence of enablers such as autonomous technologies, edge computing, and the Internet of Things has resulted in embedded systems that are increasingly interconnected and intelligent. These systems are expected to operate reliably in real-time environments while meeting evolving functional safety and cybersecurity standards. At the same time, the demand for development teams to meet tight development timelines and stringent compliance requirements without compromising on quality or innovation continues to grow.

To address these challenges, the QNX General Embedded Development Platform redefines how developers can future-proof the design and deployment of modern embedded systems. Engineered to address the demands of scalability, safety, security, and real-time performance, this platform is tailored for industries where reliability and innovation are paramount.

Introduction to QNX General Embedded Development Platform

A typical embedded system consists of application-specific software running on a hardware target board. By modifying the software, other embedded applications can be developed for different markets using the same hardware. In essence, the software modifications define the functionality of both the embedded system and its applications.

QNX General Embedded Development Platform

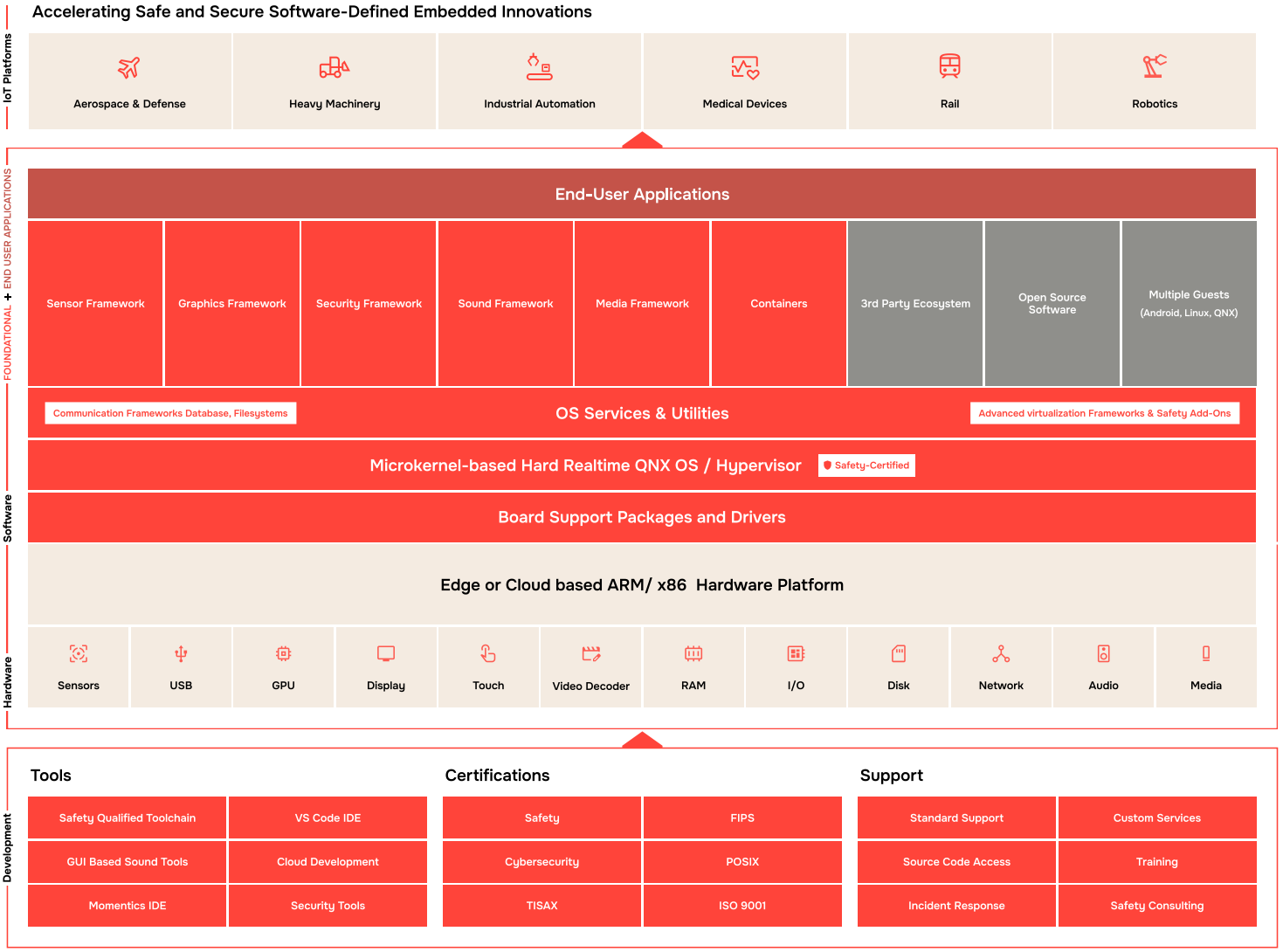


Figure 1: A reliable, comprehensive and adaptable platform to build safe, secure and high-performance software-defined embedded systems

At its core, the QNX General Embedded Development Platform integrates a modular and scalable foundational software stack, anchored by a field-proven real-time operating system with a suite of middleware and development tools. Its modular and scalable architecture supports high-performance communication frameworks, edge-to-cloud integration, and AI optimization, enabling seamless adaptability to diverse application needs. Already deployed in over 500 million mission-critical devices globally, this platform empowers developers to meet

the highest standards of functional safety, cybersecurity, and performance while accelerating time-to-market.

The QNX General Embedded Development Platform foundational software also includes safety- and security-certified components for simplifying the development process to create performance-optimized, secure, and mixed-criticality safety applications, using the latest multicore SoCs. This platform also enables streamlined CI/CD and digital-twin based workflows that reduce overall certification efforts, accelerating the implementation of safe and secure software-defined systems for any industry vertical.

In the QNX General Embedded Development Platform, thanks to the modular architecture of the QNX OS, the QNX BSP helps to decouple the rest of the embedded system from the hardware target board selected for building the software-defined embedded system. Enabled with BSP, the platform can be customized quickly with the required components to help develop the necessary user applications based on reference solution(s). Even when the hardware is not available, development with QNX software can be started using VMWare / VirtualBox / QEMU and other third-party-based virtual targets.

QNX General Embedded Development Platform Components

Platform components can be broadly classified into the following categories:

- Software
- Hardware enablement
- Development tooling

Software

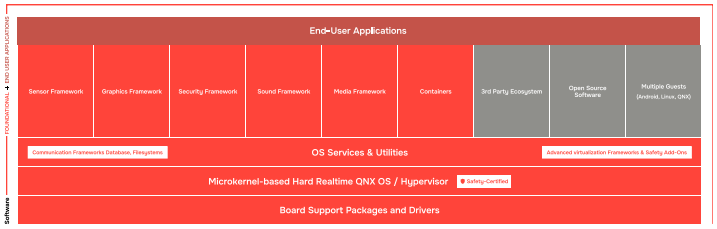


Figure 2: QNX General Embedded Development Platform's Software

The software category consists of field-proven and well-integrated components related to the platform's customizable foundational software stack, along with demo user applications and sample reference solutions for rapid prototyping and simulation-first development workflow. The QNX General Embedded Development Platform adaptable software stack contains:

- QNX Board Support Packages (BSPs) and drivers support
  - Virtual targets based on VMWare / VirtualBox / QEMU (Used when hardware target board is not available)
  - ARM and x86 SoCs based target boards for Edge (Example: SoC vendor's reference boards, production boards)
  - ARM and x86 SoCs based target boards for Cloud (Example: AWS / Azure / Custom cloud enabled digital-twin based target hardware)
- Microkernel based hard real-time QNX OS or its safety variant, QNX OS for Safety (QOS)
- QNX Filesystem for Safety
- QNX Communications for Safety
- OS Services and Utilities
- Communication Frameworks, Databases, Filesystems
- QNX Hypervisor or QNX Hypervisor for Safety

- Out-of-the-Box support for QNX guests based on QNX OS or QNX OS for Safety
- Support for Android, Linux and QNX guests
- QNX Advanced Virtualization Frameworks (QAVF)
  - Enables sharing of resources across host and guest virtual machines, including:
    - Audio sharing
    - Camera sharing
    - Filesystem sharing
    - Graphics sharing
    - Input sharing
    - USB sharing
    - VPU Sharing
    - Virtual Socket
    - Custom guest integration and optimization
- Sensor Framework
- Graphics Framework
- Security Framework
- Sound Framework
- Media Framework
- Containers
- Third-Party Ecosystem/Software
- Open-Source Software

Hardware enablement

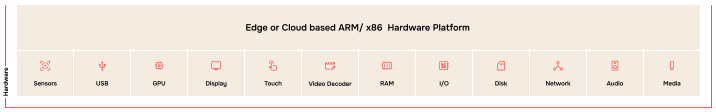


Figure 3: QNX supported target platforms

The hardware enablement category consists of any ARM/ x86 target board that has an available QNX Board Support Package (BSP), for edge or cloud (AWS / Azure) based development. This category also includes QNX enabled VMWare / virtualBox / QEMU and other third-party based virtual targets that can be used for software development when the hardware board is not available. New QNX BSPs can be developed for supporting additional target boards.

Development tooling

Figure 4: QNX Development Tools, Certifications and Support Options

The development tooling category contains QNX Tools, certifications and support services that are needed for customizing the platform to accelerate the development and deployment of embedded systems for various industry verticals.

Tools

- QNX Software Development Platform (SDP)
  - QNX Tool Suite / QNX Momentics Integrated Development Environment (IDE)
  - Safety qualified toolchain
- QNX Toolkit for Microsoft Visual Studio Code
- Cloud Development and Security Tools
- GUI based Sound tools with dynamic visualization

Certifications

- Functional Safety – ISO 26262 ASIL D, IEC 61508 SIL 3, IEC 62304 Class C, EN 50128 SIL 4*, EN 50657 SIL 4*
- Security – ISO / SAE 21434, FIPS 140-3, ISO/IEC 15408, TISAX
- Quality Management – ISO 9001
- Open Standards – POSIX PSE54, FACE 3.1

*available through custom services

Support

- Standard and Long-Term Support
- Custom Services
- Incident Response
- Training
- Source Code Access
- Safety Consulting

QNX General Embedded Development Platform Foundational Software

Board Support Packages and Drivers

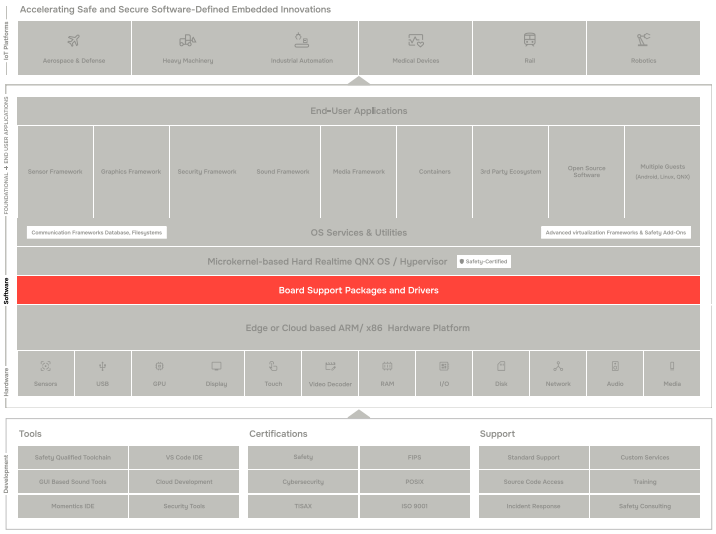


Figure 5: QNX Board Support Packages and Drivers

QNX BSPs simplify the integration of QNX software with your hardware by providing a hardware-specific software abstraction layer. Each BSP is tailored to a specific architecture, board, or even board revision, handling essential tasks like initialization and hardware preparation for system operation.

BSPs define startup behaviors and include:

- Initial Program Loader (IPL) – Hardware-specific.
- Startup Code – Prepares the system environment.
- Utilities – Real-time clock, hardware watchdog configuration etc.
- Device Drivers - Drivers for serial ports, Ethernet, PCI servers, SPI, NOR, SATA, USB, etc.

The source code for most BSP components is included, providing a reference model to write new /custom device drivers or make changes to the IPL or startup code.

The QNX BSP library features extensive support for SoCs and evaluation hardware from leading ARM and x86 manufacturers. Supported GPUs include ARM Mali, Imagination PowerVR, Intel HD, VeriSilicon Vivante, etc.

QNX Standard Support is available for BSPs listed in the QNX Software Center and some BSPs are acquired directly from the BSP supplier or board vendor. New BSPs are regularly added, and custom BSP development is available through QNX Professional Services.

Safety-Certified Microkernel-Based Hard Real-Time QNX OS/Hypervisor

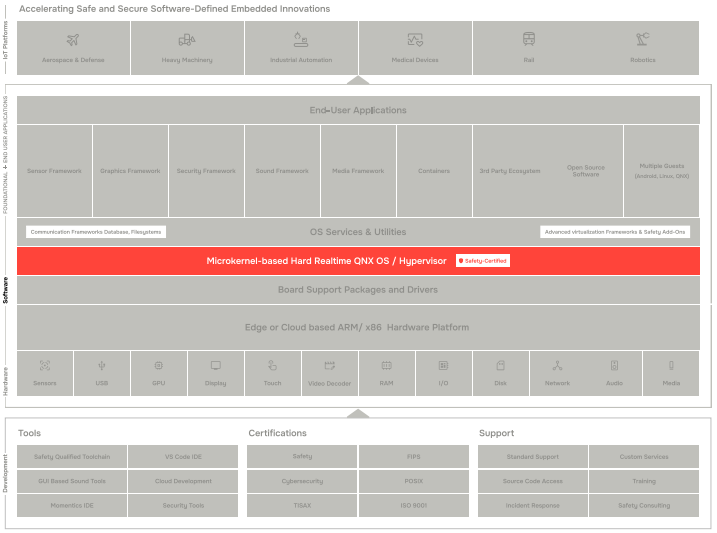


Figure 6: Microkernel based hard real-time QNX OS

Built with flexibility and longevity in mind, the latest QNX Operating System and Hypervisor represent the most advanced offerings yet, designed to deliver cutting-edge performance and reliability. While these versions showcase the newest features and enhancements, access is also provided to older versions to support legacy systems and ease transitions. This multi-version approach allows evolution of software-defined embedded systems at their own pace, with the option for source code access on end-of-life products to ensure continued customization and maintenance. This commitment ensures embedded systems remain reliable, adaptable, and future-proof throughout their lifecycle.

QNX Operating System

The QNX OS is the foundation for developing software for high-performance Systems-on-a-Chip (SoCs) and those embedded systems that run critical, real-time compute-intensive software such as autonomous applications, surgical or industrial robots. The QNX OS, a real-time operating system with our next-generation microkernel, has been augmented to support the latest 64-bit ARM and x86 hardware platforms. The QNX OS features advanced microkernel built on the pillars of performance, scalability, security, safety, and real-time execution.

Performance and real-time execution are central to the design of the QNX OS. Its real-time determinism means it can handle many time-critical tasks where predictability, scalability, and reliability are necessary for mission- and safety-critical applications. And its modular design makes it more flexible than a traditional monolithic OS. Its scaling capabilities meet the needs of high-performance, compute-intensive system architectures powered by next generation multicore silicon.

With its advanced microkernel architecture, QNX OS is safe and secure by design, providing features that isolate and protect critical processes. QNX OS comes with a suite of security features including the latest access control mechanisms (e.g., security policies and permission controls), full encryption support, secure filesystems, and more.

QNX OS Core Capabilities

Microkernel Architecture

Figure 7: QNX Microkernel Architecture

This design isolates every application, driver, protocol stack, and filesystem in its own address space, outside the kernel. This means that a failed component won't take down other components or the kernel; it can be restarted immediately with minimal impact on the rest of the system. With a fault-tolerant and secure microkernel architecture, QNX OS is the ideal OS for all projects because it enables scaling of the underlying hardware (2-core SOC to 64-core SOC). Its future-ready reliable design works across all safety and non-safety related deeply embedded to high-performance compute systems, without needing to change the OS and with limited impact on system-level performance.

High-Performance
Provides high overall OS throughput performance driven by the next- generation microkernel and networking. This enables development teams to make use of SoCs ranging from 2 to 64 cores, maximizing full potential.

Seamless Scalability
Provides near-linear scalability as the number of cores increases. This allows development teams to scale and increase the workloads on SOCs with CPU cores ranging from 2 to 64 cores as seamlessly as possible.

Hard Real-Time
Improves hard real-time capabilities. It is fully pre-emptive with strict time constraints and guaranteed response times, meaning it can fully monitor the relevant priority of competing tasks and quickly schedules the required task.

Low Latency and Jitter
Offers low latency and jitter, essential for safety, mission-critical, and real-time systems that need deterministic, high-precision, and accurate responses.

System Analysis Toolkit (SAT)
Provides sophisticated tracing and profiling mechanisms based on its advanced instrumented microkernel, allowing execution monitoring in real time or offline. Since it works at the operating system level, the System Analysis Toolkit (SAT), unlike debuggers, can monitor applications without having to modify them in any way.

The SAT allows real-time debugging to help pinpoint deadlock and race conditions by showing what circumstances led to the problem. It also offers a nonintrusive method of instrumenting the code to dynamically monitor real-time system errors and help improve overall system performance.

The QNX OS SAT consists of the following components:

- instrumented microkernel
- a small, highly efficient event-gathering module
- kernel buffer
- data-capture program (tracelogger)
- data interpreter (traceprinter)

System events can also be traced and analyzed under the control of a GUI-based Integrated Development Environment (IDE).

Unlike the SAT, debuggers lack the execution history essential to solving the many complex problems involved in application tuning.

A debugger can view a single process, while the SAT can view all processes at the same time. Because it offers a system-level view of the internal workings of the kernel, the SAT can be used for performance analysis and optimization of large systems as well as a single process.
The SAT offers valuable information at all stages of a product's lifecycle, from prototyping to optimization to in-service monitoring and field diagnostics.

QNX OS for Safety and Safety Add-Ons

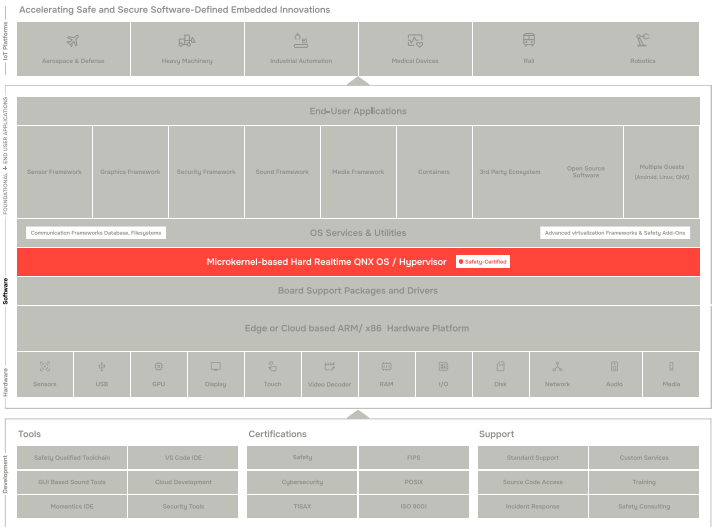


Figure 8: QNX OS for Safety

QNX OS for Safety is a safety-certified operating system that can meet requirements up to the highest levels of functional safety. It is based on the QNX SDP and contains safety-certified variants of key components of the QNX OS, including the microkernel, process manager and POSIX-compliant libraries, with support for multi-core processing.

QNX OS for Safety is certified as a Safety Element out of Context (SEooC) to ISO 26262 ASIL D (automotive), IEC 61508 SIL 3 (industrial), and IEC 62304 Class C (medical), EN 50128 SIL4 and EN 50657 SIL4 (railway - available through custom services). It includes safety variants of the system memory management unit manager (SMMU-MAN for Safety), math libraries (libm), and security components. It also includes a C++ library that is certified to ISO 26262 ASIL B for the C++ templates and headers and ISO 26262 ASIL D for the C++ runtime binaries. The safety certification also includes qualification for the C and C++ toolchain to TCL3 and T3. Using these pre-certified products contributes to reducing the effort associated with system-level safety certifications.

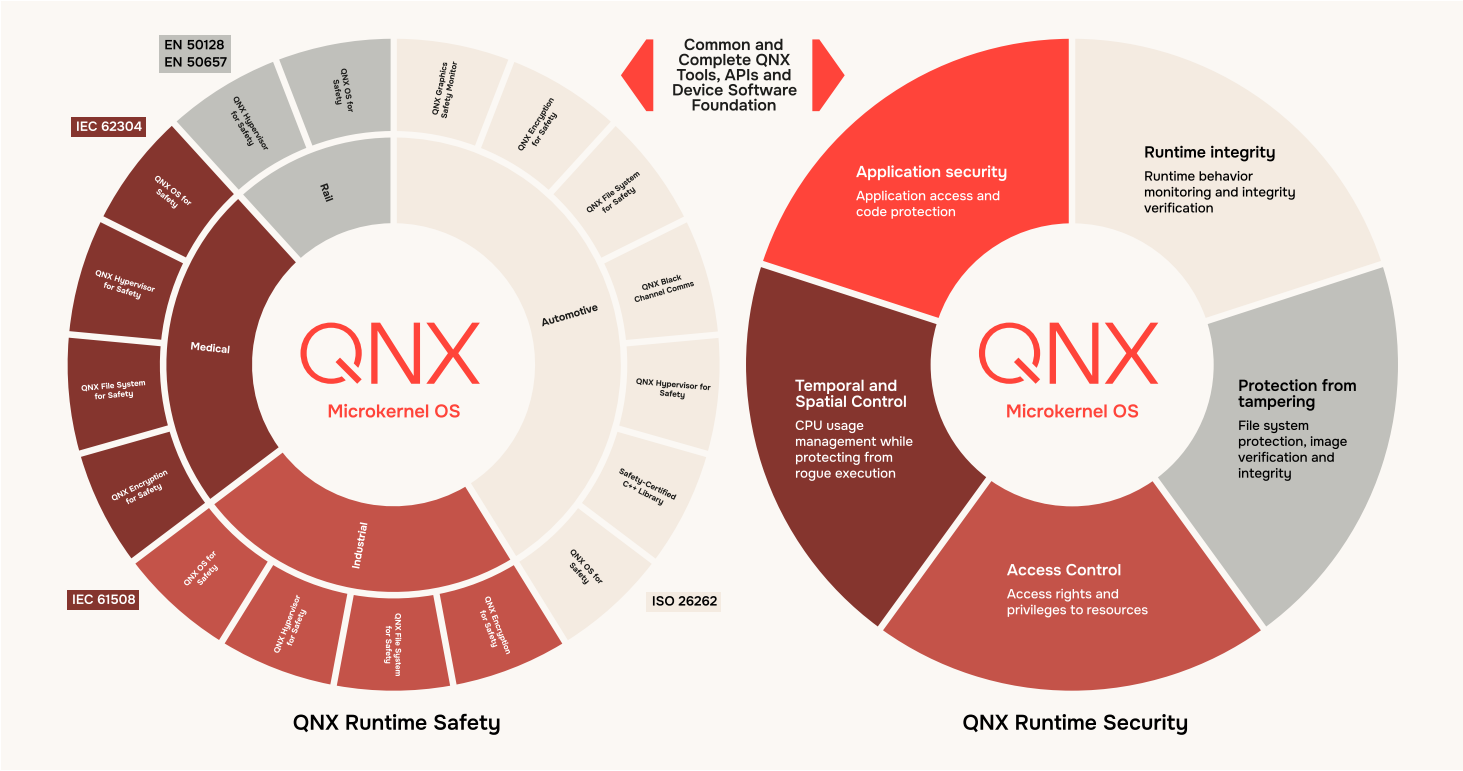


Figure 9: QNX Safety and Security

Only a secure system can be a safe system. The QNX OS for Safety provides a comprehensive, layered approach to security.

This layered approach enables the implementation of only the required security protocols that are needed to mitigate threats and harden the systems, including: granular control of system privilege levels, encrypted and self-verifying filesystems implementing AES 256 encryption and lockable encryption domains, secure logging of system activities, heap, stack and memory protection, and secure boot implementing TPM and TrustZone.

QNX Filesystem for Safety

The QNX Filesystem for Safety is a safety add-on. It is a read-only filesystem that is designed to protect the integrity of filesystem data in an embedded system. It validates and performs the essential safety checks on the filesystem data. This solution provides POSIX ACL support, can run concurrently with other QNX filesystems all with minimal impact on system performance. The QNX Filesystem for Safety is certified to ISO 26262 ASIL B on QNX OS for Safety.

QNX Communications for Safety

QNX Communications for Safety is a safety-add-on. It encapsulates the data being exchanged and performs essential safety checks to validate it at both ends.

This solution protects data communication from systematic software faults, random hardware faults and transient faults, and helps in the automatic prevention of damages from these failures, all with minimal impact on system performance. QNX Communications for Safety is certified to ISO 26262 ASIL D on QNX OS for Safety.

OS Services and Utilities

Real-time and other mission-critical applications generally require a dependable form of Inter Process Communication (IPC), because the processes that make up such applications are so strongly interrelated.

QNX OS is the first commercial operating system of its kind to make use of message passing as the fundamental means of IPC. The OS owes much of its power, simplicity, and elegance to the complete integration of the message-passing method throughout the entire system. The QNX microkernel uses kernel calls to support the following:

- threads
- message passing
- clocks
- timers
- interrupt handling
- semaphores
- mutual exclusion locks (mutexes)
- condition variables (condvars)
- barriers

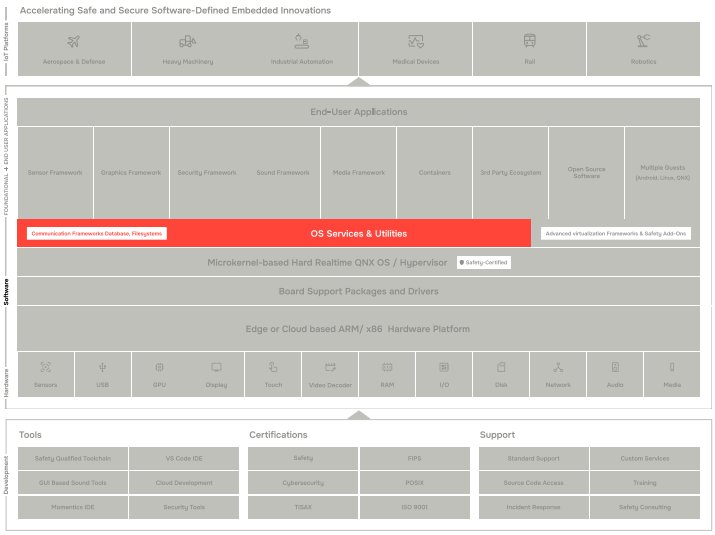


Figure 10: QNX OS Services and Utilities

The entire OS is built upon these calls. The OS is fully preemptible, even while passing messages between processes, it resumes the message pass where it left off before preemption. It acts as a kind of software bus that lets you dynamically plug in/out OS modules whenever they're needed.

Communication Frameworks

QNX io-sock is a high-performance networking stack that enables high-speed bus networks to transmit time-sensitive sensor, audio and video content across different high-performance compute nodes. High-level capabilities include:

- Improved throughput and latency performance especially on 1G and 10G channels
- IPv4/IPv6 multi-threaded network stack
- Unicast and multicast routing
- Packet Filtering
- VLAN and Network Bridging
- Wireless, Wi-Fi 6, USB and PCI Ethernet driver support
- IPSec and IKE (Internet Key Exchange) support
- Stream Control Transmission Protocol (SCTP)
- Process and network interface isolation
- Common Address Redundancy Protocol (CARP)

OS Feature Highlights:

Kernel

- High-resolution software timers
- POSIX scheduling policies
- Fully preemptive priority-based scheduling
- Symmetric multiprocessing
- Up to 64 cores
- Up to 133 million threads on a system
- Up to 16 TB of RAM
- Safe and secure interrupt handling
- Scheduling policies
- POSIX Interprocess Communication (IPC)
- QNX IPC with limitless message sizes
- SMMU management
- Fast boot
- Many POSIX features

Security

- POSIX permissions
- POSIX Access Control Lists (ACL)
- Random service generator
- Fortified system functions

- Secure Process launcher
- Security policies
- Secure boot
- QNX Trusted Disk (QTD)
- Pathtrust
- Address Space Layout Randomization (ASLR)
- Process manager abilities
- Generic crypto device driver
- QNX Binary Security Check tool
- OpenSSL 3
- FIPS 140-2/3 Crypto Libraries

Filesystems

- SMBv3
- QNX Power Safe
- NFS
- Encrypted
- Squash
- Compressed

QNX Hypervisor (Support for Android, Linux, and QNX Guests)

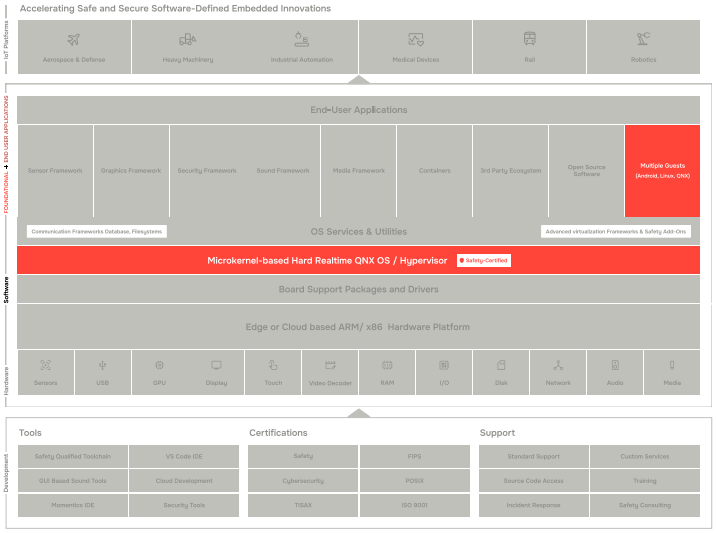


Figure 11: QNX Hypervisor

QNX Hypervisor is a microkernel-based hypervisor built for creating and managing virtual machines and their guests on embedded devices.

With the QNX Hypervisor, developers can consolidate multiple similar or heterogeneous operating systems onto a single system on a chip (SoC) to reduce the cost, size, weight, and power consumption of their systems while separating and isolating general-purpose and safety critical operating systems and applications. QNX Hypervisor inherits the reliability, performance, and security characteristics of the QNX OS, which already ships in hundreds of millions of critical embedded systems worldwide. The QNX Hypervisor follows industry standards such as VirtIO and Android Automotive HAL, minimizing the modifications needed to implement Android™ and/or Linux® guests in its virtual machines and is optimized for a number of silicon vendor SoCs. Standards-based device sharing, and flexible virtual machine configuration ensure that the QNX Hypervisor environment can be scaled up for high-performance computing and complex system/domain controllers, as well as scaled down for more resource-constrained and less complex embedded systems and controllers.

A virtual device developer's toolkit, containing an API reference and a user's guide with code samples, is available to extend the hypervisor's virtual machines with custom virtual devices, including para-virtualized devices designed and built to the VirtIO standards.

QNX Hypervisor for Safety

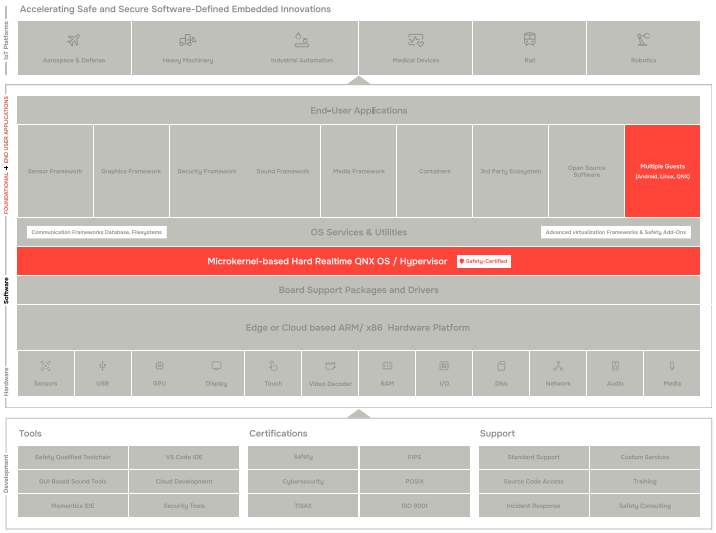


Figure 12: QNX Hypervisor for Safety

QNX Hypervisor for Safety supports virtualized systems for functional safety.

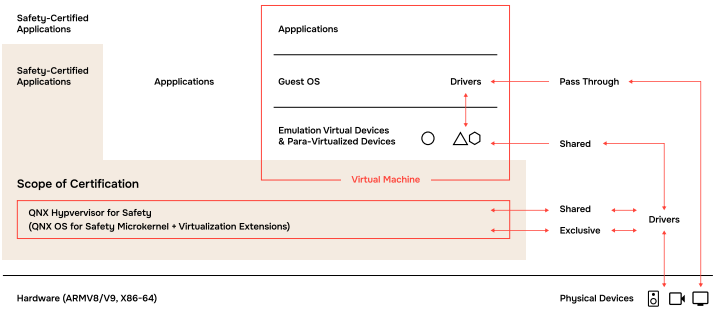


Figure 13: QNX Hypervisor for Safety

QNX Hypervisor for Safety is certified to ISO 26262 ASIL D, IEC 61508 at SIL 3, IEC 62304 Class C, EN 51028 SIL 4, and EN 50657 SIL 4. This product is foundational for developing safety-critical virtual machine systems requiring up to this level safety rating. By using the real-time QNX Hypervisor for Safety, embedded developers can consolidate multiple operating systems onto a single compute platform or system-on-a-chip (SoC) to reduce the cost, size, weight, and power consumption of their system designs while enforcing clean separation and isolation of safety-critical and general-purpose applications and operating systems. Note that QNX OS for Safety can also be used as a powerful real-time, pre-certified safety guest managed by the QNX Hypervisor for Safety.

The virtual device developer's toolkit, with code samples, can be used to extend the hypervisor's virtual machines with VirtIO based custom virtual devices designed to run in safety hypervisor environments.

QNX Advanced Virtualization Frameworks (QAVF)

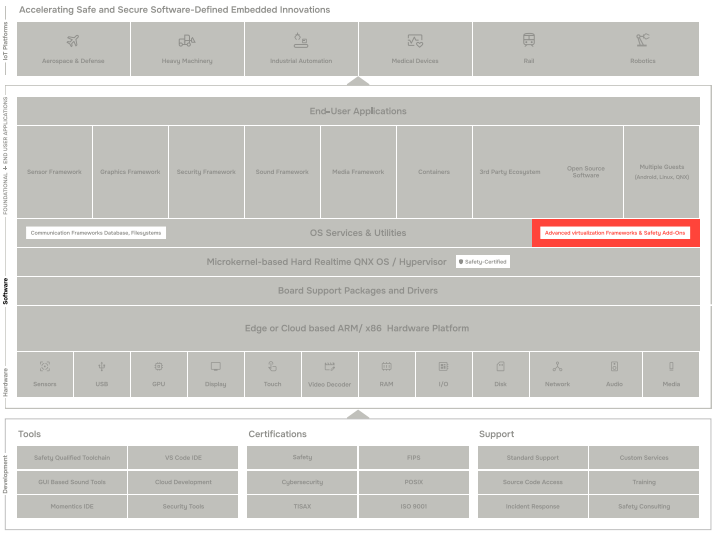


Figure 14: QNX Advanced Virtualization Frameworks

The QNX Advanced Virtualization Frameworks are designed to work with the QNX Hypervisor product family and are offered as middleware extensions to QNX Hypervisor and QNX Hypervisor for Safety products. These frameworks enable sharing of devices among the virtual machines and the hypervisor host environment. Advanced device sharing components include:

- display sharing between Android, Linux and QNX systems
- graphics surface sharing
- video stream sharing, camera sharing
- audio sharing for playback and capture
- audio management
- socket communications, filesystems, and touchscreen support

For example, QAVF enable a guest OS such as Android/Linux/QNX, to share device access with other operating systems in a virtualized environment.

Technology	Standard	Guest Support
Shared GPU & Display	Virtio 1.2.5.7 GPU Device	Android & Linux
Shared graphic surface	Android guest display in QNX host	Android & Linux
Virtual Socket	Virtio 1.2.5.10 socket Device	Android & Linux
Shared Input	Virtio 1.2.5.8 GPU Device	Android & Linux
Shared Audio	Virtio 1.2.5.15 Sound Device	Android & Linux
Shared video and camera	Latest technical committee draft	Android HAL specific (Linux requires integration)
Shared USB	USB port shared by guest or hypervisor host	Android & Linux
Sensor Sharing	Android HAL	Android
Shared Filesystem	Virtio 1.2.5.11 File System Device	Android & Linux
QNX guest graphics sharing	QNX guest graphics	QNX
Virtual Bluetooth	Bluetooth Device (media, HandsFree)	Android (Linux requires integration)

Table 1: QAVF Guest Support

By following standards such as VirtIO, an Android/Linux/QNX guest can run unmodified and share underlying hardware and software services such as graphic displays, acoustic environments, touchscreens, media storage devices, video streams and cameras.

Of importance, QAVF runs outside of the guest. This permits not only the sharing of devices while the guest virtual machine is active, but also user interaction with cameras, displays, audio and input devices while the guest is still booting.

Notable features of QNX Advanced Virtualization Frameworks include:

- Extensive optimized support for the VirtIO standard as well as custom silicon vendor device sharing interface.
- Supported backend services designed to run in the QNX Hypervisor host domain, thus providing high performance while removing the need for a special Service OS or Domain0 guest.
- Frameworks that use sharing, communication, and isolation techniques that are the best fit for each virtualization requirement. This can involve a combination of VirtIO standards, agent/client software, and custom-built virtual devices.
- Frameworks that are modular and allow for partitioning with some software running in the Hypervisor host and some in guest VMs. This is especially important for graphics sharing. For example, some digital cockpit designs require a separate QNX guest for safety-certified instrument cluster that share the same GPU with Android that controls the infotainment environment.
- Supported backend services in the host to enable achieving key performance indicators (KPIs) such as fast boot, time to play audio, and video streaming, more easily than if those services were available only after the guest starts running.
- A path to functional safety is considered in the design for each and every framework.

Sensor Framework

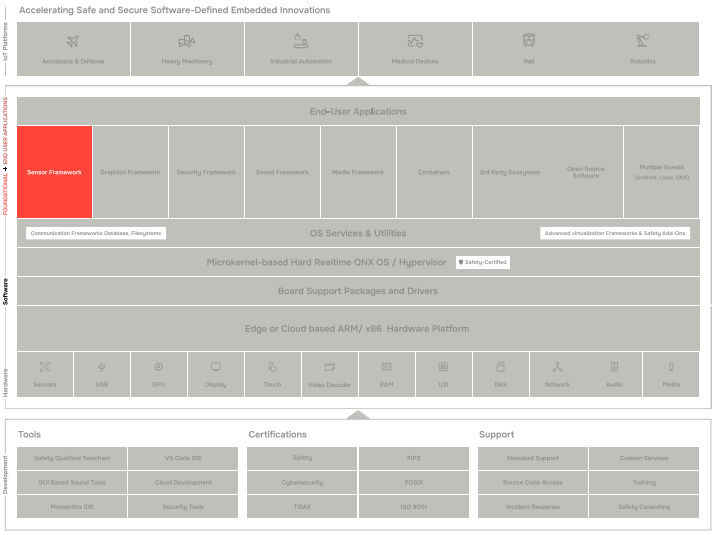


Figure 15: QNX Sensor Framework

The QNX Sensor Framework provides a set of technologies for multi-sensor data acquisition (radar, lidar, camera and others), distributed processing and vehicle networking. The QNX Sensor Framework is intended for use in

a variety of advanced driver assistance systems (ADAS) and automated driving applications, as well as other embedded applications with sensor technology, including:

- Informational ADAS systems such as surround view and forward-facing, machine vision processing systems with perception capabilities for industrial automation
- Active safety systems such as autonomous emergency braking systems
- Consolidated CPU / domain controllers that provide active safety functions
- Autonomous driving and robotic systems, including smart sensor control systems, and centralized high-performance compute nodes

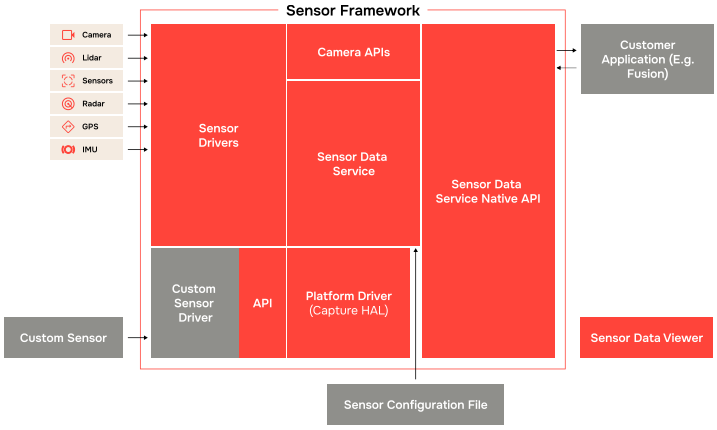


Figure 16: QNX Sensor Framework Architecture Overview

Graphics Framework

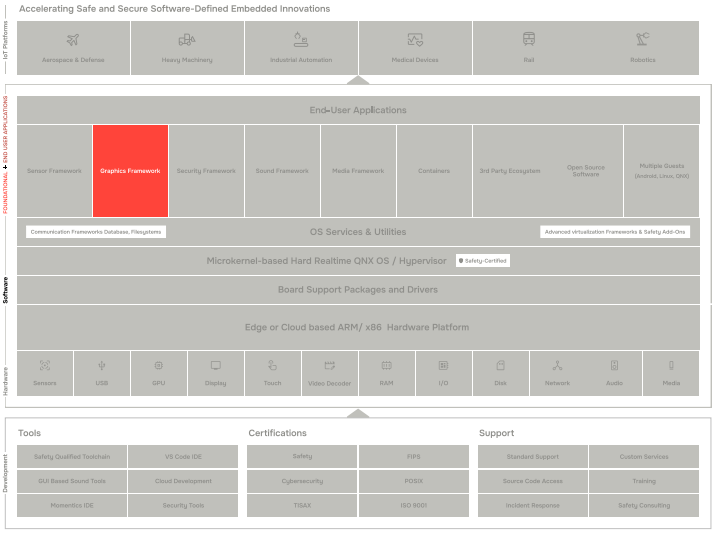


Figure 17: QNX Graphics Framework

QNX Screen is a middleware extension and a graphics framework that provides all the functionality necessary to develop interactive user experiences. It is a compositing windowing system that can composite graphics from several different rendering technologies.

QNX Screen allows developers to easily create simple to advanced GUI interfaces for embedded applications using industry-standard UI development tools and technologies. It also enables developers to create separate windows for the output of each rendering technology (e.g., HTML5, Qt, Video, OpenGL ES, Vulkan, GTK, Unity 3D etc.) so that each window can be transformed (e.g., scaling, translation, rotation, alpha blending, etc.) to build the final scene for display.

Security Framework

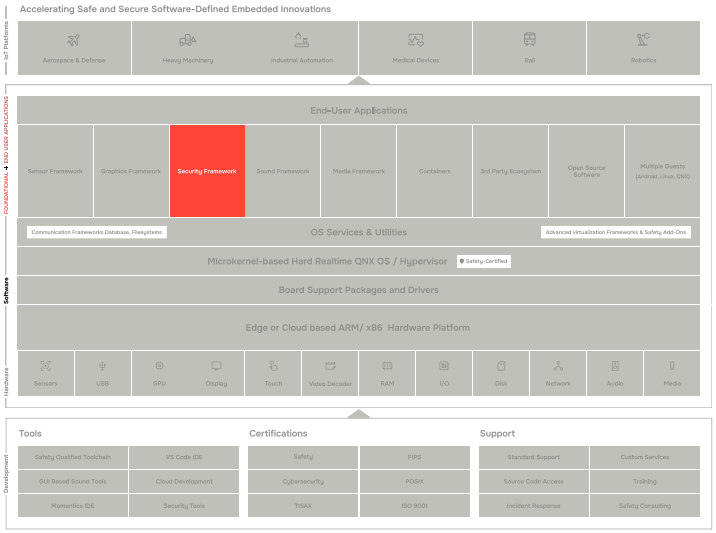


Figure 18: QNX Security Framework

With more than 500 patents covering Elliptic Curve Cryptography (ECC), the QNX Security Framework provides device security, anti-counterfeiting, and product authentication to deliver end-to-end security with managed public key infrastructure, quantum resistant code signing and other applied FIPS validated cryptography and key management solutions.

Sound Framework

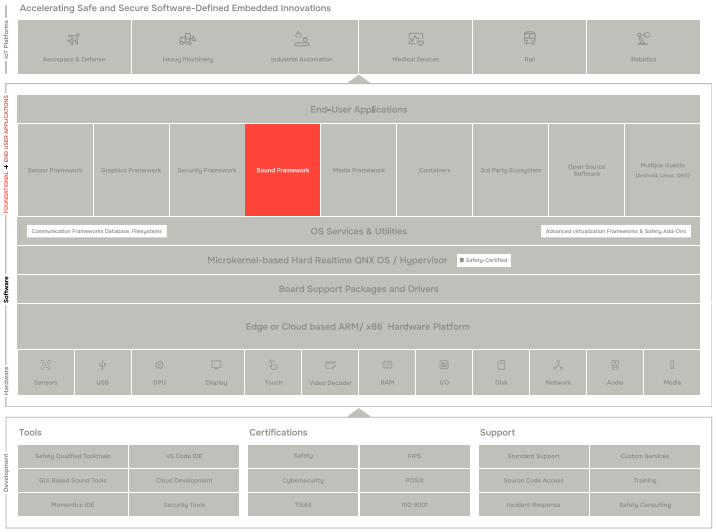


Figure 19: QNX Sound Framework

QNX Sound is a comprehensive sound solution that provides creative freedom for audio and acoustics professionals who are building and deploying software-defined audio solutions. Advanced graphical design, tuning and development tools are provided supporting leading automotive SoC hardware that include both application processor and digital signal processing (DSP).

Low-latency runtime modules consist of voice processing, sound synthesis, noise reduction, in-car communications, chimes and safety alerts, audio signal processing, and signal flows.

The runtime environment also adds audio policy management and integration with virtual environments (hypervisor) including Android, Linux and QNX guests. In addition, QNX Sound supports extensions for third party media processing and value-add technology organizations allowing OEMs to meet continually evolving automotive business challenges. Cloud and digital-twin based multi-channel system-wide audio application development is supported, including rapid testing and deployment to the embedded edge devices.

QNX Sound also includes a graphical host-based tool (QNX® LiveAMP) that enables real-time adjustment of system parameters, signal streaming, injection at multiple tap-points in the system and includes real-time spectrum and waveform displays. Additionally, QNX LiveAMP offers offline sound design capabilities and accelerates real-time tuning, diagnosis, configuration, deployment, system-wide audio management and analysis of acoustic/audio applications.

QNX Sound provides a path to consolidating acoustics as part of centralized / advanced computing architectures and accelerates the implementation and deployment of software-defined audio solutions for smart embedded systems.

Media Framework

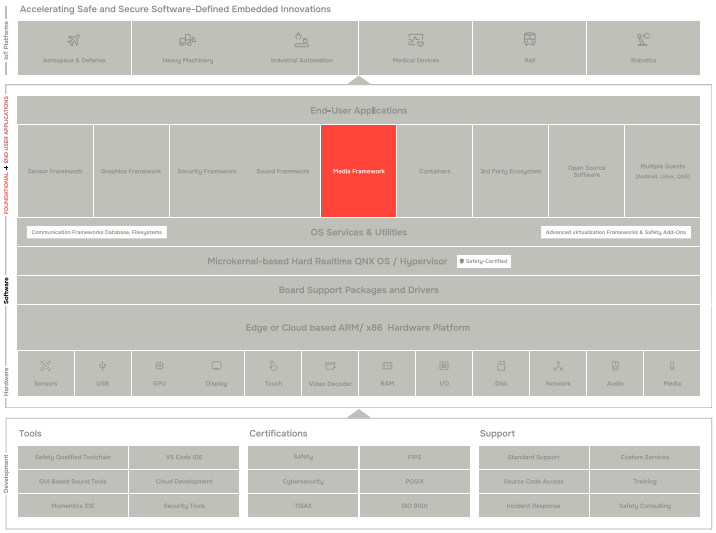


Figure 20: QNX Media Framework

QNX Multimedia is a media framework middleware extension that enables media-rich, high-quality playback, encoding and streaming of audio and video files for QNX embedded systems.

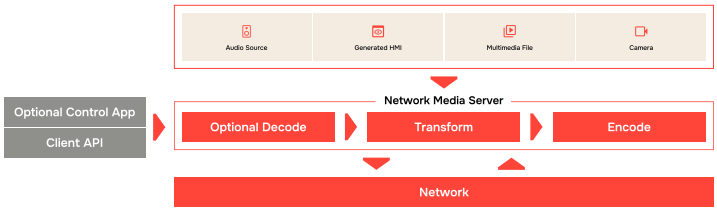


Figure 21: Multimedia Encode and Casting

The high-level capabilities of this framework include:

- Media playback from file or streamed content
- Support for hardware-accelerated audio and video CODECs
- OpenMAX AL
- Media recording and streaming of content to a receiving endpoint
- QNX Multimedia Playback
- QNX Multimedia Encode and Casting

Containers

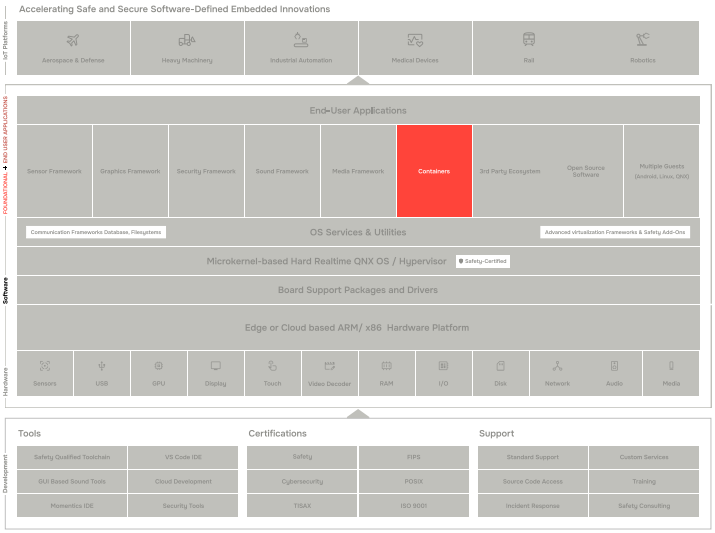


Figure 22: QNX Containers

Powered by QNX microkernel technology, QNX Containers are lightweight environments that enable efficient deployment of embedded system applications and their dependencies to QNX target device(s), ensuring isolation, security, portability, and scalability.

Whether it is the development of medical robots that require enhanced security and reliability of mixed-criticality safety applications using latest multi-core SoCs, or creating an application for next-generation industrial control systems that can be cost-effectively ported across product lines, QNX Containers provide a standards-based solution for management and control of these runtime container environments. It implements the most popular standards-based container solutions:

- OCI compliant (Open Container Initiative).
- Kubernetes-based toolchains for creation, deployment and management.
- Docker (industry standard) repositories for remote storage and retrieval. Local storage is also supported.

QNX Container runtime environment follows the OS restrictions and security features on networking, filesystems, devices, memory, communications, access control and CPU. These restrictions set provides highly secure and isolated embedded containers while still maintaining the high performance and hard real time nature of the QNX operating system.

QNX Containers isolate ML (Machine learning) capable applications from the rest of the system and allows for a cloud-first and/or cloud-centric development approach and digital-twin workflows. This accelerated development approach enables portability between targets, integration into automated pipelines for development and test, collaboration across geographical regions, and scalability using cloud resources. The software can be made more modular and extendable for easily adapting the embedded devices to evolving needs and technologies without compromising performance or security.
The ability to quickly launch QNX Containers based digital-twins at scale in the cloud for development, testing, staging, and deployment enhances overall productivity and code quality.

Figure 23: Accelerated cloud-first development workflows

Additionally, existing OCI and Kubernetes based orchestration infrastructures and tools can be used to streamline the process of managing updates to ensure software-defined embedded systems remain up-to-date and secure.

Third-Party Ecosystem

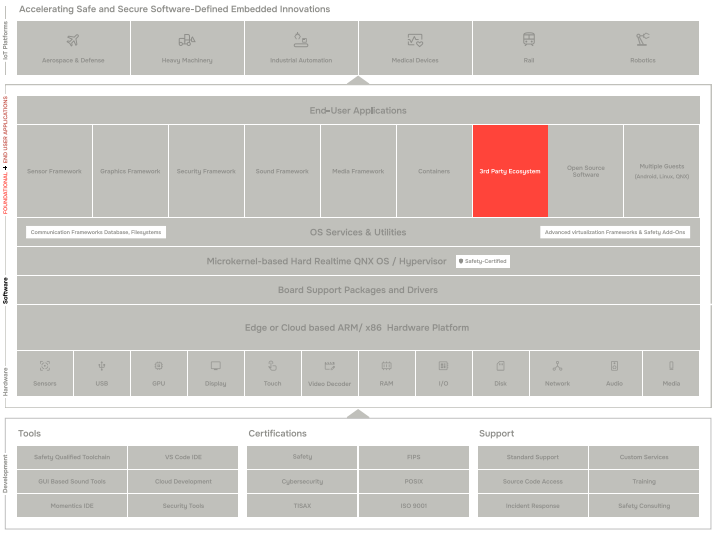


Figure 24: Integrated Third-Party Software

For decades, QNX has worked with a network of collaborative organizations to provide complementary technologies and broaden QNX support across industries and regions. These important relationships have enhanced our ability to provide the foundational software, middleware, and services behind the world's most critical embedded systems.

Several QNX based third-party complementary solutions are available that can be used in QNX General Embedded Development Platform for accelerating your development and deployment of smart embedded systems. For example, support for middleware partner solutions include:

- Communication Protocols: Data Distribution Service (DDS), Time Sensitive Networking (TSN), Profinet, Profisafe EtherCAT, Fail Safe over Ethercat (FSoE), CANOpen Safety, Zenoh, Modbus
- Human Machine Interface (HMI)
- Embedded Database
- Server Message Block (SMB 3), etc.

Open-Source Software Support

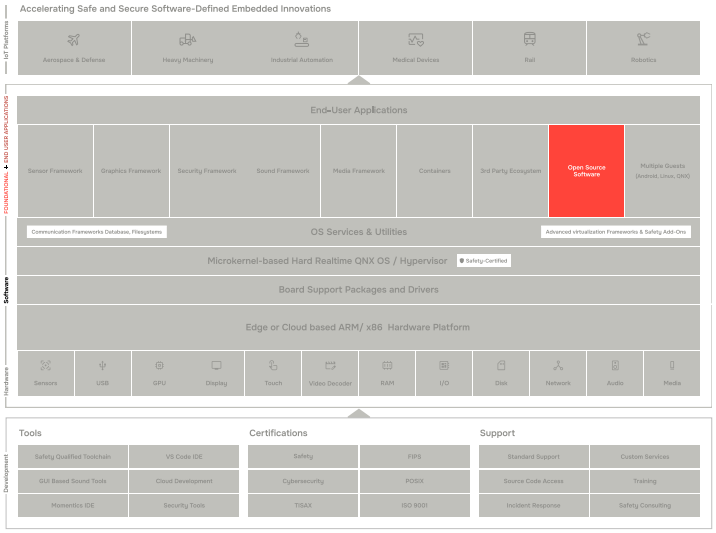


Figure 25: Open-Source Software Support

QNX supports open standards and there are several open-source software (OSS) packages available for building innovative software-defined systems. Developers can access these open-source ports through compiled binaries, delivered as open-source add-on packages from QNX Software Center, or source code upstreamed to original project repository or hosted in a public QNX repository. Examples of open-source integrated software with QNX General Embedded Development Platform, include:

Cairo	dlt-daemon	EmulationStation
Fast-DDS	gtsam	Protobuf
grpc	gtk	Gflags
jsoncpp	weston	C-ares
libmodbus	aws-crt-app	Re2
opencv	METIS	Googletest
pixman	OpenBLAS	Mosquito
sqlite3	SuiteSparse	Azure-iot-sdk-c
csmith	Glog	Pytorch
tinyxml2	Tensorflow	ROS 2 Humble
rust	farmhash	Etc.

QNX General Embedded Development Platform Development Tooling

Tools Certifications and Support

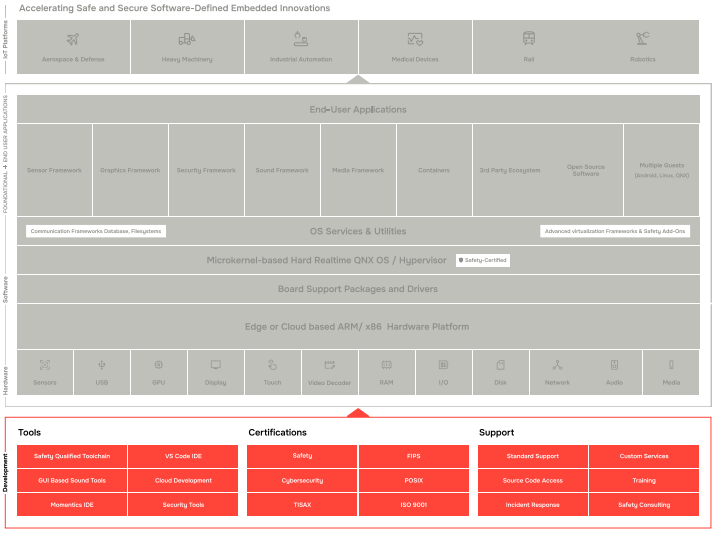


Figure 26: Development Tools, Certification Artifacts and Support Models

QNX General Embedded Development Platform's development tooling contains tools, certification artifacts, and flexible support and training options for accelerated development of embedded systems that have to address dynamic product requirements, driven by the ongoing demand for new features and the need to comply with ever-evolving regulatory standards.

Development Tooling: Tools

QNX Software Development Platform (SDP)
QNX SDP provides the core development tools and runtime components required to build QNX-based products. Also, the well-integrated, high-performance QNX General Embedded Development Platform's foundational software stack is built using these field-proven runtime software components from the QNX SDP. It contains more than 32,000 binaries, libraries, scripts, and configuration files, and 14,000 pages of documentation.

QNX SDP incorporates security and safety-by-design and is compatible with standard tools, libraries, and frameworks. It consists of three primary components:

- QNX Operating System (OS) components
- QNX Tool Suite
- QNX Software Center

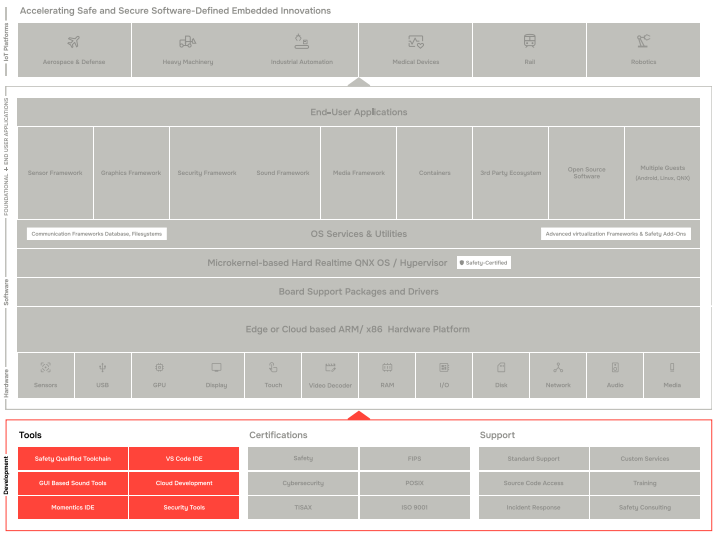


Figure 27: Development Tooling - Tools

QNX Tool Suite

The QNX Tool Suite is comprised of the following components:

- QNX Momentics IDE
- QNX® Toolkit for Microsoft Visual Studio Code
- QNX® Command Line Tools

QNX Momentics IDE is an industry standard Integrated Development Environment (IDE) that helps to quickly set up the project, select a programming language, choose a target processor, compile the code, connect to the target, transfer the application to the target, then run, debug, profile and fine-tune the application. This tool suite provides the flexibility to use the IDE as primary development interface, or to use command-line tools (compiler, linker, debugger) to do the development.

The Momentics IDE can be used to modify and configure QNX runtime software, and to develop applications to run on the QNX OS. It includes compilers, debuggers, libraries, header files, utilities, sample source code, test suites, performance optimization tools, etc., within an integrated development environment that is based on the open Eclipse IDE framework.

Figure 28: QNX System Profiler Tools

Additionally, QNX Momentics IDE provides rich GUI based tools, including:

- Memory Analysis and several Valgrind tools to find memory problems such as leaks and corruption, and to measure memory usage of programs.
- Application Profiler, Valgrind and Cachegrind tools, which measure the performance of programs running on targets.
- System Information and QNX System Profiler tools to understand process and thread interaction on the target, reduce application and system startup times, and debug deadlock and improper CPU usage levels.

It also lets you use commercial test frameworks to write unit tests and then execute them by launching a project. While running a test program, the Code Coverage tool can be used to determine how much of the code is exercised (covered) by the tests. The test frameworks supported are:

- Boost.Test Library
- GoogleTest Framework
- Qt Testing Framework

QNX Toolkit for Microsoft Visual Studio Code is also available which can be used instead of the Eclipse based Momentics IDE. The Microsoft Visual Studio Code (VS Code) is a host-based code editor optimized for building and debugging modern applications and includes the following:

- Thousands of third-party extensions
- Advanced Git integration; cloud and container workflows
- Intellisense smart code completion and dynamic syntax error Highlighting

The QNX Toolkit for Microsoft Visual Studio Code is a QNX extension that is available on the Visual Studio Marketplace, providing an alternative development environment to the QNX Momentics IDE and includes support for QNX-specific functions such as:

- QNX System Information
- QNX System Profiler
- QNX Target Management

The QNX Command Line Tools component is a comprehensive set of development utilities to create and manage executables, object files, libraries, and other operations such as system profiling and debugging capabilities.

The QNX Tool Suite also supports **QNX Accelerate** initiative. QNX Accelerate enables access to AWS or Microsoft Azure cloud-based targets, running foundational QNX OS/Hypervisor software, including safety-certified variant of the software. The same QNX Tool Suite can be used for both the cloud and edge hardware target boards. The developers can quickly innovate and collaborate on safety and non-safety projects using cloud instances of QNX software that offer binary parity, ensuring that the code will work as intended on the embedded target board.

QNX Software Center

The QNX Software Center is a software delivery tool used to manage discovery, delivery, and dependencies of QNX development products in a centralized fashion. With the QNX Software Center, QNX developers are proactively alerted when relevant security updates, patches, and new product releases are posted, or updates related to QNX product licensing arise. The developers can also quickly export their product development environment, saving time and effort required to install the tools and working configuration for additional team members.

Digitally signed software delivery associated with the QNX Software Center is designed to ensure the integrity of software packages. QNX Software Center also provides enhanced artifacts to assist developers with compliance as it pertains to open-source licensing and QNX runtime licensing.

Development Tooling: Certifications

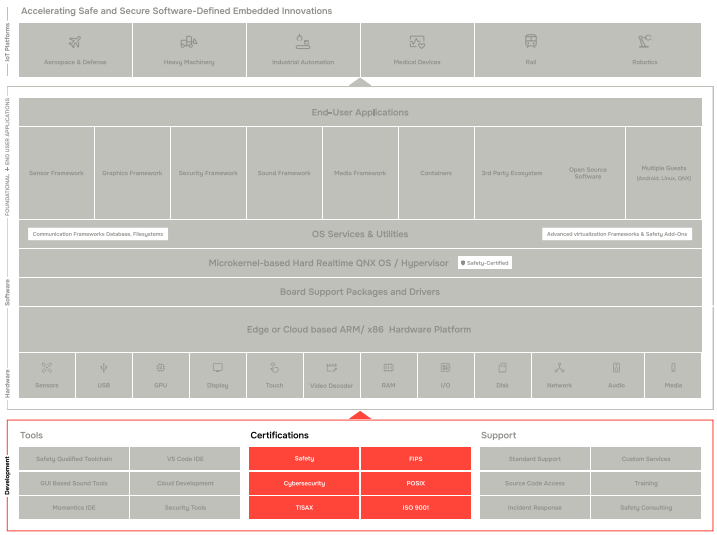


Figure 29: Development Tooling – Certification Artifacts

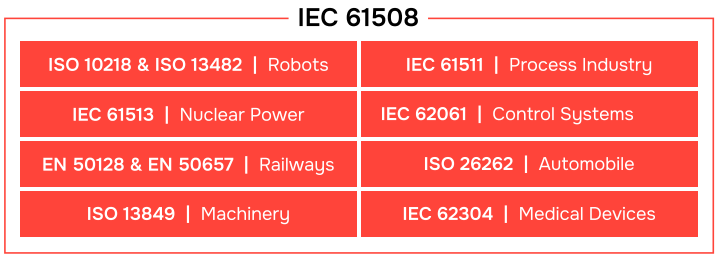


Figure 30: IEC 61508 - a generic functional safety standard

Safety certificate artifacts related to the pre-certified microkernel based QNX OS for safety, QNX Hypervisor for Safety, several safety certified middleware software components along with pre-qualified compiler toolchains are provided to accelerate the overall certification of safety-critical embedded systems, for various industries.

Development Tooling: Support & Training

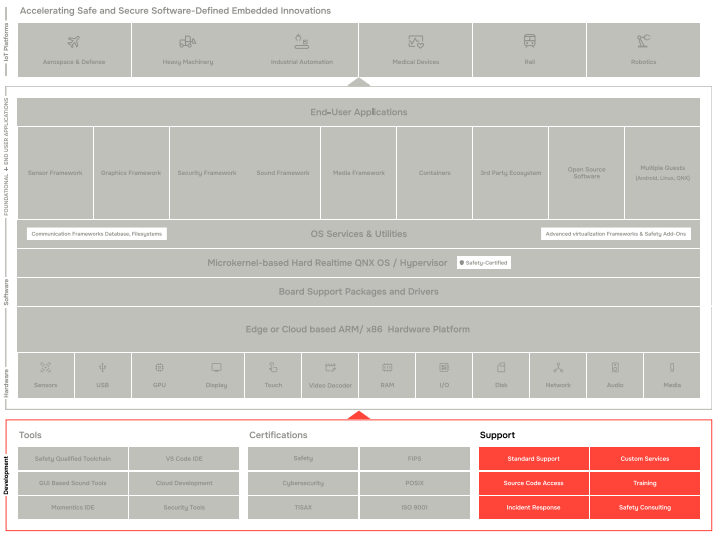


Figure 31: Development Tooling – Support and Training Options

The QNX Support team consists of developers, engineers, and architects with diverse skill sets who not only provide customer support but also work on QNX core products. Various support options are available through a dedicated online portal, person-to-person help lines, a community portal and QNX online knowledge base. Long-term support options are available for products that have extended product lifecycles beyond 10 or more years.

The QNX team also offers customized professional services to bring safe and reliable products to market on time, on budget and with excellent quality and has decades of experience successfully delivering embedded systems for organizations in multiple industries. Flexible engagement models are available based on time-and-materials or fixed-cost, and provide the options of either remote delivery or on-site consulting.

Figure 32: QNX Safety Services for successful product launches

Custom software development services are offered across a range of embedded systems in automotive, medical, robotics, industrial automation, defense and aerospace and other industry verticals.

Having certified QNX products to ISO 26262 (ASIL D), IEC 62304 (Class C) and IEC 61508 (SIL 3) with a 100 percent success rate, safety services are also offered to reduce risk and streamline the development of safety-certified embedded products.

Additionally, QNX training courses are available. Hosted on-or off-site, these are hands-on and instructor-led using real-world examples to give any development team the jumpstart needed in QNX best practices.

For more information about the components of the QNX General Embedded Development Platform and its releases, please reach out to your QNX representative.

February 2025

QNX

About QNX

QNX, a division of BlackBerry Limited, enhances the human experience and amplifies technology-driven industries, providing a trusted foundation for software-defined businesses to thrive. The business leads the way in delivering safe and secure operating systems, hypervisors, middleware, solutions, and development tools, along with support and services delivered by trusted embedded software experts. QNX® technology has been deployed in the world's most critical embedded systems, including more than 255 million vehicles on the road today. QNX® software is trusted across industries including automotive, medical devices, industrial controls, robotics, commercial vehicles, rail, and aerospace and defense. Founded in 1980, QNX is headquartered in Ottawa, Canada.

Learn more at qnx.com →