Windows® Embedded Automotive 7

A Technical Companion to Windows Embedded Automotive 7

Proven technology adapted for the automotive industry

Published: July 2010

For the latest information, please see:
www.windowsembedded.com/auto

Abstract

Windows® Embedded Automotive 7—based on the newest generation of embedded operating systems from Microsoft and combining the award-winning Windows® Automotive and Microsoft® Auto platforms—is designed specifically for developing state-of-the-art, in-vehicle infotainment systems. It offers a standardized, industry-proven platform for building communication, entertainment, and service-enabled location-based solutions. This release of Windows Embedded Automotive includes a large set of integrated, tested, and flexible middleware components and tools, in addition to hundreds of components that are available with Windows® Embedded Compact 7. These components make it possible for Windows Embedded Automotive 7–based systems to scale across a broad range of automotive makes and models. By capitalizing on these tools and on the broad Microsoft partner ecosystem, suppliers can reduce development costs and speed time-to-market while extending customers' lifestyles into the vehicles that they drive.

Table of Contents

Table of Figures

Overview

As an industry, telematics and in-vehicle infotainment is evolving. Consumers increasingly expect on-the-go access to multimedia content and productivity applications, driving market opportunities for integrated solutions. Consumers want vehicle-based systems that let them use their personal digital devices and formats, and they look for innovative, connected entertainment, driver assistance, productivity, and communication services in the vehicles they buy.

Windows® Embedded Automotive 7 leads the automotive infotainment industry by empowering manufacturers and suppliers to offer vehicle-centered, connected software-plus-services that are designed to transform the driver and passenger experience. With more than 10 years in the automotive space, Microsoft is invested in helping manufacturers and suppliers meet evolving customer demands and compete globally. As part of the Windows Embedded portfolio, the automotive platform offers a broad partner ecosystem that encompasses 8.5 million developers worldwide to provide ongoing support and resources.

Windows Embedded Automotive provides a flexible, robust, high-performance platform that makes it possible for manufacturers and suppliers to create differentiated, market-driven solutions, helping them succeed in today's competitive in-vehicle technology space. This latest release builds on the Microsoft® award-winning automotive platforms—Microsoft® Auto and Windows® Automotive—with new features and functionality; Windows Embedded Automotive 7 is a smart choice for bringing telematics (such as navigation) and infotainment solutions to market faster with lower costs.

This white paper introduces Windows Embedded Automotive 7 and explores the many benefits that Windows Embedded Automotive 7 provides to automakers and tier-one suppliers; it also describes many of the platform's unique features in detail.

This white paper is divided into several sections:

- The first section, The Business Case for Windows Embedded Automotive 7, discusses the challenges that automakers and suppliers face in bringing integrated solutions to market. It also explains how the Windows Embedded Automotive 7 platform can help overcome those challenges by providing the foundation for quickly and reliably creating a broad range of extensible, customizable, and advanced in-vehicle solutions.
- The second section, Deep Dive into Windows Embedded Automotive 7, provides a close look at many of the features of Windows Embedded Automotive 7.
- The third section, Delivering a Windows Embedded Automotive 7–Based Solution, describes the tools that are included in Windows Embedded Automotive 7 for building innovative solutions with features that drive sales and build customer loyalty.
- Finally, this white paper includes a summary, links to further information, and appendices that delve further into some Windows Embedded Automotive 7 features. It also includes an extensive glossary that defines the terms and the acronyms that are used in this white paper.

Note: For the latest information about Microsoft Embedded Automotive 7, visit the following web site:
www.windowsembedded.com/auto

The Business Case for Windows Embedded Automotive 7

The market is ripe for a new generation of in-vehicle systems. Consumers are demanding on-the-go access to multimedia content and productivity applications: they want in-vehicle infotainment solutions that let them use their existing digital devices and formats, including mobile phones, MP3 players, DVDs, and CDs. They want innovative, connected services for entertainment, driver assistance (such as navigation and emergency calling), productivity (such as email, web browsing, and calendaring), and communication (including conferencing and calling)—all seamlessly integrated, as if the vehicle were just another node on the home and office network.

The Infotainment Market Opportunity

There is a healthy market for in-vehicle infotainment devices and services. As of 2007, there were more than 254 million vehicles registered in the United States, according to the National Highway Traffic Administration.[1] The 2008 annual Gallup survey of Work and Education finds that American workers report spending an average of 47 minutes commuting to and from work in a typical day, and this does not include time in the car for errands and other trips.[2] Consumers are looking for ways to make these hours more productive and entertaining; car owners now expect their vehicles to not only serve as a mode of transportation, but also to provide an extension of their regular lives.

Hands-free driving legislation is being adopted in a growing number of countries and U.S. states. Because of this legislation, original equipment manufacturers (OEMs) are under significant pressure to create cost-effective solutions for bringing hands-free devices to market, in addition to a rise in consumer demand for Bluetooth® wireless technology. The European Union is working to make eCall a mandatory service in all new cars starting in 2010, potentially propelling European telematics to the forefront and driving the inclusion of speech and Bluetooth technology in in-vehicle systems.

"The success of SYNC proves that customers want to be connected…the premium SYNC adds at auction, and the improvements in purchase consideration show that it is a true differentiator for us, adding real value for the customer."

~Ken Czubay,
Vice President of U.S. Marketing, Sales, and Service
Ford Motor Company

The market for "green" in-vehicle solutions and innovations is also growing, and Microsoft sees green in-vehicle solutions as a key responsibility to help OEMs build environmentally friendly concepts on the Windows Embedded Automotive 7 platform. For example, with Blue&Me (powered by a previous version, Microsoft Auto), Fiat Auto Group built the eco:Drive solution, which helps consumers monitor their cars' behavior and therefore become more environmentally considerate.

Infotainment capabilities also help sell cars. A recent study by the Ford Motor Company clearly shows that cars equipped with Ford SYNC, a fully integrated in-vehicle communications and entertainment system, command a higher resale value than cars without SYNC. The study

[1] http://www.bts.gov/publications/national_transportation_statistics/html/table_01_11.html
[2] http://www.gallup.com/poll/1720/Work-Work-Place.aspx#2

looked at the 2008 Ford Focus—a Focus with SYNC sold for more than $200 on average than a Focus without SYNC.[3]

There is no doubt that the era of infotainment devices is here. By standardizing on a reliable, updatable software platform—such as Windows Embedded Automotive 7—carmakers can maximize sales and keep their vehicles up to date with the latest consumer and technology trends.

Introducing Windows Embedded Automotive 7

Windows Embedded Automotive 7 aims to bring the vehicle into today's digital lifestyle by delivering entertainment and communications features that afford the opportunity to drive new revenue through remote services. Windows Embedded Automotive 7 also lets automakers and their partners move closer to their customers, accurately matching mobility products and services to those who demand them.

Built on a Solid Foundation: Windows Embedded Compact 7

Windows Embedded Automotive 7 is based on Windows® Embedded Compact 7, the next generation of the Windows Embedded family (Microsoft® operating systems designed for use in embedded devices, such as kiosks and handheld devices). Windows Embedded operating systems are available only to OEM system builders, who make it available to users on OEM hardware.

Windows Embedded Compact 7 is a componentized, real-time operating system that is the successor to Windows® CE. With Windows Embedded Compact 7, hardware manufacturers and developers get the resources they need to bring high-performing devices to market quicker, with support for multicore CPUs, along with the latest Advanced RISC Machine (ARM)–based architecture and tools, including Platform Builder, the Microsoft® Visual Studio® development system, Expression Blend®, and Silverlight® for Windows Embedded.

Windows Embedded Automotive is based on Windows Embedded Compact 7, which offers:

- ***Tools*** to bring devices to market faster.
- ***Technologies*** that enable immersive device scenarios.
- ***A rich user experience*** on devices.

What else does Windows Embedded Compact 7 bring?

- **Base operating system.** The core operating system technologies and drivers serve as the foundation for a wide variety of automotive devices.
- **Connectivity.** New networking and transport technologies for connected device scenarios.
- **Multimedia**. Multimedia technologies deliver high-quality user experience across multiple media types.
- **Browser**. Updated browser for rich, user-friendly web browsing experience on devices.
- **User experience**. Silverlight for Windows Embedded, the power of Silverlight running natively on the device.

[3] http://media.ford.com/article_display.cfm?article_id=32262

Built on Previous Success

Previously, Microsoft offered two platforms on which carmakers and tier-one suppliers could build their infotainment solutions: Microsoft Auto and Windows Automotive.

Microsoft Auto: The Technology

Microsoft Auto provided an integrated middleware stack and hardware reference design for a robust starting point. Built on the Windows® CE operating system, Microsoft Auto used an open architecture that made it possible for carmakers and tier-one suppliers to extend the functionality with their own custom solutions and to use their own tool set to design a custom user interface (UI).

Windows Automotive: The Tools

For those who preferred to have complete control over the implementation of their devices, Windows Automotive provided a stable platform on which to build. Windows Automotive includes all the benefits of Windows CE, such as Platform Builder, plus several tools built especially for the automotive industry. The Automotive User Interface Toolkit (AUITK) lets customers develop automotive-specific, custom UIs, while the Automotive System Toolkit (AST) provides tuning and fast cold boot capabilities. The AST includes tools to help optimize performance through CPU management, diagnostics, and error handling. Like Microsoft Auto, Windows Automotive also features a set of common application programming interfaces (APIs) that are familiar to developers the world over.

The Best of Both Worlds: Windows Embedded Automotive

The industry asked for Microsoft Auto and Windows Automotive in one platform, and Microsoft delivered—successfully merging the tools and technology of Microsoft Auto and Windows Automotive to provide a complete solution for carmakers and tier-one suppliers.

These tools and technologies rest on a Windows Embedded Compact 7 base, which adds the newest in Microsoft technologies, including:

- **Silverlight for Windows Embedded.** A native code (C++) user interface framework that lets you dramatically improve the UI on devices while reducing the time—and cost—to deliver a differentiated experience.
- **Microsoft speech engines.** The speech engines that were developed for the Kia UVO are now available to all customers. Microsoft continues to support third-party speech engines, such as those from Nuance, SVOX, and Loquendo.

What's New for Windows Automotive Customers?

For those familiar with Windows Automotive, Windows Embedded Automotive 7 offers many new features and capabilities, including:

- **Hardware.** The development platform is a hardware implementation of all Windows Embedded Automotive 7 features, which helps to facilitate rapid prototyping.
- **Board Support Packages (BSPs)/Drivers**. BSPs and drivers are available in the Windows Embedded Automotive 7 Platform Development Kit (PDK) and through hardware suppliers.
- **Middleware.** A rich set of middleware and services, including a Bluetooth wireless technology stack, phone modules, and radio and media modules.

- **Application cores.** The application cores are the most visible part of the software platform; they are organized and structured so that programming knowledge and techniques can be reused.
- **Microsoft speech engines and Silverlight for Windows Embedded.**

What's New for Microsoft Auto Customers?

Those familiar with Microsoft Auto can enjoy the new tools that the Windows Automotive development environment (the Automotive Adaptation Kit [AAK]) brings to Windows Embedded Automotive 7:

- **Next-generation Automotive System Tools**. The AST tools support the stable integration of advanced, high-performance systems. They include improved test modules and easy-to-use product engineering guidelines to help simplify the development process and increase reliability.
- **A wider selection of middleware components.** These include Windows® Internet Explorer® and Windows Media® technology, required for the development of an automotive multimedia system
- **Microsoft speech engines and Silverlight for Windows Embedded.**

The Windows Embedded Automotive 7 Vision

Windows Embedded Automotive 7 is based on a vision to enrich the in-vehicle experience for drivers and passengers with an industry-leading platform that provides integrated services for communication, entertainment, navigation, and information for the mass market.

- **Enriched in-vehicle experience.** The technology in Windows Embedded Automotive 7 is designed to provide easy-to-use infotainment experiences for the front and rear seats, facing drivers and passengers alike. The platform provides update capabilities to help ensure state-of-the-art support for modern consumer electronic devices.
- **Industry-leading platform.** Windows Embedded Automotive 7 reduces the cost and time-to-market of telematics and infotainment solution development with integrated communication and entertainment features. Windows Embedded Automotive 7 provides a connectivity platform for navigation and information service features. Customers can use the tools in Windows Embedded Automotive 7 to create unique solutions that let automakers and suppliers set themselves apart from the competition.
- **Integrated communication, entertainment, navigation, and information.** The integrated components in Windows Embedded Automotive 7 help automakers and tier-one suppliers connect drivers with a wide range of devices, services, and technologies, such as hands-free Bluetooth phone communication and high-fidelity

Windows Embedded Automotive is committed to providing:

- *Integrated features*—High-quality state-of-the-art communication, entertainment, and connectivity features at low cost.
- *A robust platform for infotainment*—Platform helps reduce risk, shorten time-to-market, and drive down the overall cost for suppliers and automakers.
- *Rich tools*—Industry-leading tools for application and human–machine interface (HMI) development by engineers and designers.
- *Relevance*—Upgradable to evolve with consumer trends and technology, boosting customer satisfaction and loyalty.
- *Connectivity*—The latest in networking and transport technologies.

digital entertainment. These technologies are not delivered in silos—separated from each other as individual components; rather, they are integrated to provide a seamless experience for customers. Windows Embedded Automotive 7 does not define or require a specific operation concept or HMI. Therefore, the infotainment system can match the look and feel of the automaker's or tier-one supplier's target customer.

- **For the mass market.** The Windows Embedded Automotive 7 platform lets suppliers reduce design and engineering costs so that they can create devices at a lower per-device cost. Windows Embedded Automotive 7 helps suppliers get to market faster through the use of standardized components and partners. Therefore, automakers can bring features that were once found only in luxury vehicles to consumers of any vehicle across all model lines.

Feature Highlights

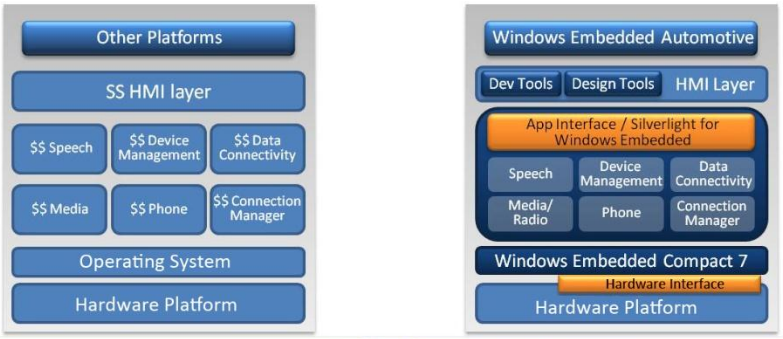
Table 1 shows highlights of Windows Embedded Automotive 7. Note that acronyms and unfamiliar terms are defined in the Glossary.

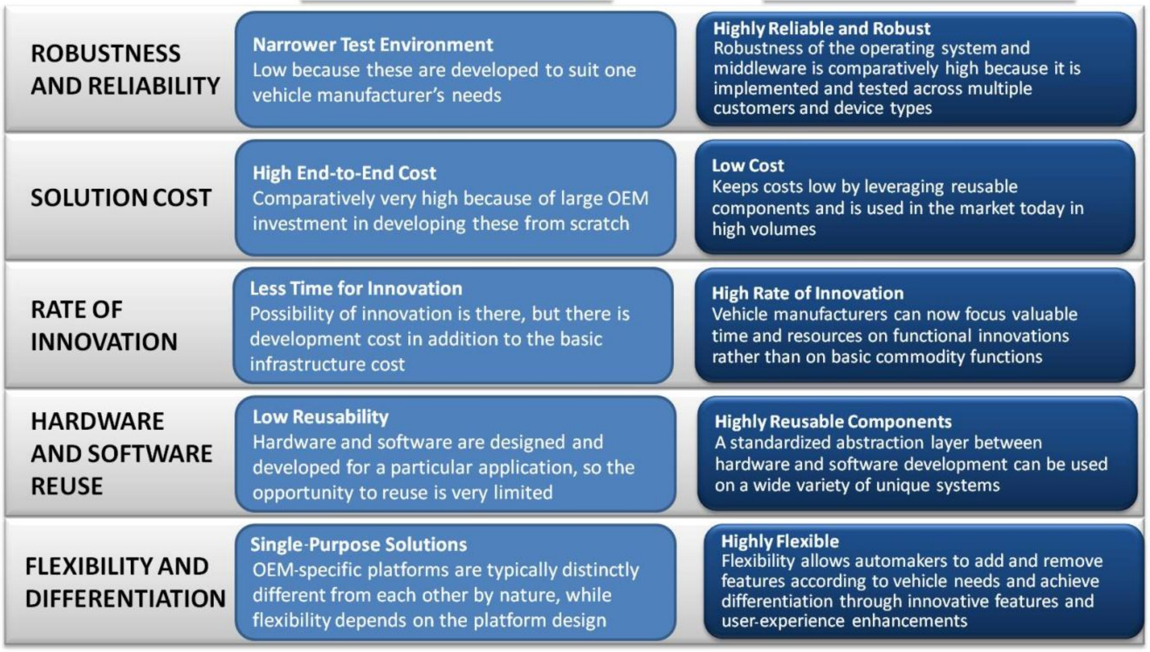
Table 1. Highlights of Windows Embedded Automotive 7

Systems	Phone	Media	HMI	Speech	Software Development Environment
Image Update • Signed images • Delta packages	**Phone Core** • HFP • Call mgmt	**Media core** • Browsing & Indexing • Playback control • Now Playing List mgmt	**Silverlight for Windows Embedded**	**Speech engines** • Microsoft • Third party	**Microsoft® Visual Studio® 2008**
Boot loaders • Fast cold boot • Snapshot boot	**Bluetooth** • 2.1+EDR • CSA1 • Secure Simple Pairing (SSP)	**Device categories** • iPod/iPhone • Zune • MTP • USB MSD • Audio CD • Data CD • DLNA • DMS • M-DMS	**PC Runtime**	**SAPI 5.41**	**Expressions Blend/Windows Embedded for Silverlight tools**
Power management • Auto State machine • Suspend/ Resume • Delayed reboot	**CellCore** • Embedded cellular • 3G • RIL: Cinterion AC75i and HC25	**Radio core** • AM/FM • HD	**Compositor**	**Speech Service**	**Device Management**
Audio • Arbitration • Routing • Multi-zone • AEC/NS	**Phonebook, Calendar, and Task Mgmt** • PBAP • SyncML • GSM-AT • Flashfile • POOM				**Auto System Tools (AST)**
Graphics Drivers • OpenVG 1.1 • OpenGL ES 2.0	**Messaging** • AT-SMS • MAP email and SMS				**Advanced Logging**
Networking • CAN/IPC • MOST • IEEE-1394					**CPU Time Measurement Tool**
Watson/device tools					**Memory Measurement Tool**

Benefits of Windows Embedded Automotive 7

As a world-leading developer of operating system software, the choice of Microsoft for a robust and reliable operating system is clear. Over 80 percent of information technology resources on a typical manufacturing plant floor run on Microsoft platforms and technologies, which represent some of the most mission-critical elements of an IT infrastructure. Microsoft has many years of experience in the consumer electronics and services sector and is ideally suited for providing standard interior vehicle interfaces so that drivers can use their own portable devices. By providing standard interfaces (or APIs), Microsoft can save automakers and suppliers the time it takes to develop basic functions, leaving them free to customize their products. Some of the benefits of using standard Microsoft interfaces are shown in Figure 1.





ROBUSTNESS AND RELIABILITY	**Narrower Test Environment** Low because these are developed to suit one vehicle manufacturer's needs	**Highly Reliable and Robust** Robustness of the operating system and middleware is comparatively high because it is implemented and tested across multiple customers and device types
SOLUTION COST	**High End-to-End Cost** Comparatively very high because of large OEM investment in developing these from scratch	**Low Cost** Keeps costs low by leveraging reusable components and is used in the market today in high volumes
RATE OF INNOVATION	**Less Time for Innovation** Possibility of innovation is there, but there is development cost in addition to the basic infrastructure cost	**High Rate of Innovation** Vehicle manufacturers can now focus valuable time and resources on functional innovations rather than on basic commodity functions
HARDWARE AND SOFTWARE REUSE	**Low Reusability** Hardware and software are designed and developed for a particular application, so the opportunity to reuse is very limited	**Highly Reusable Components** A standardized abstraction layer between hardware and software development can be used on a wide variety of unique systems
FLEXIBILITY AND DIFFERENTIATION	**Single-Purpose Solutions** OEM-specific platforms are typically distinctly different from each other by nature, while flexibility depends on the platform design	**Highly Flexible** Flexibility allows automakers to add and remove features according to vehicle needs and achieve differentiation through innovative features and user-experience enhancements

Figure 1. Benefits of Windows Embedded Automotive 7

According to the 2009 Best Global Brands List by Interbrand, Microsoft is the third-strongest brand worldwide, with a brand value of over 56 billion U.S. dollars.[4] This is a unique asset that no other player in the automotive supplier environment is able to offer.

Benefits for Carmakers

With accelerating product development cycles and increasing pressure on development costs, design flexibility is crucial to staying ahead of the competition with device compatibility updates and features that let drivers to stay connected. Windows Embedded Automotive is an investment in a high-performance, extensible platform; OEMs can use Windows Embedded

[4] http://www.interbrand.com/best_global_brands.aspx

Automotive to seamlessly integrate consumer electronics devices and services into vehicles, with a short development time, and while maintaining competitive price points.

Benefits for Suppliers

Standardizing development on one platform maximizes supplier capability to meet a broad range of OEM requirements across multiple geographies. Windows Embedded Automotive provides a scalable platform with layered software architecture and standardized interfaces that maximize code reuse so that suppliers can focus on innovation. In addition, powerful development tools provide flexibility with a wide variety of supported third-party components.

Benefits for Consumers

Escalating consumer demand for sophisticated vehicle entertainment and communication systems does not have to be at odds with affordable vehicle pricing. In-vehicle solutions developed on the Windows Embedded Automotive platform offer integrated features that keep consumers connected with familiar content, while making it easier to keep vehicles updated and compatible with the latest consumer devices and in-vehicle services.

Microsoft: The Right Partner for You

When you use Windows Embedded Automotive 7, you gain the advantages that come with working with Microsoft, a known, stable company that has a long history of customer commitment.

- **Microsoft is committed to understanding and serving consumers** at work, at home, and on the go. Microsoft products and services reach more than 420 million households, delivering digital experiences to more than a billion consumers every month.
- **Microsoft is committed to planned innovation that evolves predictably**, innovation that is compelling today and remains compelling tomorrow. Planned innovations eliminate the risk that is associated with using “rip-and-replace” alternatives. Those alternatives may seem worthy today, but unless they are financially supported, they may not be there tomorrow, leaving the responsibility for ongoing maintenance, support, and technical evolution to others.

Key reasons why Microsoft is the right partner for you:

- We deliver state-of-the-art software platforms.
- We apply our software assets for lower project implementation risk and faster time-to-market.
- We employ a collaborative approach to achieve solution fit and knowledge transfer.
- We use world-class partners and technology centers to conduct efficient and professional projects.

Partnering with Microsoft also makes the best use of relevant Microsoft assets, as shown in Figure 2.

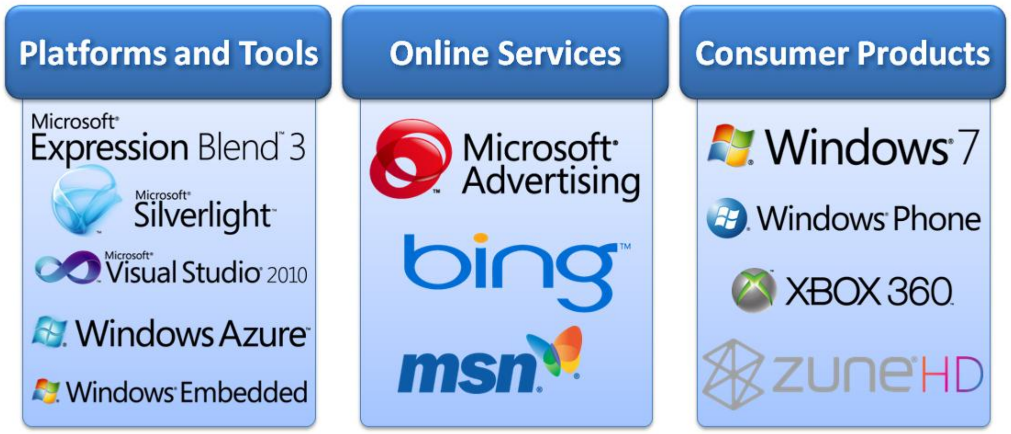


Figure 2. Microsoft assets

Windows Embedded Automotive 7 Development Ecosystem

Both Microsoft and customers are working toward a common goal—developing high-quality in-vehicle infotainment devices in the shortest time possible and in innovative ways. The Windows Embedded Automotive 7 development ecosystem, which includes Microsoft engineering and the Partner Solutions Team (PST), provides support options to help ensure that this goal is reached.

Microsoft Engineering

The headquarters-based engineering team develops the platform, the applications, the middleware, the online services, and the tools that make up the Windows Embedded Automotive 7 system. Located in the new "Studios West" campus in Redmond, Washington, this engineering team is co-housed with the Windows Embedded Compact, Zune®, and Windows® phone engineering teams—engineers of the components that are provided with Windows Embedded Automotive 7. This proximity lets the teams collaborate closely—engineers stay well connected to the communication and entertainment trends and keep Windows Embedded Automotive 7 at the leading edge of innovation.

Partner Solutions Team

In addition to the standard product support that Microsoft offers in several payment models, Windows Embedded Automotive 7 also provides access to a highly skilled team: the Partner Solutions Team (PST).

The PST option gives customers the opportunity to secure Windows Embedded Automotive–managed engineering resources that are dedicated to the success of their projects. These resources will work on any tasks that the customers and Microsoft agree are appropriate, particularly focusing on project elements that are technically complex and difficult to implement. The ultimate objective of the PST engineers is to accelerate the project's time-to-market and to make sure that the project satisfies all of the necessary requirements.

Extensive Partner Ecosystem

The Windows Embedded Automotive partner ecosystem connects Microsoft, tier-one suppliers, and other industry partners in a thriving community of Microsoft platform expertise. It is a means to identify partners of every type, all around the world and at every phase of the development cycle, helping automakers and suppliers find and support their next customer. The partner ecosystem can help automakers and suppliers bring their products to market faster, drive down overall cost, and gain competitive advantage in the marketplace.

The Windows Embedded Automotive partner ecosystem provides simplified discovery of qualified independent software vendors, system integrators, and hardware vendors. Providing technical training and support for this growing partner community is also at the forefront of the expanding Windows Embedded Automotive partner program. (See Figure 3.)

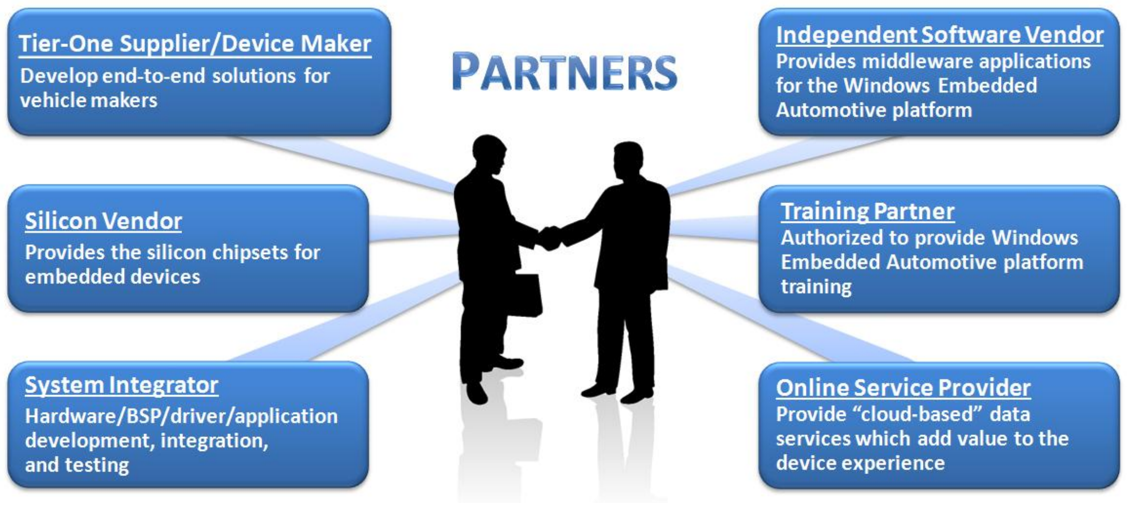

PARTNERS
Tier-One Supplier/Device Maker
Develop end-to-end solutions for vehicle makers
Silicon Vendor
Provides the silicon chipsets for embedded devices
System Integrator
Hardware/BSP/driver/application development, integration, and testing
Independent Software Vendor
Provides middleware applications for the Windows Embedded Automotive platform
Training Partner
Authorized to provide Windows Embedded Automotive platform training
Online Service Provider
Provide "cloud-based" data services which add value to the device experience

Figure 3. Windows Embedded Automotive partner ecosystem

The Windows Embedded Automotive partner ecosystem provides automakers and suppliers with increased business opportunities, market awareness, and technology advice to build next-generation infotainment.

- **For carmakers**: Grow the business and get smart, connected devices to the market faster. Gain strength in the marketplace and differentiate from the competition.
- **For suppliers**: Shorten development time and risk of development delays by using a robust and extensible platform with inherently flexible entertainment and communication applications that are built in.

Through a broad partner ecosystem, leadership in the consumer market, and over a decade of experience in the automotive industry, Microsoft helps automakers and suppliers accelerate telematics design cycles to quickly adapt to consumer electronics trends and innovations. Microsoft has an ongoing commitment to the automotive industry to help ensure that vehicles can stay relevant with feature updates, device compatibility updates, and driver updates, which keep the platform connected to new devices, applications, and content.

Success Stories for the Windows Embedded Automotive Platform

In the competitive in-vehicle technology space, Windows Embedded Automotive provides a flexible, robust, high-performance platform that lets manufacturers and suppliers create differentiated, market-driven solutions. The platform's extensibility lets carmakers keep their devices up to date with the latest technological trends.

Today, drivers and passengers can experience Windows Embedded Automotive 7 in more than 80 vehicle models worldwide.

Fiat Blue&Me

In less than two years, Fiat Group Automobiles and Microsoft created a new infotainment concept from scratch: Fiat Blue&Me. Blue&Me empowers customers to connect their personal mobile devices with the integrated solution found in many vehicle models from Fiat, Alfa Romeo, Lancia, Iveco, and Fiat Light Commercial Vehicles.

The competitively priced, Windows Embedded Automotive-based infotainment package is voice controlled and comes with Bluetooth wireless technology and USB connectivity, letting drivers connect a large number of mobile phone models in addition to media players. The Blue&Me system is based on a modular structure and it can be easily updated to support different services. For example, customers can download language packs from the Fiat web site and update their system to support a language that was not originally installed or they can add a new application as soon as the carmaker makes it available.

As of January 2010, more than **1 million vehicles** equipped with Microsoft-powered Blue&Me have been sold.

First presented in 2006, Blue&Me was originally developed as an infotainment system capable of allowing mobile phones and MP3 players to be used in the car safely with voice recognition commands and steering wheel controls. In 2008, the system evolved further with the introduction of eco:Drive, a free application exclusive to Fiat that helps drivers understand how their driving style can affect consumption and CO_2 emissions. Fiat continues to press forward with development of the Blue&Me system: from Blue&Me Nav to TomTom integration, from Nokia Ovi integration to eco:Drive—numerous applications continue to make Fiat Blue&Me an international success.[5]

With an original design developed with Magneti Marelli specifically for the Fiat 500, Blue&Me Nav represents a new frontier in portable navigation systems market.

Since its launch in 2006, Fiat eco:Drive has already reached more than **80,000 downloads**. As a result, users have analyzed over five million journeys, improving their fuel consumption and CO_2 emissions. Some 2,400 tons of CO_2 have been saved, an amount equivalent to the CO_2 emitted to light a city of 30,000 inhabitants for five years.[6]

- **Blue&Me Nav** extends the phone and media functionalities with a Global Positioning System (GPS) device system and an embedded phone. It provides a simple, user-friendly satellite navigation system that uses pictograms, and it can be activated by voice command or by buttons on the steering wheel. Blue&Me Nav with Services offers SOS emergency, information services, and insurance services that can be personalized.
- **Blue&Me MAP** is a multi-functional portable navigator that gives drivers a completely integrated and connected infotainment experience. The Blue&Me TomTom personal navigation device helps find the best route based on date and time of travel, and locates the nearest gas station, points of interest, and emergency services within an area.
- **Fiat eco:Drive** collects all the necessary data that relates to vehicle efficiency and, through the Blue&Me USB port, transmits the data to a standard USB key that the driver plugs into a PC. The Fiat eco:Drive system presents the driver with detailed

[5] http://download.microsoft.com/download/3/0/2/302323E4-0BC5-4FB1-83CE-024EDDC9A945/FIAT_Release.pdf
[6] http://download.microsoft.com/download/3/0/2/302323E4-0BC5-4FB1-83CE-024EDDC9A945/FIAT_Release.pdf

environmental performance factors, including the CO_2 emission level for each trip. Available free of charge via web download, it analyzes the drivers' style and recommends modifications to reduce CO_2 emissions and to save money on fuel. For more information, see the Fiat.co.uk|Eco:drive web site.

The Microsoft partnership with Fiat has garnered several industry awards for innovation, including the following:

- The "Excellence in Technology of the Year Award for European Automotive Telematics and Infotainment Market" from Frost & Sullivan.
- The "Telematics Update of Detroit," which recognized Blue&Me as the best telematics solution of the year.
- The "Eurostars 2006" prize from Automotive News Europe.

Ford SYNC

SYNC Powered by Microsoft

Ford and Microsoft have had great success in their decade-long partnership to provide customers with superior in-car experiences. The Ford SYNC communications and infotainment system, built on the Windows Embedded Automotive platform, has been installed on more than 2 million Ford, Lincoln, and Mercury vehicles since its launch in 2007.

SYNC comes standard on all 2010 Lincoln models, and is available on select 2010 Ford and Mercury models.

And not content to rest on past success, Ford has continued to expand SYNC with new features and functionality. Ford has developed and brought more than six different applications for SYNC to market. Ford is also able to deliver software updates from Microsoft to SYNC using USB, helping drivers stay connected to the latest devices that come on the market. Best of all, these features follow drivers to any vehicle that is equipped with SYNC via the drivers' smartphone.

SYNC's success continues to grow, from industry awards, skyrocketing sales, and a growing numer of rapidly developed applications. (See Figure 4.) Ford's game-changing vehicle connectivity model and affordability have helped make it one of the largest technology introductions in the industry, launching on 12 vehicles simultaneously.

A 2008 Ford Focus with SYNC sold on average for $240 more than a similar vehicle without SYNC.

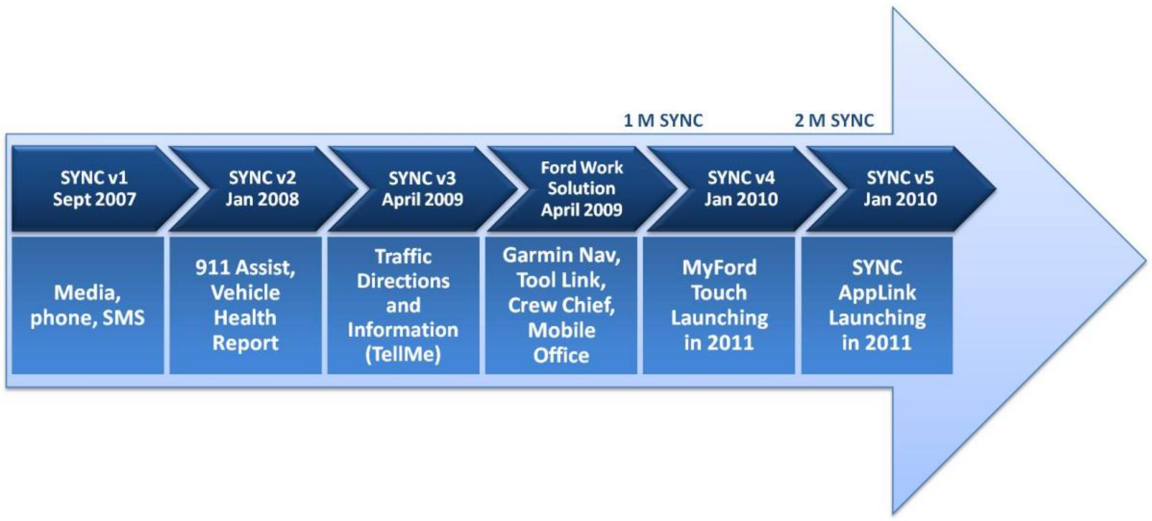

1 M SYNC
2 M SYNC
SYNC v1 Sept 2007
SYNC v2 Jan 2008
SYNC v3 April 2009
Ford Work Solution April 2009
SYNC v4 Jan 2010
SYNC v5 Jan 2010
Media, phone, SMS
911 Assist, Vehicle Health Report
Traffic Directions and Information (TellMe)
Garmin Nav, Tool Link, Crew Chief, Mobile Office
MyFord Touch Launching in 2011
SYNC AppLink Launching in 2011

Figure 4. SYNC road map

SYNC helps sell Ford vehicles. Ford sold its two-millionth SYNC-equipped vehicle only 10 months after the one-million mark. In a recent study conducted by Ford, customers reported that SYNC played an important role in their purchasing decision. A full 80 percent of customers who watched a SYNC demonstration said it improved their opinion of Ford, and 70 percent said that they were then more likely to consider purchasing a Ford vehicle.

Ford SYNC and Applications

Ford SYNC is a factory-installed, fully integrated in-vehicle communications and entertainment system that provides drivers with hands-free voice-activated control over mobile phones and digital music players. It automatically connects phones and music players with the vehicle's in-vehicle microphone and sound system. Most popular media players work with SYNC, including iPod, Microsoft Zune, "Plays for Sure" devices that are certified for Windows Vista® or compatible with Windows® 7, and most USB storage devices. Supported audio formats include MP3, Advanced Audio Coding (AAC), Windows Media Audio (WMA), and Waveform Audio Format (WAV).

- **SYNC with Traffic, Directions, and Information** provides personalized, real-time information to help drivers reach their destination with the information they need. SYNC with Traffic, Directions, and Information was developed in partnership with TellMe, a Microsoft subsidiary. SYNC with Traffic, Directions, and Information provides simple, hands-free access to personalized traffic reports, precise turn-by-turn driving directions, and up-to-date information, including business listings, news, and sports and weather updates.
- **911 Assist**, an update to SYNC, connects drivers and passengers through their mobile device to 911 operators in the event of an airbag deployment.
- **Vehicle Health Report** shows vehicle diagnostics, scheduled maintenance and recall information, and lets drivers schedule service online at www.syncmyride.com.

Ford Work Solutions

An award-winning in-dash computer, Ford Work Solutions was developed by Ford and Magneti Marelli and is powered by Windows Embedded Automotive 7. It provides high-speed Internet access over the Sprint Mobile Broadband Network and navigation by Garmin. This system lets customers print invoices, check inventories, and access documents stored on their home or office computer networks—right from the job site.

MyFord Touch

Powered by the second generation of Ford Sync, MyFord Touch offers improved voice recognition, touch-sensitive buttons, touch screens, and thumb-wheel controls to replace most of the knobs and switches a driver must usually contend with. MyFord Touch also boosts Wi-Fi and connectivity with the help of a secure digital (SD) card slot.

"SYNC's open platform approach is unique in the industry and allows us to capitalize on the ever-improving capabilities of mobile devices…Smartphone mobile apps are experiencing explosive growth, and consumers are becoming dependent on them for news, entertainment and information, so we're excited to be working with some of the most popular apps on the market, OpenBeak, Pandora, and Stitcher, to showcase the potential of the SYNC API."

~Doug VanDagens
Director of Connected Services Solutions Organization
Ford Motor Company

In January, Ford announced that Pandora, Stitcher, and Orangatame's OpenBeak (formerly TwitterBerry) are the first partners to let their applications be controlled in the car by SYNC using the new API in the SYNC software development kit (SDK). By making it possible for developers to integrate the SYNC API into their applications, Ford is providing customers with the capability to access the applications they use most while in the car. With SYNC voice commands and steering wheel controls, drivers can keep their hands on the wheel and their eyes on the road. The open API model makes it possible for Ford to use the mobile device as an operating system and processor, while using SYNC to access the vehicle controls.

Kia UVO

Kia Motors and Microsoft have introduced UVO, powered by Microsoft, a new, intelligent in-car communications and entertainment system with voice- and touch-activated experiences. Through this innovative system, drivers and passengers can quickly and directly access music files, operate a rear-view camera, change radio stations, make or answer phone calls, and more through voice- or touch-activated controls. By supporting complex grammar, UVO needs only short voice commands to connect drivers and passengers with their desired functions. An interactive system, UVO responds to inquiries such as “What’s playing?” and provides audible answers and related functions, letting drivers’ eyes stay safely focused on the road.

"UVO powered by Microsoft is a breakthrough for in-vehicle infotainment...With UVO, Kia is able to offer drivers a rich telematic experience and a clear competitive advantage, allowing Kia drivers to safely and easily use all of their personal technologies and create personalized in-car communications and entertainment experiences."

~Michael Sprague
Vice President of Marketing
KiaMotors America

Key features of UVO powered by Microsoft:

- **Advanced Speech Recognition**Support for large grammar commands and faster response time means that the content is delivered when you ask for it. Kia Motors’ UVO system is the first in-vehicle solution to integrate full Microsoft speech engine technology.
- **Natural Interface Advancements:** A full-color, easy-to-use in-dash monitor lets occupants quickly scroll through media and mobile device content through intuitive voice and touch-screen commands.
- **Custom Media Experiences with MyMusic**: UVO’s “Jukebox” function features a 1-gigabyte (GB) hard drive for media storage so that users can rip music from CDs or an MP3 player into personal MyMusic folders and store up to 250 songs, sorted by title and/or artist—all through voice commands. The system can shuffle through an MP3 player or AM/FM and SIRIUS® radio stations and instantly identify what’s playing, all through simple voice commands.
- **Rear Backup Camera**: When the vehicle is put in reverse, a built-in rear backup camera uses the UVO in-dash display to provide a clear image of the environment behind the car so that the driver can identify objects that may be difficult to see otherwise.
- **Ability to Continuously Update Features and Services**: Based on a flexible Windows Embedded Auto platform, updates and services can be delivered in a number of ways (over the air, over the web), so Kia can continue to provide a superior user experience after the system enters the market.

UVO demonstrates the flexibility and reliability of the Windows Embedded Auto software platform and is the first in-vehicle solution with full Microsoft speech engine technology.

UVO, powered by Microsoft was honored with the 2009–2010 Industry Newcomer Award at the annual Telematics Update Awards Conference, the world's biggest forum for telematics industry leaders that recognizes achievement and innovation within the telematics market.

Daimler Car2Go

With car2go, Daimler is launching a completely new mobility concept and is providing a future-oriented answer to increasing traffic volume in urban areas through a car-sharing project that uses Smart Fortwo vehicles with micro-hybrid drives. Daimler chose Microsoft as its technology partner because Microsoft provides a comprehensive end-to-end solution.

The head unit and a powerful back-end infrastructure are the building blocks for communication and sending services to the car and back. Delivered by Magneti Marelli, the head unit is based on Windows Embedded Automotive, and the infrastructure is a 100-percent Microsoft platform and solution.

As the volume of vehicles increases and the concept extends to further global markets, car2go will use Windows Azure™ as the future software-plus-services platform.

PACCAR SmartNav

Microsoft and PACCAR used the Windows Embedded Automotive software platform to develop a new in-truck system, SmartNav, which is tailored specifically to the commercial trucking industry. SmartNav gives truck drivers simple and convenient access to a mix of business tools, vehicle monitoring features, and entertainment options, from enhanced navigation and diagnostics for commercial trucks to hands-free phone calling and music management. All features are accessed through voice or touch-enabled commands.

Used for the first time in the transportation space, Microsoft Silverlight helped PACCAR create a rich and engaging user experience on the full-color, high-resolution 7-inch touch screen that provides real-time vehicle monitoring, truck navigation, hands-free Bluetooth connectivity, voice recognition, an integrated audio system with satellite radio, in addition to MP3 and iPod capabilities with an internal storage space of 8 GB.

Alpine Blackbird and NVE-N872A

Alpine Electronics of America built a portable navigation device called Blackbird and a new aftermarket in-vehicle navigation system, the NVE-N872A, on Windows Embedded Automotive technology. The navigation systems provides satellite-guided turn-by-turn directions and useful features such as voice-guided controls and information on more than 7 million points of interest, including Zagat Survey restaurant guide information, to make getting from place to place easier than ever.

Pioneer PAIS

Pioneer Electronics has developed a Windows Embedded Automotive–based content and services platform, the Platform for Aggregation of Internet Services (PAIS), which provides a seamless home/car/work experience for different content sources and services in conjunction with any connected device. The PAIS platform presents open-standard interfaces for voice, navigation and maps, local search, social networking, music and radio, and video and television.

Award Highlights

In 1999, Microsoft developed the first infotainment system, Auto PC. Auto PC won a “Best of What’s New” award from *Popular Science* and received an “Excellent” rating as a navigation system from a leading consumer product rating publication. Auto PC was described as “revolutionary,” “redefining the industry,” and “innovative.”

Microsoft Auto and Windows Automotive continued to win awards over the following years, including the J.D. Powers & Associates Customer Satisfaction Award for Alpine’s Windows Embedded Automotive–powered device in 2006 and the2010 Consumer Electronics Show Innovation Award for Ford Work Solution and SYNC connectivity systems powered by Windows Embedded Automotive.

For a more complete and up-to-date list of the awards won by Windows Embedded Automotive 7 and its predecessors, see Appendix 2: Award Highlights.

Deep Dive into Windows Embedded Automotive 7

Windows Embedded Automotive 7 is an ideal application development platform—it provides a rich programming environment that lets application software developers add their own functionality so that they can create a broad range of advanced, in-vehicle solutions that meet the growing needs of their consumers while setting themselves apart from the rest of the field. The flexible Windows Embedded Automotive 7 platform targets a wide range of devices, including connectivity gateways, connected radios, and multimedia devices.

This section of the white paper takes a more detailed look at the components of the Windows Embedded Automotive 7 platform. First, the paper discusses the individual components of the platform. Then, the white paper describes the tools that can be used to put the components together to provide a foundation for a range of in-vehicle devices.

Windows Embedded Automotive 7 Components: The Basic Building Blocks

The following diagram gives an overview of the various building blocks that make up the Windows Embedded Automotive 7 system; some are provided by Microsoft, while others are provided by partners. (See Figure 5.) Note that acronyms and unfamiliar terms are defined in the Glossary.

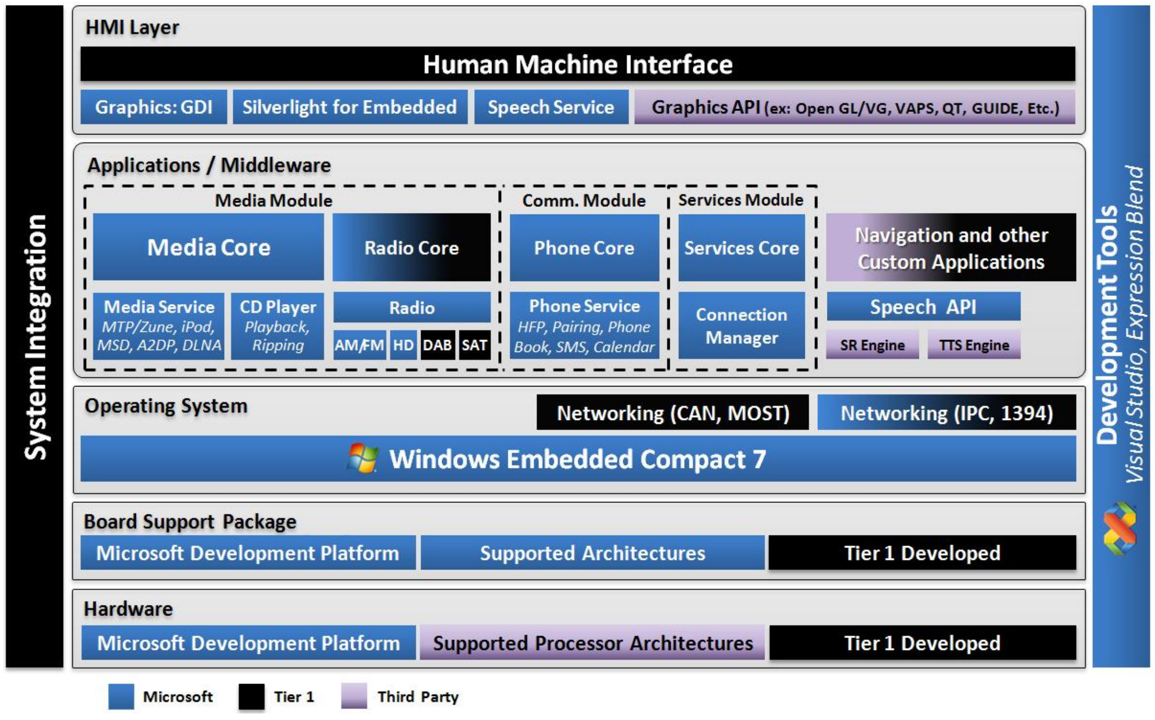


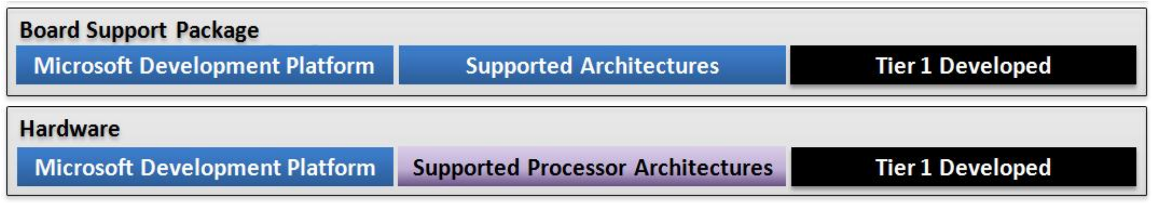
Figure 5. Windows Embedded Automotive 7

Windows Embedded Automotive 7 includes the following components:

- **Hardware**. The Windows Embedded Automotive 7 development platform is a hardware implementation of all Windows Embedded Automotive 7 features that facilitates rapid prototyping. This development platform is built on a Freescale i.MX35 processor. Other hardware options are available from Microsoft silicon partners, which include Freescale, Intel, NVIDIA, Renesas, Samsung, and Texas Instruments.

- **Board Support Packages/Drivers**. The Windows Embedded Automotive 7 Platform Development Kit provides support for ARM, Intel Architecture (iA), and SH4 architectures. BSPs and drivers are available in the PDK and through hardware suppliers. The PDK includes sample BSPs for the Microsoft® Automotive Reference Platform (MARP-F2) that is based on the Freescale i.MX35, and for the Renesas Pilsner, which is based on a dual-core SH4 processor and is Symmetric Multiprocessor (SMP) enabled. The Microsoft Auto CE PC–based hardware platform (MACEPC) provides a sample BSP for iA.
- **Windows Embedded Automotive 7 base operating system.** Windows Embedded Automotive 7 is built on Windows Embedded Compact 7, providing a component-based, real-time operating system for embedded devices. Windows Embedded Compact 7 uses the same kernel and driver model as Windows Embedded CE 6.0 and has been enhanced to support SMP.
- **Windows Embedded Automotive 7 middleware**. Windows Embedded Automotive 7 provides a rich set of middleware and services, including a Bluetooth wireless technology stack, phone modules, and radio and media modules. These modules enable the creation of integrated applications, such as hands-free phoning, media device integration, and CD and radio support.
- **Windows Embedded Automotive 7 application cores.** The application cores are the most visible part of the software platform. The APIs are organized and structured in a similar way to the desktop version of Windows, so that programming knowledge and techniques can be reused. This makes it possible for new development resources to be deployed and to become productive more quickly. The applications have been designed so that the HMI is easily separated; for example, the media player is composed of a media player core and a supplier-provided application HMI.
- **Third-party and HMI applications**. Unique, innovative components can be easily integrated into the system at any level of the software stack. Additionally, OEMs and suppliers can choose from many development tools and runtime libraries—including Silverlight for Windows Embedded, OpenGL/OpenVG, and the Windows Embedded Compact standard Graphics Device Interface (GDI)—to develop two-dimensional (2-D) or three-dimensional (3-D) graphical HMIs. The layered software architecture permits changes to the user experience without changes to application functionality and provides additional asset scalability and reusability. This portion of the application can be easily changed without disturbing the underlying application.

Hardware and BSP


Board Support Package
Microsoft Development Platform
Supported Architectures
Tier 1 Developed
Hardware
Microsoft Development Platform
Supported Processor Architectures
Tier 1 Developed

This layer includes the hardware and board support package.

Hardware Design

Processor support:

- ARM-based processors such as Freescale's i.MX35
- Intel iA-based processor (x86 or Atom)
- Renesas SH4-based processors (such as dual-core SH4A)

Functionality and features of a system rely on fundamental primitives that are supported by the base system hardware components (such as management of device power states and transitions, management of NAND flash, and support for data transfer over USB ports)—and these are supported by the development hardware.

Windows Embedded Automotive 7 supports a hardware development design, MARPF2, which is based on the Freescale i.MX35 microprocessor. This design provides a good match for modern in-vehicle infotainment and telematic requirements. (Note that suppliers provide the final hardware.) The development hardware design approximates an automotive head unit, including an optical disk drive, networking support, and a multimode radio receiver that supports a number of analog and digital transmission technologies. The MARPF2 includes digital AM/FM/HD radio, CD player, 6 channel audio output, Bluetooth for audio streaming and hands-free phone, Digital Visual Interface (DVI) video output with touch-screen support, USB, and Wi-Fi. The development platform uses NAND flash for non-volatile storage (the board has 128-MB DRAM and 256-MB NAND flash).

Note also that you can purchase a Windows Embedded Automotive Development Kit (which includes the reference hardware platform) from Qualnetics Corporation, a Windows Embedded silver partner. For more detailed product information, please visit Windows Embedded Automotive Development Kit (WE-ADK).

The Windows Embedded Automotive 7 software is also compatible with other ARM-based, SuperH (SH)-based, and iA-based processors, including the newest options for the Intel Atom Z5xx series processors (which offer an industrial temperature range).

Note that the development hardware reference design is not mandatory to build devices that are based on Windows Embedded Automotive 7. Microsoft provides documentation that describes the requirements for transforming a standard Windows Embedded Compact 7 BSP into one that can fully support the functionality of the Windows Embedded Automotive 7 platform. Note also that the Windows Embedded Automotive 7 middleware requires audio arbitration, and therefore the Windows Embedded Compact BSP will not work if it is unmodified.

For additional details about the development hardware reference design and for a schematic of the development platform, see Appendix 3: Windows Embedded Automotive 7 Base Components and Hardware Reference Design.

Board Support Package and Systems

Included board support packages:

- MARPF2 development hardware based on Freescale i.MX35 processor
- Renesas Pilsner development hardware based on dual-core SH4a
- MACEPC development hardware based on x86

Boot times measured by Microsoft on the Windows Embedded Automotive hardware reference design:

- First drivers: 440 ms
- Radio: 680 ms
- Minimal shell: 1.4 sec
- Full sample applications: approximately 5 sec

The BSP is the hardware-specific code that consists of the boot loader, the OEM adaptation layer (OAL), the run-time configuration files, and board-specific device drivers. The BSPs in Windows Embedded Automotive 7 are provided in source code so that they can be customized and adapted to the actual production hardware to be used.

The starting point for a Windows Embedded Automotive 7–compliant BSP is a standard Windows Embedded Compact (formerly Windows CE) BSP. A minimal set of capabilities is required, including the following standard components:

- **Kernel Independent Transport Layer (KITL).** The KITL is the basic debugging protocol used for debugging Windows CE devices. Ethernet KITL is preferred, but USB KITL is also an option.
- **Debug serial port.**
- **USB 2.0**. The host controller interface used to connect the devices (for example, iPod or Mass Storage Device [MSD]).
- **Bluetooth 2.1**. The host controller interface driver to the Bluetooth chip.
- **NAND Flash**. The NAND flash is used to store image and file system. (Note that NOR and Hard Disk Drive [HDD] can be used, but they are not supported out of the box by Windows Embedded Automotive 7.)

In addition to the standard components, a Windows Embedded Automotive 7 BSP requires:

- Audio driver
- Power-aware drivers
- Enhanced boot loaders to support Image Update

Note that many of the drivers are based on the two-layer Windows Embedded CE model-device driver (MDD)/platform-dependent driver (PDD) driver model:

- The MDD layer is completely platform (hardware) agnostic. It implements all operating system entry points and accesses devices indirectly through the PDD layer.
- The PDD layer directly accesses device hardware and is specifically written to the specified platform. The PDD layer almost always needs to be modified when a new hardware platform is adopted.

Figure 6 shows the recommended drivers and boot options.[6]

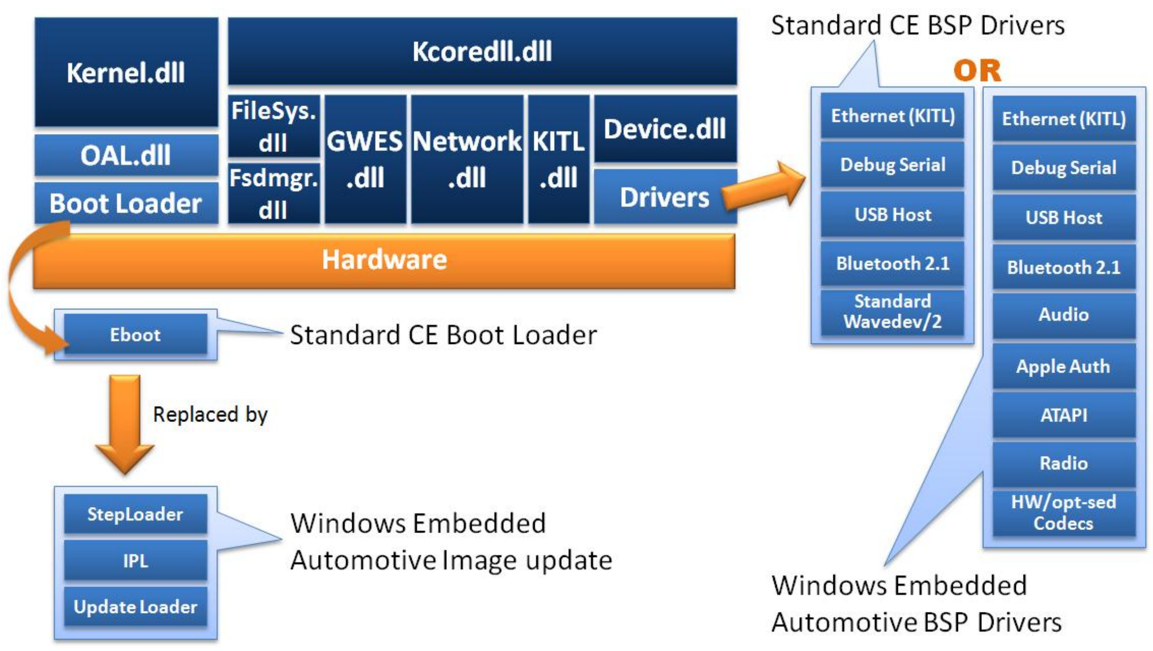


Figure 6. Recommended drivers and boot

Audio Driver

Windows Embedded Automotive 7 uses the standard MDD/PDD driver model for the audio stack based on the WAVDEV2 model. The MDD provides support for dynamic routing, sample rate conversion, and AEC/NS. A platform-specific PDD is required. (See the section The Wavdev Driver in this white paper.)

Note that the current solution meets Verband der Automobilindustrie (VDA) standards for AEC/NS performance, with a sending delay of 67.5 milliseconds (ms) and a receiving delay of 57.62 ms (VDA standard is less than 120 ms), and an echo delay of 125.12 ms (VDA standard is less than 260 ms).

Power-Aware Drivers

Windows Embedded Automotive 7 usage scenarios require the drivers to support “Suspend” (typically when the car engine is turned off) and “Resume” (when the car engine is turned back on) operations. This provides the “instant-on” experience that is unavailable with a cold boot. During a Suspend state, the system has to be placed in a low-power mode to conserve the life of the car battery.

Windows Embedded Automotive 7 uses the Windows Embedded Compact power management mechanism. For a detailed description, see the Power Management web page.

For most drivers, the device power states D0 and D4 (Full On and Off) will have associated code to support Resume and Suspend. The USB driver should also support the D3 power state (Sleep), because it is highly recommended that all external devices are detached during D3, ensuring

[6] Note that the colors used in the figure are for readability only; they have no other significance.

that they will be enumerated upon Resume. See the Power Management section of this white paper for more information.

Enhanced Boot Loaders for Image Update

One of the key capabilities of Windows Embedded Automotive 7 is the ability to deliver signed updates to the field. To support this capability, Windows Embedded Automotive 7 defines a specific approach and provides a set of common boot loader components and a set of build tools. Figure 7 shows a model of the Image Update, with the following components:[7]

- Step Loader
- Initial Program Loader (IPL)
- Master Boot Record (MBR)
- Update Loader (UPL)

The operating system image is partitioned during build time into an image update file system (IMGFS) region and one or more transaction-safe FAT (TFAT) regions. Read-only portions of the NAND flash hold the IPL, the UPL, the device parameter store, and the field-programmable gate array (FPGA) configuration modules.

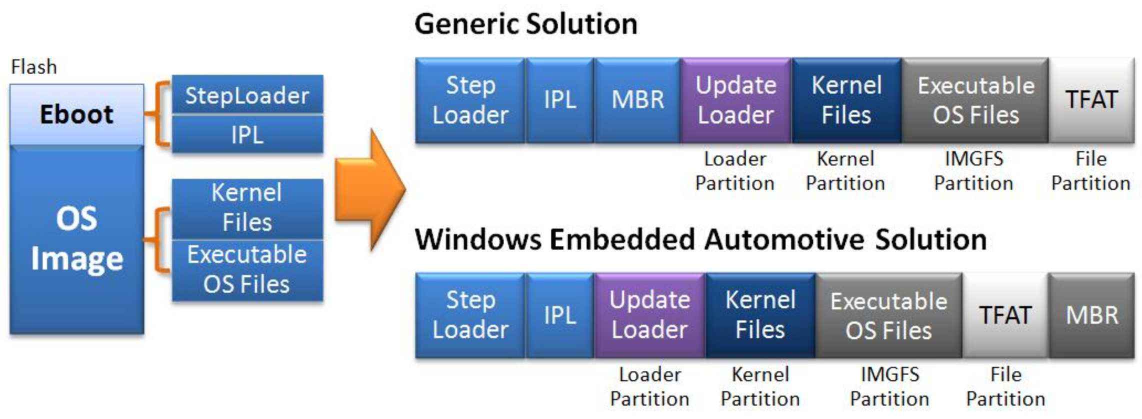


Figure 7. Image Update model

The various boot loaders have specific functions:

- **Step Loader**. The Step Loader is the initial software that is loaded from the first block of flash and is of fixed size (2 kilobyte [KB] on the MARPF2). The Step Loader performs the following tasks:
  1. It initializes and configures the CPU for boot context.
  2. It copies the IPL into RAM and then jumps to the IPL starting address.

  Platforms that do not support booting from NAND flash must support booting from another type of non-volatile store (NVS) such as NOR flash; a Step Loader for that type of NVS must be created. In this scenario, a dummy Step Loader is added to the image that is flashed to NAND.
- **Initial Program Loader.** The IPL module determines the device boot mode (normal boot, update, or development mode) based on a parameter stored in the boot args area (see the Creating Driver Globals and Boot Args web page for more information).

[7] Note that the colors used in the figure are for readability only; they have no other significance.

The IPL performs the following tasks:

1. The IPL determines which component to load depending on the boot mode (the run-time image, the Update Loader, or a boot loader image such as the Ethernet Boot Loader [EBoot]).
2. For each component, the Device Parameter Store (DPS) stores the address of the first block, the number of blocks, and the start address in RAM where the component is to be loaded. The IPL locates the image by reading the DPS and by retrieving the data for the location (on the flash memory), the image size, and the jump address on the synchronous dynamic random access memory (SDRAM).
3. The IPL copies the component from NAND to RAM and then jumps to the image.

- **Ethernet Boot Loader**. The EBoot is a development boot loader that provides a simple menu interface that can be configured or customized to expand upon the default menu options that are present (such as providing test suites for exercising hardware or launching custom boot loaders). The EBoot is loaded by the IPL; the EBoot launches resident images or the downloads and flashes new images onto the board. Note that the EBoot is not part of a final released product image.

  The EBoot performs the following tasks:

  1. It creates and populates the reserved memory area that is to be shared with the operating system.
  2. It sets up the debug Universal Asynchronous Receiver/Transmitter (UART) and provides a user menu for configuring loader options.
  3. It initializes the Ethernet controller.
  4. It obtains the IP address for the target from a Dynamic Host Configuration Protocol (DHCP) server or assigns a static IP address.
  5. It initializes the Trivial File Transfer Protocol (TFTP) connection and, based on the loader options, downloads an image (through Platform Builder) or launches the resident image on the persistent memory.

- **Secure Boot Loader (SBoot).** The SBoot is a fast, more secure USB downloader that can flash images. Like the EBoot, the SBoot is loaded by the IPL based on whether the user selected the SBoot (from the EBoot menu) or whether the platform is put into the SBoot mode (platform specific). The SBoot can this be used outside of EBoot.

  The SBoot itself does not provide any menu options. As soon as the SBoot launches, it tries to connect to the desktop USB downloader application to initiate the image download over USB.

  The SBoot can be included on both development and production images, which require security features for flashing the new images. See the section on the Device Management for more information.

- **Hardware Test Utility (HWTU)**. The HWTU is a suite that is used to test and verify several hardware features and is platform specific. It is loaded and launched by the EBoot via a menu option. Like the EBoot, the HWTU is a development utility and is not shipped as part of the production image.

- **Update Loader (ULDR)**. The ULDR is a functional subcomponent of the Device Management Service, which provides reliable updates to the Windows Embedded CE image. The ULDR uses the Windows® phone model and libraries.

  The Image Updater and the primary image, which contains the installer, depend on the IPL to load the appropriate Windows Embedded CE image (IMGFS image, UPL) based on information that is provided by the OAL.
- **DPS access library.** DPS provides persistent storage that can be accessed in both boot and operating system context. Registry hives cannot be used for this purpose because they are not available to the boot components. Typically DPS is used to store Media Access Control (MAC) addresses (for example, Bluetooth), device manufacturing information, and so on. It also contains the flash Table of Contents (TOC). This provides the start block address and size in blocks of each component in flash (for example, IPL and ULDR). Currently, the DPS only supports NAND flash.

Windows Embedded Automotive 7 boots from flash memory instead of from fixed disk, which greatly reduces the boot time. As measured by Microsoft on the Microsoft hardware reference design, the first drivers are started only 440 milliseconds after the platform has been powered on, and the radio is operational after 680 milliseconds. For more information about the boot loader's operation, see Appendix 4: Boot Times.

Power Management
The goal of power management is to preserve the life of the car battery when the engine is off. Windows Embedded Automotive 7 provides a standard Windows Embedded Compact 7 Power Manager (PM) with a default set of system power states, transitions, and time-outs that works on all platforms. Delayed reboot is also supported. The PM component provides an extension mechanism to support ignition handling. (See Figure 8.)

CE Device State	Registry Key	Description
Full On	**D0**	Full power; full functionality
Low on	**D1**	Full functionality at lower power state than D0
Standby	**D2**	Partial power, standby for wakeup on request
Sleep	**D3**	Sleeping; minimal power for device-initiated wakeup
Off	**D4**	No power

Standard definit
(based on Windo
CE)

System State	State type; Power type	Functionality (Enabled; Disabled)
OFF	Initial; no power	None; all
WAIT_ON	Initialization; fully powered	SOC, audio, USB, 1394, Bluetooth
KEY_ON	Operational; fully powered	All; ignition is on
KEY_ACCESORY	Operational; fully powered	All; ignition is in Accessory position (display off)
WAIT_SUSPEND	Transitional; fully powered	System on Chip (SoC), USB; audio, Bluetooth, IEEE1394
WAIT_OFF	Transitional; fully powered	SOC, USB; audio, Bluetooth, IEEE1394
SUSPEND	Sleep; minimally powered	Wakeup sources, RAM self-refresh; all

Power management enforces an overall system power policy through the following mechanisms:

- **A set of defined device power states.** Each system power state is mapped to standard power states (D0–D4) for various devices. For example, a particular power state might require the Global System for Mobile Communications (GSM) module to be fully powered (D0) but the GPS module to be in suspended mode (D3).

  The device power states are defined by Windows Embedded Compact (based on Windows Embedded CE); the power states have the same meaning for all devices.
- **Defined conditions or triggers.** These triggers cause transitions between the power states. Windows Embedded Automotive 7 supports real-time clock (RTC) wake up on the MARPF2 platform (if programmed).
- **Notifications.** The PDD provides notifications to applications (and to device drivers) about power state transitions. Each application can then choose a policy for functioning in different power states and for actions to perform during the transitions.

The quiescent current for the MARPF2 development platform when in the Suspend state is approximately 1.5 milliamps (mA). Potential wake-up sources for the processors are as follows:

- **Ignition**. Ignition goes on.
- **Controller Area Network (CAN) bus**. The CAN state becomes active.
- **GSM**. A wakeup message from GSM is received (for example, a Short Message Service [SMS] is received).
- **Input 1**. A user-defined wake-source through General Purpose Input/Output (GPIO) is received.

Audio

Windows Embedded Automotive 7 supports multiple audio input sources, multiple audio outputs, and multiple zones. It provides routing and source-rate conversion, AEC/NS, plus the audio driver MDD. The device maker provides the PDD for the specific hardware.

Audio sources include internal sources, such as Advanced Audio Distribution Profile (A2DP) devices, Bluetooth wireless technology–enabled phones, USB devices, and text-to-speech (TTS) devices. Audio output channels include stereo out 1 (front zone of the car), stereo out 2 (rear zone of the car), and mono out (used for TTS and phone audio output, plays the left channel only). The different sources of audio that are processed by the system have different routing requirements, sampling rates, pre-processing and post-processing requirements, and serializing requirements. Audio management functionality is responsible for these requirements and considerations. Note that because only one source can use a codec at a time, arbitration is essential.

Figure 9 shows an overview of the audio system architecture. Terms and acronyms are defined in the Glossary.[8]

[8] Note that the colors used in the figure are for readability only; they have no other significance.

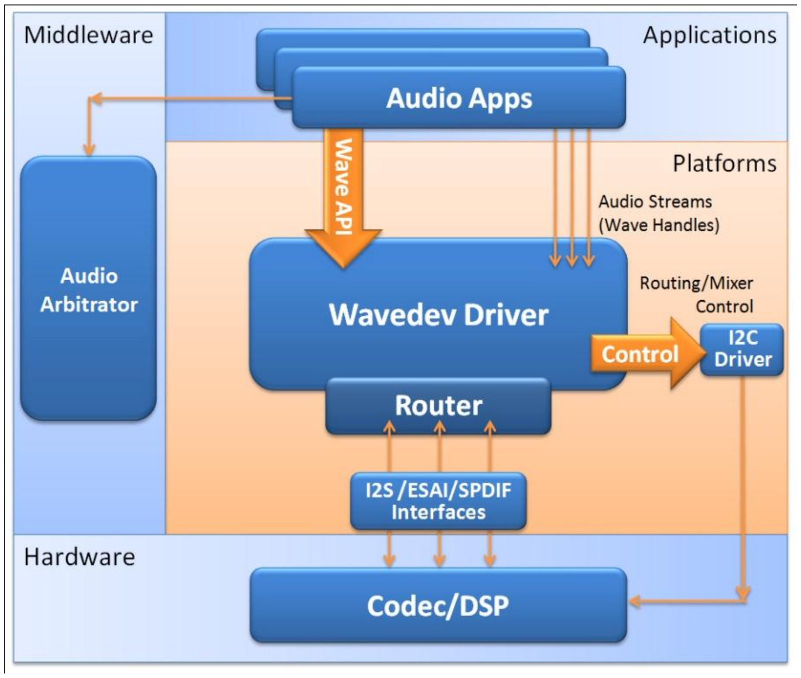


Figure 9. Audio system architecture

The Windows Embedded Automotive 7 audio system includes three layers: application, middleware, and platform.

- **The application layer** includes functionality that presents audio services to the user and enables user control over those services. Examples of audio services include audio control requests and notifications.
- **The audio middleware layer**, composed of the Audio Arbitrator, presents a high-level abstraction of the audio hardware to the application layer and implements the functionality that is required for orderly operation of the audio system.
- **The platform layer**, or Systems Audio Layer (SAL), exposes an abstraction of the audio functionality that is supported by the hardware. The SAL provides processing and multiplexing of audio streams and multiplexing of physical audio channels. These multiplexers are controlled by the Audio Arbitrator for routing of audio. This layer includes the Wavedev driver.

The Wavedev Driver

The Windows Embedded Automotive 7 Wavedev driver supports the standard Windows Compact Wavedev APIs, in addition to automotive-specific features, such as multi-channel audio, the Audio Arbitrator interface, the AEC/NS interface, software sample rate conversion, and command and control of audio and hardware codecs. Note that the Wavedev MDD driver is released in source code format for customization.

All interactions between the Wavedev driver and the application and middleware layers occur through the standard Windows Compact Wave API (as documented in the Windows Compact Help files that are shipped with Platform Builder). Each Wave API request is handled by the Audio Device Manager (which manages audio devices). The Audio Device Manager in turn makes a request to the Wavedev driver through the Wave Device Driver Interface.

In the Windows Embedded Automotive 7 implementation of the Wavedev driver, the application streams (or wave handles) that are opened by using the Wave API are not bound directly to a physical device; the Audio Arbitrator manages the routes or connections between these application streams and the physical audio hardware.

The Audio Arbitrator also interacts with the Wavedev driver through the Wave API by defining custom properties to make additional requests, such as connect/disconnect routes or queries to the connection list. Typically, audio applications use the Wave API interfaces for playback, while the Audio Arbitrator uses the Wave API interfaces for providing routing functionality.

In addition to the Wave API, applications can also interact with the Wavedev driver through the mixer API to query or modify low-level audio-related controls, such as volume, bass/treble, and the equalizer.

Audio Management

High noise environments such as cars need AEC and NS to remove the echo captured by a microphone when a sound is simultaneously played through speakers located near the microphone. Audio routing, echo cancellation, noise reduction, and mixing support are provided through a software library by using parameters that are dependent on the source and destination of the audio. Software lines are triggered by the Wavedev driver or the Hands-Free Profile (HFP) service, depending on the nature of arbitration that is required. For example, when a phone call is accepted, the HFP service instructs the software to allow the Synchronous Connection Oriented (SCO) audio to be output from Windows Embedded Automotive 7 to play over the car’s speaker system.

Networking

- CAN
- IEEE 1394 support
- MOST

Although vehicle network stacks are not part of Windows Embedded Automotive 7, two different models to integrate vehicle networking into a Windows Embedded Automotive 7–based device have been used in current production vehicles. A direct vehicle connection stack runs on the infotainment CPU and implements all vehicle networking protocols. An indirect vehicle connection runs on a secondary CPU, often referred to as the VMCU, with inter-process communication (IPC) that provides the relevant information to the infotainment CPU.

Support for Media Oriented System Transport (MOST) vehicle networking support is available from third parties. Microsoft has worked with K2L, a German software development company, to provide a working MOST stack for Windows Embedded Automotive 7. Third-party products can also be used for other networks, such as Local Interconnect Network (LIN) or FlexRay, if desired.

IEEE 1394 Bus Architecture

Windows Embedded Automotive 7 provides IEEE 1394 interface support for audio and TCP/IP (with Digital Transmission Content Protection [DTCP] support available) for the in-vehicle network, for streaming video, and for music from the head unit to another display in the vehicle.

The IEEE 1394 bus in Windows Embedded Automotive 7, a high-performance serial bus that was specifically designed for entertainment and communication applications, can be used to transfer data at high speeds. Figure 10 shows how an application can use the IEEE 1394 to interact with the Windows Embedded Automotive 7 hardware.[9] Terms and acronyms are defined in the Glossary.

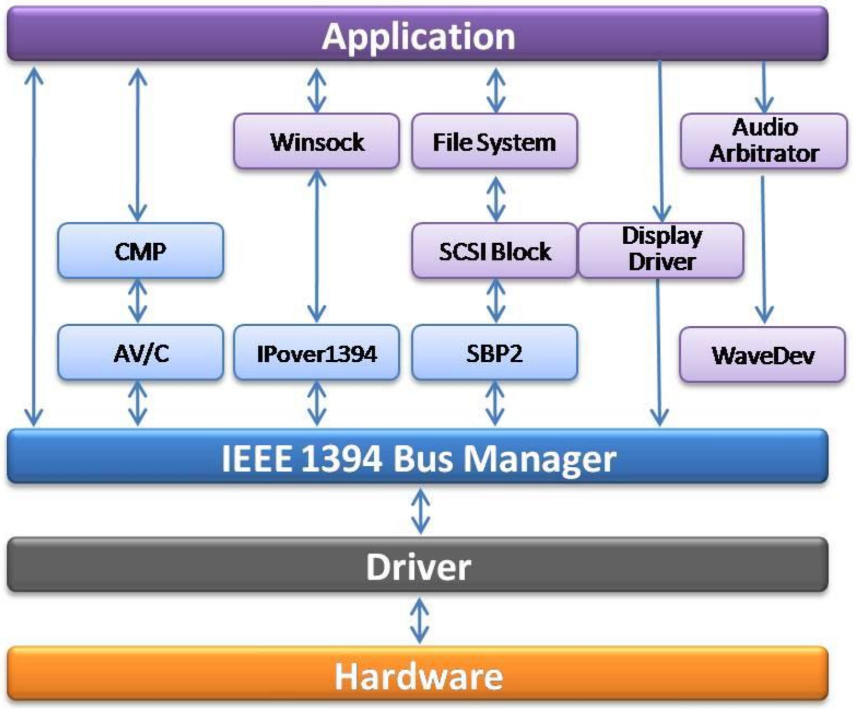


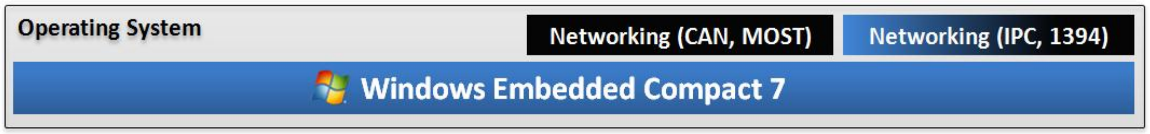
Figure 10. IEEE 1394 architecture

The IEEE 1394 bus consists of the following items:

- **IEEE 1394 hardware**.
- **A driver**.
- **A bus manager**, which is used to interact with the other components.
- **IEEE audio video control protocol (AV/C protocol)**, which is used to manage audio and video devices in the IEEE 1394 network.
- **Connection Management Procedures (CMP)**, which is used to manage broadcast and point-to-point connections in the applications.
- **IP over 1394**, which uses the IEEE 1394 driver to embed TCP/IP packets in IEEE 1394 packets so they can be transmitted on the IEEE 1394 bus. IP over 1394 is a protocol adaptation layer that converts TCP/IP packets into IEEE 1394 standard packets.
- **Serial Bus Protocol 2 (SBP2)**, which is used to handle requests from the file system that are encapsulated within an IEEE 1394 packet and then transmitted on the IEEE 1394 bus.

[9] Note that the colors used in the figure are for readability only; they have no other significance.

Operating System



Windows Embedded Automotive 7 takes advantage of the many features and benefits of the newest in Microsoft embedded technology—the Windows Embedded Compact 7 operating system.

Operating system features:

- Small footprint (300 KB, 700 components)
- 32-bit native real-time support unified kernel
- A Win32® (API) subset, including file and memory management, device and service management, threads and process management, and networking stacks
- Varied networking protocols, security and encryption technologies, Internet client technologies, Wi-Fi, video, GPS support, hard disk support, XML, Internet servers, graphic displays, and file system and database support
- Multilanguage support
- The Microsoft® .NET Compact Framework
- Enables Variance Frequency Processors (VFP) support in ARM processors
- Dual-core support (SMP, ARMv6)

Windows Embedded Automotive 7 is built on top of the Windows Embedded Compact 7 operating system and adds automotive-specific features. Windows Embedded Compact 7 is component based and customizable. Real-time support provides the bounded, deterministic response times that time-critical car infotainment applications require. Flash disk boot capabilities enhance reliability even in challenging temperature and vibration conditions.

Windows Embedded Compact 7 streamlines the development process by taking advantage of developer knowledge of existing tools (such as Microsoft Visual Studio) for creating applications and drivers. From shared source code availability to a broad selection of production-quality drivers, Windows Embedded Compact 7 offers a competitive advantage through faster, more efficient design cycles and an extensive array of features.

Functions of the Base Operating System Layer

The Windows Embedded Automotive 7 platform builds on a specific set of Windows Embedded Compact 7 modules and components that are included in the default operating system image during the System Generation (SYSGEN) build process by way of SYSGEN variables. They provide the platform-level features of the Windows Embedded Automotive 7 system. Device makers who are using Windows Embedded Automotive 7 can customize the Windows Embedded Compact 7 and Windows Embedded Automotive 7 components that are included in their specific device image.

This base operating system layer includes support for:

- File systems.
- Windowing and focus management.
- Access to operations that are supported by the various hardware modules on the system.
- Networking transports and protocols (TCP/IP and Bluetooth wireless technology).

The Windows Embedded Compact 7 kernel exposes core system functionality and provides the infrastructure through which the rest of the system software can interface with the processor-specific code that initializes the CPU, the memory controller, and the other hardware modules in the processor. The kernel is primarily responsible for process and thread management, predictable thread scheduling, memory management, interrupt handling, and support for system calls.

Features of Windows Embedded Compact 7 that are used by Windows Embedded Automotive 7 are shown in Table 2. Acronyms and unfamiliar terms are defined in the Glossary.

Table 2. Features of Windows Embedded Compact 7

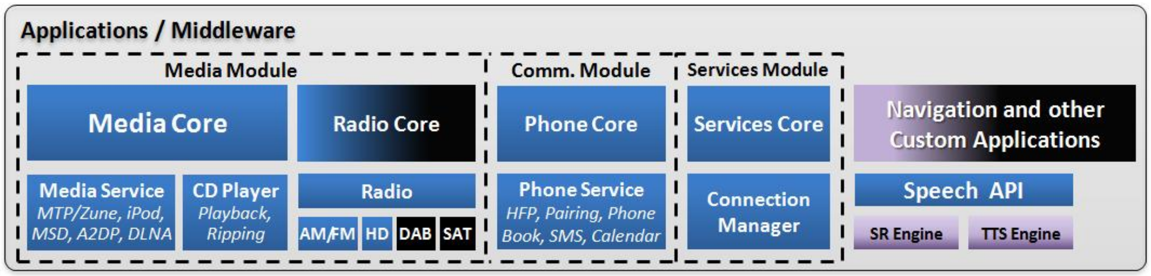
Base operating system	Based on the Windows Embedded CE 7.0 kernel, with 2 GB of virtual memory per process and 32,000 simultaneous processes Updates to the base operating system include: • Support for latest ARM processors with ARMv7 and NEON • SMP • New BSP test suite for improved analysis of silicon platform • > 512MB addressable physical RAM • Kernel cache improvements • Heap improvements • Driver optimizations
Connectivity	Windows Networking Stack (Network Driver Interface Specification [NDIS] 6.1/5.0) Link Layer Topology Discovery (LLTD) technology Wi-Fi, Wireless Provisioning Services (WPS) technology Connection Manager, works with CellCore and Bluetooth Zune Wi-Fi setup Bluetooth 2.1 protocol support for improved performance Profile updates, simple pairing Improved CellCore/RIL Potential for a third-party stack
Multimedia	Media library, management of music, photos, video, TV, podcasts Latest implementation of digital rights management (DRM) Codec optimizations Media Transfer Protocol (MTP) support OpenGL 2.0 DLNA 1.5 support, Digital Media Player, Digital Media Controller, Digital Media Renderer Updated HTTP Streaming, stream high resolution media to Windows Embedded Compact–based devices Buffer filter, improves user experience by reducing skips, jumps, etc. TTS Touch input
Browser	Windows Internet Explorer 7 rendering engine for improved content rendering speed Rich media experience to take advantage of advanced graphical technologies Enhanced XAML host application Customizable browser UI Browser technologies

	• Flash Lite 3.1/Flash 10.1 • Zoom behaviors • Really Simple Syndication (RSS) Feed support • Performance optimizations Possible third-party browser
Business communications	Full end-to-end data experiences (mail, calendar, tasks) Update to ActiveSync®, including AirSync support Update mail infrastructure, new engines Update to POOM infrastructure Third-party application support
User experience	XAML C++ Application Framework • Integrated Designer—Developer experience for efficient UI design and development • Powered by Silverlight XAML presentation engine for rich experiences Add XAML effects to earlier applications Default UI templates, customizable device UI templates to speed device time-to-market Expression Blend tools

For more information, see the Windows Embedded Compact 7 web site. Also see the Related Links section later in this document.

Middleware and Services

The next layers in the software stack are the Windows Embedded Automotive 7–specific middleware and services. These components form the heart of the Windows Embedded Automotive 7 platform; they distinguish this platform from the standard Windows Embedded Compact and from the other platforms that are based on Windows Embedded Compact technology.


Applications / Middleware
Media Module
Comm. Module
Services Module
Media Core
Radio Core
Phone Core
Services Core
Navigation and other Custom Applications
Media Service MTP/Zune, iPod, MSD, A2DP, DLNA
CD Player Playback, Ripping
Radio
AM/FM
HD
DAB
SAT
Phone Service HFP, Pairing, Phone Book, SMS, Calendar
Connection Manager
Speech API
SR Engine
TTS Engine

The middleware components provide a stable foundation for connectivity and consumer device applications. Third parties or suppliers can add additional components that adapt the platform to specific requirements (for example, the MOST network stack).

The middleware layer provides the infrastructure support that is required to easily develop applications while using the powerful features that are exposed by the base platform.

Applications that are written for Windows Embedded Automotive 7 can use a broad array of C/C++ APIs. The base Windows Embedded Compact 7 system exposes an API for its services (including memory and process management, file systems, and registry access). The automotive-specific platform components expose APIs for support of functionality such as speech, hands-free telephony, media playback, and radio.

Telephony and Data Communication

Telephone and data communications are an important in the middleware and services layer.

Windows Embedded Automotive 7 includes a set of services for:

- Hands-free telephony
- AEC/ NS
- Bluetooth profiles
- Phone management
- Data connectivity
- Messaging

Applications that rely on telephony and data communications reside at the topmost layer of the telephony and data communications architecture and use Windows Embedded Automotive 7 middleware services and Bluetooth profiles.

For example, if an application uses Bluetooth wireless technology for data communications, a profile is applied to describe how to exchange specific types of data. An HFP application must apply the HFP profile to define how to use Bluetooth wireless technology to place phone calls, to receive phone calls, and to perform other phone-related functionality on the Windows Embedded Automotive 7–based device using a Hands Free Profile Audio Gateway (a connected mobile phone). Support is provided for HFP calls using the vehicle's audio system and a speech-based and button-based interface.

SMS messages can also be sent or received through Bluetooth-connected phones and through phone modules that are connected through CellCore. Other interesting features—such as the ability to use Windows Embedded Automotive 7 as a Personal Area Networking (PANU), SIM Access Profile (SAP), Bluetooth Dial-Up Networking (DUN) profile gateway Phonebook Access Profile (PBAP), A2DP/ Audio/Video Remote Control Profile (AVRCP) 1.4, Object Push Profile (OPP), and Message Access Profile—are also supported.

Data connections can be established with other data sources through the Bluetooth mobile phone (through the DUN-DT or PANU profiles) or an embedded phone module and can be managed by the connection manager (see the Connection Manager section). The connection manager centralizes and automates the establishment and management of these connections for Windows Embedded Automotive 7 applications by handling the details of each connection.

Figure 11 shows the phone and data communications structure.[10] Terms and acronyms are defined in the Glossary.

[10] Note that the colors used in the figure are for readability only; they have no other significance.

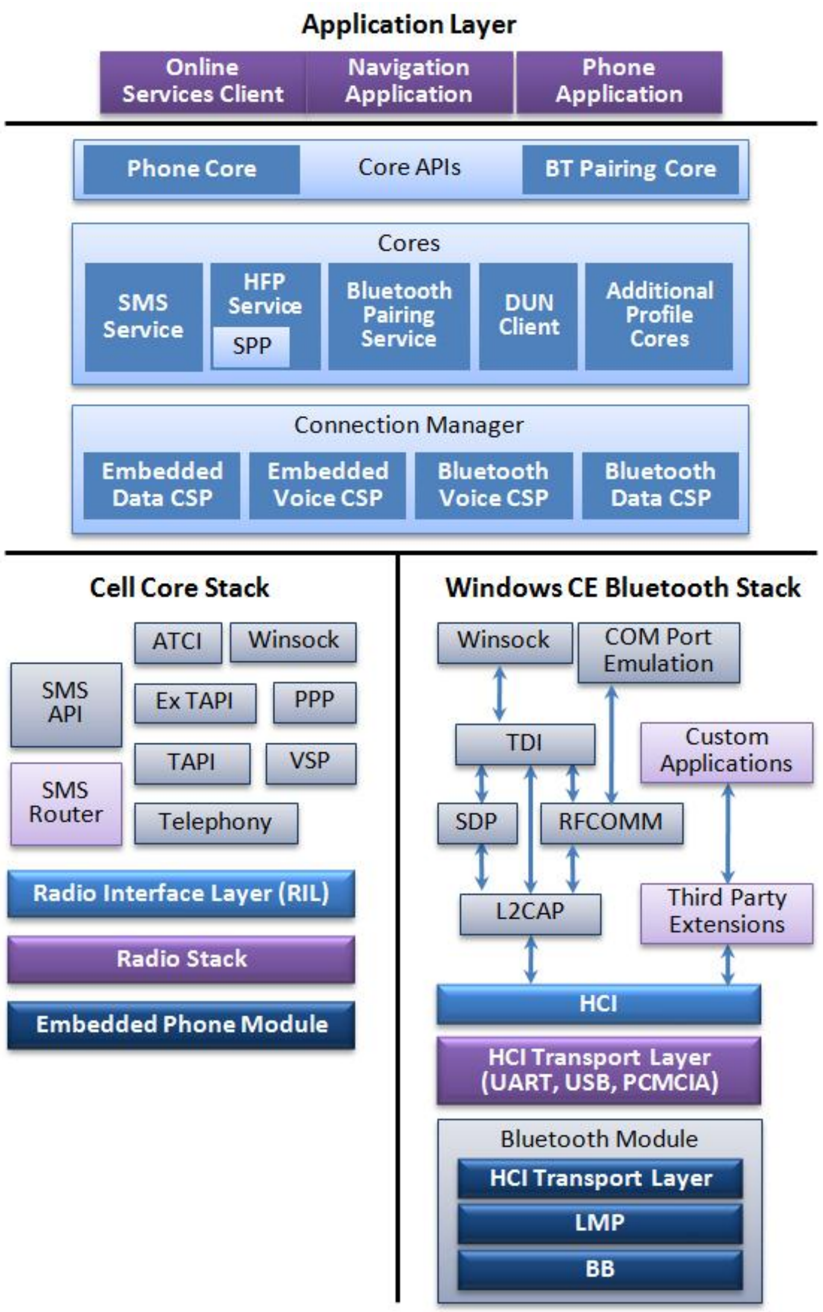


Figure 11. Telephone and communication architecture

CellCore

CellCore is a set of services that lets suppliers create wireless-connection services on a device, including:

- **Radio Interface Layer (RIL)**. The RIL handles the communication between the CellCore system software and the radio hardware.
- **Telephony.** These Telephony programming elements that are applicable to CellCore: Extended TAPI (ExTAPI), Assisted TAPI, and Telephony Service Provider (TSP) API.

- **Wireless Application Protocol (WAP) API.** This open specification defines both a communications protocol and an application environment.
- **WAP Wakeup**. Using the WAP Wakeup feature, an asynchronous application can be available to process incoming messages, yet be neither running nor loaded in memory.
- **SMS provider registration**. SMS providers enable the SMS client to determine the application that should receive an incoming SMS message. This service associates message types with an SMS provider by modifying the registry.
- **Enabling Enhanced Messaging Service (EMS) processing.** EMS processing is disabled by default, but can be activated.
- **SMS API.** This SMS feature supports sending messages of up to 160 characters to mobile-based devices.
- **SIM manager reference**. The Subscriber Identity Module (SIM) Manager API is used to access information stored on the SIM card.

The Windows Embedded Automotive 7 CellCore features 3G support and sample RIL implementation for the Cinterion AC75i and HC25. For more information, see the CellCore web page.

Bluetooth Software Stack

New profile support

- Device ID Profile
- SAP–SIM Access Profile
- Fully integrated Personal Area Network (PAN) and Human Interface Device (HID)
- Windows Embedded Automotive 7 MAP has been tested across a broad set of devices and includes many compatibility improvements over previous implementations, MAP improvements include multiple MAS instance support for multiple MAP mailboxes

Improved troubleshooting

- Supports connection failure responses from Profiles (for example, not paired or not in-range)
- New HCI packet capture capability -> compatible with Frontline FTS4BT for sniffing without the sniffer hardware
- HFPGetLastError feature now includes over 40 additional CME/CMS errors (errors for Global System for Mobile Communication [GSM] devices)

The Microsoft Bluetooth wireless technology stack implementation is a modular, general-purpose Bluetooth 2.1 + EDR–compatible software stack that is built on the Windows Embedded Compact 7 Bluetooth stack. The protocol stack makes up the core portion of the Bluetooth wireless technology implementation. Through a Bluetooth wireless technology connection, devices can exchange data and interact with one another by using the applications. The host controller interface (HCI) software module supports various connections (UART and USB) to the Bluetooth wireless technology chip.

Figure 12 shows a schematic of the Bluetooth software stack.[11] Terms and acronyms are defined in the Glossary.

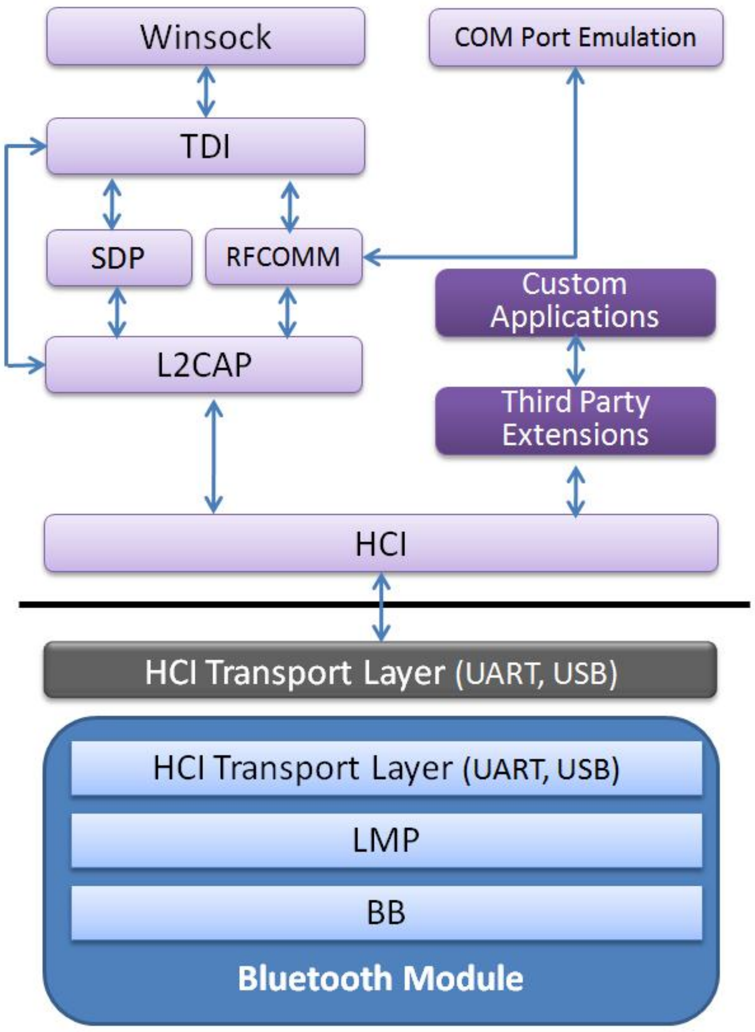


Figure 12. Bluetooth software stack

[11] Note that the colors used in the figure are for readability only; they have no other significance.

Bluetooth Pairing

Supported Bluetooth technologies:

- Generic Object Exchange (GOEP) 1.1
- Object Push Profile (OPP) 1.1
- Serial Port Profile (SPP) 1.1
- Phonebook Access Profile-PCE (PBAP) 1.0
- Advanced Audio Distribution Profile (A2DP)-SNK 1.2
- Audio/Video Remote Control Profile (AVRCP)-Controller 1.4
- Generic A/V Distribution Profile
- Hands-Free Profile (HFP) 1.5 (backward compatible to HFP 1.0)
- DUN Profile-DT and GW 1.1
- Message Access Profile (MAP) 1.0
- SIM Access Profile 1.1
- Device ID Profile 1.3
- Personal Area Network Profile/Bluetooth Network Encapsulation Protocol 1.0
- Human Interface Device Profile 1.0

Windows Embedded Automotive 7 provides the following Bluetooth Pairing features that build on the Bluetooth software stack:

- Bluetooth Pairing Service: A middleware component that manages the service discovery process and the Bluetooth-enabled device-pairing process.
- Bluetooth Pairing Core: An in-process dynamic link library (DLL) that manages data and provides convenient interfaces to the Bluetooth Pairing Service.

Bluetooth Pairing Service

The Bluetooth Pairing Service is a middleware component that manages the service discovery process and the Bluetooth-enabled device-pairing process. It also maintains a database of Bluetooth profile information for each of the paired devices. Multiple phones can be paired (the exact number is configured by the carmaker or supplier).

The Bluetooth Pairing Service provides the following capabilities:

- Enable or disable Bluetooth radio.
- Start or stop discovery of Bluetooth-enabled devices nearby.
- Start or stop pairing with a selected Bluetooth-enabled device.
- Enable or disable Bluetooth discovery mode.
- Signal to the Phone Core API that a device has been paired.
- Provide management capabilities to the Phone Core API to control paired devices' profile information.
- Allow the Phone Core API to activate (or deactivate) a specific paired device (provides the ability for the Phone Core API to append data to the paired device profile record as name/value pairs).
- Support for secure simple pairing.

The Bluetooth Pairing Service itself is not power aware. Because the Bluetooth Pairing Service is designed to be used by other applications in the Windows Embedded Automotive 7 system, the connected applications are expected to maintain their own power state awareness and to use the pairing service when in the appropriate power state.

When discovery is activated, the pairing service waits for a pairing event from the Bluetooth stack. Upon receipt of this event, the device Bluetooth address (BT_ADDR) is checked against the existing set of known devices. If the device is not found, a Secure Simple Pairing legacy or pairing negotiation and link key exchange is initiated. Once paired, the device is queried for a set of profiles that Windows Embedded Automotive 7 can use with the Service Discovery Protocol (SDP).

The list of supported services is stored with the device pairing record in the registry. Higher protocol layers can append name/value pair attributes to each pairing record to support storing custom data. As new devices are added to the device list, the higher protocol layers are signaled by using shared named events. When a paired device is deleted, the information that is related to it is removed.

Figure 13 shows how the Bluetooth Pairing Service fits with the Bluetooth software stack and the Bluetooth Pairing Core, in addition to other components of the telephony and data communications stack.[12]

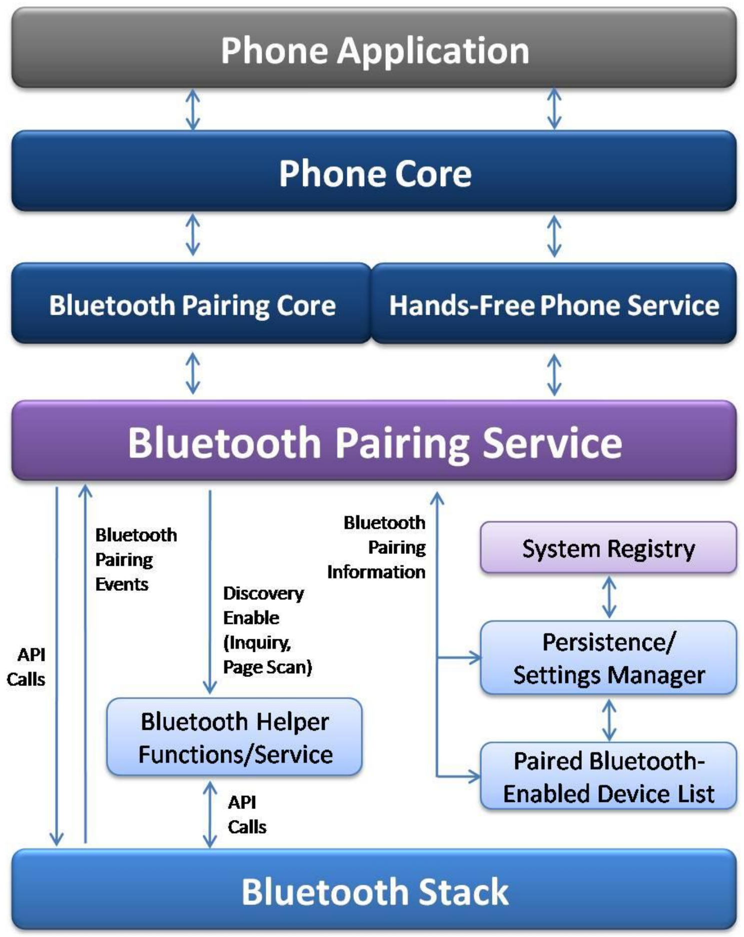

Phone Application
Phone Core
Bluetooth Pairing Core
Hands-Free Phone Service
Bluetooth Pairing Service
API Calls
Bluetooth Pairing Events
Discovery Enable (Inquiry, Page Scan)
Bluetooth Pairing Information
System Registry
Persistence/ Settings Manager
Bluetooth Helper Functions/Service
Paired Bluetooth-Enabled Device List
API Calls
Bluetooth Stack

Figure 13. Bluetooth Pairing Service architecture

[12] Note that the colors used in the figure are for readability only; they have no other significance.

Bluetooth Pairing Core

The Bluetooth Pairing Core is one of the application cores that is used to provide core services to Windows Embedded Automotive 7 applications.

Specific pairing applications may be developed that use the Bluetooth Pairing Core API to take advantage of the capabilities of the underlying Bluetooth Pairing Service to pair to and communicate with Bluetooth-enabled devices. Figure 14 shows how the Bluetooth Pairing Core fits with the other components of the telephony and data communications stack.

The Bluetooth Pairing Core provides the following functionality:

- Support for in process (in-proc) DLL, a provision of straight APIs that wrap the input/output control (IOCTL) calls to the service.
- Conversion of the pairing service “events” to Windows messages.
- Determination of the paired device type as media, phone, or other for use by the media and phone applications.

Hands-Free Phone

The HFP architecture is shown in Figure 14.[13] The phone core fills the gap between a phone application HMI layer and the Windows Embedded Automotive 7 HFP service. The phone core is responsible for phone connection, call handling, call history, and phonebook management.

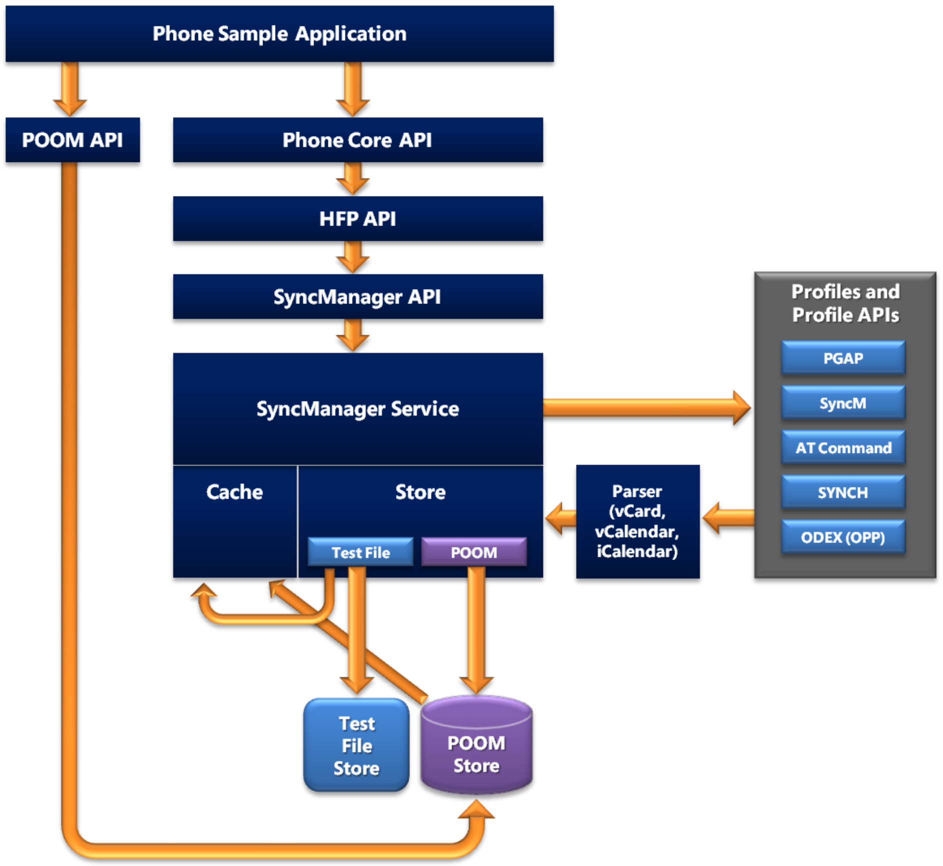


Figure 14. HFP architecture

[13] Note that the colors in the figure are for readability only; they have no other significance.

Phone Core

Phone features:

- Support for hundreds of mobile phones that were tested to help ensure broad market compatibility.
- Twice per year device compatibility updates to ensure that automotive infotainment devices meet changing consumer needs.
- Pocket Outlook Object Model (POOM) library that provides address book storage and contact picture storage for additional data types.
- Support for the new Bluetooth Phone Book Access Profile (PBAP) Message Access Profile (MAP), and new high-quality ringtones.

Phone core updates:

- Improved brought-in device support:
    - Image auto-compression when saving contact pictures to POOM.
    - Sync Manager - support iCal Calendar appointment recursion.
    - Sync Manager - Task download and POOM storage.
- Improved platform support
    - Support for vCard parsing in new character-sets (BIG5; GB18030;SHIFT_JIS;EUC-KR; GB2312/GBK).
    - Phonebook download progress notifications.

Windows Embedded Automotive 7 includes a set of services for HFP, Bluetooth profiles, phone management, data connectivity, and SMS. Windows Embedded Automotive 7 provides support for HFP calls by using the vehicle's audio system, a speech-based interface, and a button-based interface for phone calls made by using an external mobile phone (paired with the Windows Embedded Automotive 7–based device using Bluetooth).
The phone core supports the following functions:

- **In-proc DLL.** The phone core wraps HFP API messages and extends the HFP service for convenient application consumption.
- **Bluetooth phone objects.** The phone core manages phone objects and provides an optimized automatic phone connection feature.
- **Ringtone management.** The phone core provides interfaces for the phone application to support ringtone management and lets the application set up the appropriate ringtone for each connected device (locally provided sound file [such as WAV and WMA], Bluetooth in-band ringtone through Synchronous Connection Oriented [SCO] link, or Bluetooth high-quality ringtone through A2DP).
- **Extensive call control capabilities**. The phone core wraps the HFP service in convenient APIs to enable a phone application to answer incoming calls, to dial a new call, to switch between calls, to hang up calls, and to redial the last number called. Support is also provided for call-waiting information. The phone core downloads and makes the mobile phone's call history available, so that the application can expose the information to the customer as well. The phone core also controls the phone call audio and can direct it to the handset by toggling the privacy mode of the phone.
- **Phone status support.** The phone core exposes the HFP service's ability to get the carrier name, the battery level, the network state, and the signal strength for those phones that expose the information over Bluetooth wireless technology.

- **Broad handset compatibility.** Windows Embedded Automotive 7 regularly ships device compatibility updates, empowering partners to issue updates to their in-vehicle devices that include support for the latest mobile devices available on the market.

Phonebook, Calendar, and Task Management

The phone core uses the Sync Manager to download the phonebook from mobile phones that support functionality through Bluetooth wireless technology. In the default configuration, a phonebook download also triggers a calendar and task download if it is supported by the phone. After an initial phonebook download, data is persisted on the device so that the user has instant access to the information the next time that the phone is connected to the device. Upon connection, the phone core automatically downloads a new phonebook in the background while the customer maintains access to the persisted phonebook. When the automatic download is completed, the persisted phonebook is exchanged for the refreshed version. Contacts can be automatically downloaded by Phone Book Access Profile, by Synchronization Markup Language (SyncML), or by AT commands. The phonebook can also be filled through an OPP/ Object Exchange (OBEX) vCard object that is push initiated by the user.

There are also alternatives. The system developer may choose to use the previous phonebook storage method, which only downloads phonebook data and stores it in a flat text file on flash. The previous phonebook storage method stores only names and phone numbers. However, it stores them in a manner that makes retrieval efficient and provides APIs for maintenance and manipulation of these records.

The system developer may also choose to use the POOM, a Microsoft Component Object Model (COM)-based library that provides programmatic access to Microsoft® Office Outlook® Mobile Personal Information Management (PIM) data items and container objects, for phonebook storage. In its default configuration, the POOM method stores photos, calendar appointments, and other data if it is available from the user's phone. POOM provides an object-oriented framework for creating, modifying, and displaying appointment, task, and contact items—and for manipulating the folders that contain them. The Windows Embedded Automotive 7 version of POOM provides an optimized search algorithm and specialized fields for convenient storage of records for each connected phone.

Both of these methods use the Windows Embedded Automotive 7 standard download logic with respect to protocols and timing when connected to the Bluetooth phone. With the POOM method, the system developer has more options for tuning the system to filter out unwanted data and improve download speeds, depending on the application.

Hands-Free Phone Service

Hands-free phone integration of Bluetooth mobile phones:

- HFP 1.5 and 1.0 support
- Access to the contacts on the connected mobile phone
- Rich call control and multiple-call scenarios
- Compatibility with a wide range of mobile phones

The HFP service can use a user's mobile phone paired over the Bluetooth module for making phone calls. For a paired Bluetooth wireless technology–enabled phone, various features of phone call management (such as digit dialing, dialing by name, conference calling, and call hold) are supported. The HFP service is used for contacts and phonebook management.

The HFP service exposes APIs that are consumed by the phone core and that can be used by a hands-free phone application that is provided by the carmaker or supplier to expose the functionality of these services to the user.

Short Message Service Support

Windows Embedded Automotive 7 supports access to SMS messages that are received by a connected Bluetooth phone and supports sending of SMS messages by a connected Bluetooth phone. SMS messages can be retrieved via AT commands or the Message Access Profile (MAP) service. Windows Embedded Automotive 7 also supports MAP email from any Message Access Service (MAS) instance. Email can be retrieved from the Bluetooth-connected device and then handled by an appropriate application or service.

When AT commands are used, the HFP service interacts with the Bluetooth phone through either the Serial Port Profile (SPP) port or the HFP port. It sets up the event notifications so that it can receive SMS messages and send SMS messages, but the HFP service does not necessarily enumerate through and read the existing SMS messages that are stored on the phone.

Figure 15 shows the Windows Embedded Automotive 7 SMS support architecture.[14]

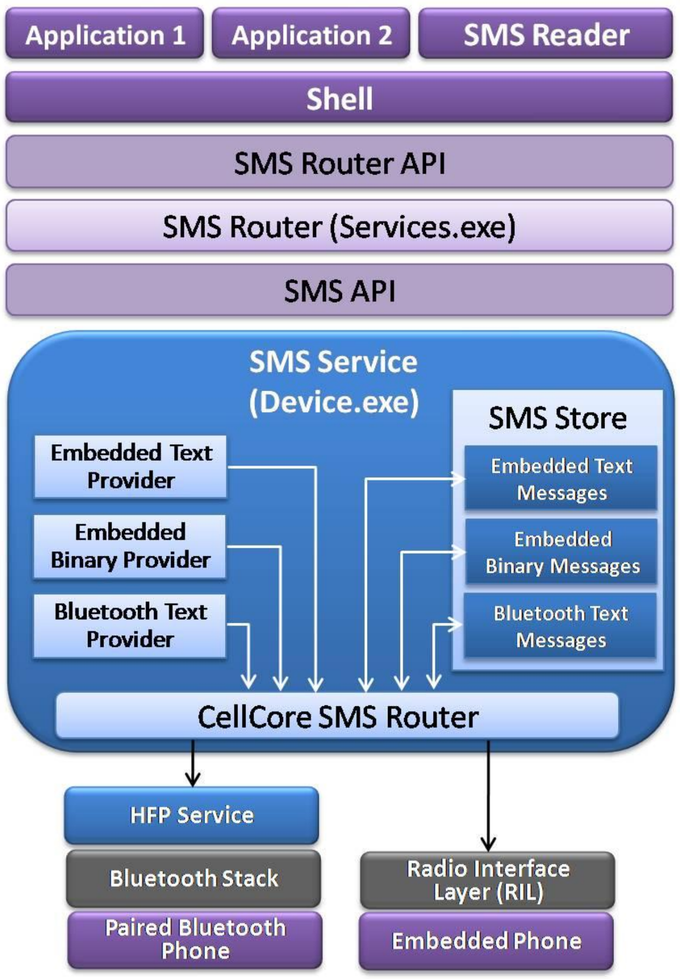


Figure 15. SMS support architecture

[14] Note that the colors used in the figure are for readability only; they have no other significance.

When a message is read from an embedded phone or a Bluetooth phone by the RIL interface, by the HFP service (SMS via AT command), or by the MAP Manager, it is sent by using the CellCore SMS router. The message passes through the providers that are set up in the SMS service, and the message is decoded. Then, applications that have subscribed SMS router notifications are alerted that the message is available. The message is cached in the SMS store for a period of time that is configurable by the developer.

An SMS application that is provided by the carmaker or supplier should use the SMS router API to subscribe to messages, which it can then make available to users. Messages can be filtered and even targeted for specific applications through the SMS router. This message filtering, parsing, and targeting are all configurable by the developer.

Connection Manager

- Manages platform network connections
  - Application access to network connections
  - Multiple available network connection types
- Supported connection service providers (CSPs) include remote access services (RAS), voice, proxy, and Bluetooth voice

The connection manager is the central component for managing connections on the Windows Embedded Automotive 7 platform. The connection manager provides an API to let applications request connections, specify priorities, and close connections after use. It can be configured to manage platform network connections including embedded and Bluetooth-connected phones, wireless connections (WLAN), and Wi-Fi. In the Windows Embedded Automotive 7 default configuration, the connection manager is used to manage voice and data calls on the embedded phone and voice and data calls on the Bluetooth-connected phone.

When an application requests a network connection, the connection manager first retrieves all the possible connections from a set of CSPs. Then, the connection manager configures a set of costs with these routes and ultimately determines the optimal connection based on cost, latency, bandwidth, and other factors. Finally, the connection manager queues the requested connection and uses the CSP to establish the connection at the appropriate time.

Figure 16 provides a schematic of the connection manager.[15] Terms and acronyms are defined in the Glossary.

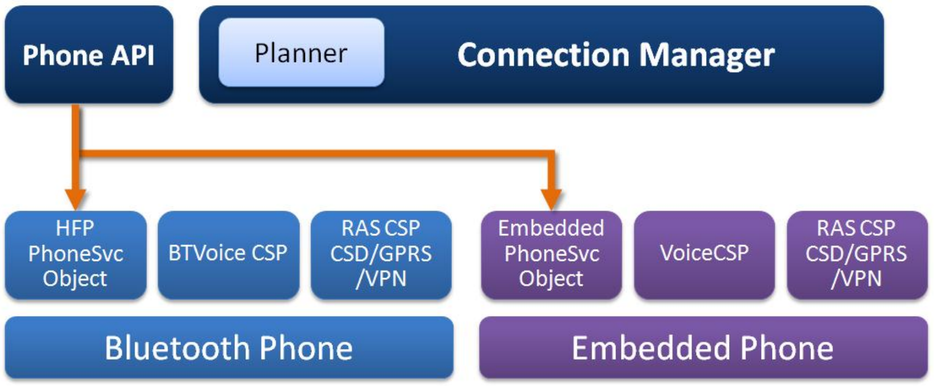


Figure 16. Connection manager

The Windows Embedded Automotive 7 connection manager supports the following CSPs:

- **RAS CSP** provides General Packet Radio Services (GPRS) and dial-up connection support. When used with the Bluetooth phone, RAS CSP relies on the setup menu HMI for configuration of the dial strings.
- **Voice CSP** helps with the coordination of circuit switched data (CSD) and voice calls on an embedded phone. The HFP service calls into the voice CSP for operations that are related to voice calls on an embedded phone.
- **Proxy CSP** allows the insertion of proxy links between defined network destinations. It could be used to create a "virtual" destination that is used by applications logically linked to a "real" destination, which can then be reprovisioned without the need to change the applications.
- **Bluetooth Voice CSP** enables coordination between data and voice calls on a Bluetooth phone. It keeps the connection manager aware of whether the Bluetooth phone is present or not. When a Bluetooth voice call occurs, it creates a pseudo-Bluetooth voice connection so that attempts to create a CSD connection will fail; existing CSD connections are then disconnected, and GPRS connections are suspended.

Entertainment

Windows Embedded Automotive 7 provides entertainment access through the radio and media cores.

[15] Note that the colors in the figure are for readability only; they have no other significance.

Media Core

Media features:

- Supported media classes:
  - Digital Living Network Alliance (DLNA)
  - iPod/iPhone (both one and two wire)
  - Zune
  - MTP
  - Mass storage (USB and SD)
  - A2DP and Audio/Video Remote Control Profile (AVRCP)
- Control over a broader set of devices, including internal hard disks, data CDs, DVDs, and other local storage
- Pluggable interface for music metadata databases
- Ability to tag music for online purchase (iPod users)
- Album art support across device types
- iPod video support
- Enhanced media functionality for in-vehicle entertainment, including media indexing and CD ripping

Media updates:

- Better “brought-in” device support:
  - Numerous iPod and iPhone compatibility improvements
  - DLNA able to be tested with multiple devices, many compatibility improvements
  - Upgraded iAP support (new Coupage library), EAF support (application link protocol)
- Improved platform support, dynamic locale support

The media core is the primary subsystem of the Windows Embedded Automotive 7 platform that media applications use to abstract and manage device-specific interactions, such as indexing and playing media.

By using the media core, developers can work with any of the wired device types, in addition to locally stored media, through a single standard interface and provide the customer with a single experience, regardless of the type of wired media device that is connected. The user experience can be driven by buttons on the display or by voice, depending on the requirements of the supplier and automaker. For more information about device services, see Appendix 5: Media Device Services.

The media core module is responsible for actual media playback, metadata indexing, hardware event handling (for example, power and speech), and maintaining a “now playing” list with history and shuffling ability. All sorting of metadata is done at this level. The media core also supports the use of the A2DP and Audio/Video Remote Control Profile (AVRCP) Bluetooth profiles to enable playback of music wirelessly from phones and other devices that support those Bluetooth profiles. The media core adds media capabilities that include:

- **Zune support**. Windows Embedded Automotive 7 also offers a software add-on package that lets a device fully interact with all the available Zune devices for playback of audio content through a USB connection. The Zune support includes full support for all digital rights management (DRM)-protected content that is purchased through the Zune Marketplace or that is obtained through the Zune Pass subscription service.
- **Media Transfer Protocol (MTP) device support**. MTP refers to the communication protocol that is used to communicate with a variety of media players over the USB

connection. Windows Embedded Automotive 7 supports MTP-based devices from companies such as Sansa, Creative, and iRiver (including the DRM-protected content on those devices) with a software add-on pack. These types of devices are connected by USB for audio playback.

- **iPod support**. Older generation iPod models are supported through two-wire style connections; newer generation models that require a one-wire connection with Apple authentication hardware are also supported. Playback of all audio content (including content that is protected by the FairPlay DRM mechanisms) is supported, as are the iPhone and iPod Touch. Browsing and playback of video from Apple devices is also supported.
- **Mass storage device support**. The media core lets a user bring digital audio that is not DRM protected into the vehicle on mass storage devices, such as USB storage devices and SD cards. Windows Embedded Compact has video playback functionality that can be used outside of the media core to play video from mass storage devices.
- **Supported media formats**. The media core can access, index, and play files that are in WMA, MP3, PCM, WAV, and AAC formats. The media core also supports playlists, including those in Moving Picture Experts Group Audio Layer 3 Uniform Resource Locator (M3U), Advanced Stream Redirector (ASX), and Windows Media Player Playlist (WPL) formats, in addition to the native formats that are supported on iPod and Zune devices. New playlist formats and codecs can be added through the media core extension models.
- **Indexing**. The media core supports index caching in nonvolatile storage, enumeration, and searching by album artist, genre, and title.
- **Browsing media files**. Regardless of the type of the physically connected device, the media core provides full access to the audio media on the device based on the metadata that is included in the digital media files. This helps ensure that the user experience is the same, regardless of the device type that is attached. The user interface can be driven by a folder-based hierarchy or it can be filtered by a variety of available metadata, including track name, album name, artist, and genre. This index is maintained per device by the media core, and the index for each known device can also be persisted (enabling fast access to the same device the next time that it is brought into the vehicle). Device-specific browsing is available during indexing, and category-based browsing is available after index completion.
- **Playback control**. The media core makes it possible to have full playback control over the media files that are available on the device, including play, pause, fast forward, rewind, previous/next track, shuffle, and repeat.
- **Album art**. The media core supports viewing embedded album art and folder .jpg images, in addition to album art from MTP, Zune, and iPod devices.
- **Extensibility.** The media core makes it possible to query custom metadata, to add custom metadata to the index, to use a metadata file parser plug-in, and to direct access to services and devices. Additionally, new codecs can be added, new playlist formats can be supported, and entirely new classes of devices can be added to the media core by using the extensibility model.
- **DLNA.** DLNA defines a standard for moving movies, photos, music, and other media from one device to another. DLNA servers can store media in one location and stream

the media to DLNA compliant players (such as PS3 or Xbox 360®), without any setup or configuration.

Radio Core

Standard programming interface for controlling the radio in the car:

- Support for AM, FM, and HD radio
- Data from Radio Data System (RDS)
- HD radio support for Main Program Service (MPS), Supplemental Program Service (SPS), Station Information Service (SIS), and Program Service Data (PSD)
- Unified tuning API for different types of radios
- Support for multiple tuners, phase diversity tuning, and scanning antenna diversity
- Extensible architecture for additional radio types
- Radio comes on and is operational within 1 second of a cold start on the Freescale iMX35 reference platform

The radio core provides a consistent interface for accessing radio information. Radio core and the radio drivers run in kernel space, making operation fast and efficient. The unified API is also extensible for additional radio types (such as Digital Audio Broadcasting).

Figure 17 shows the radio core architecture.[16]

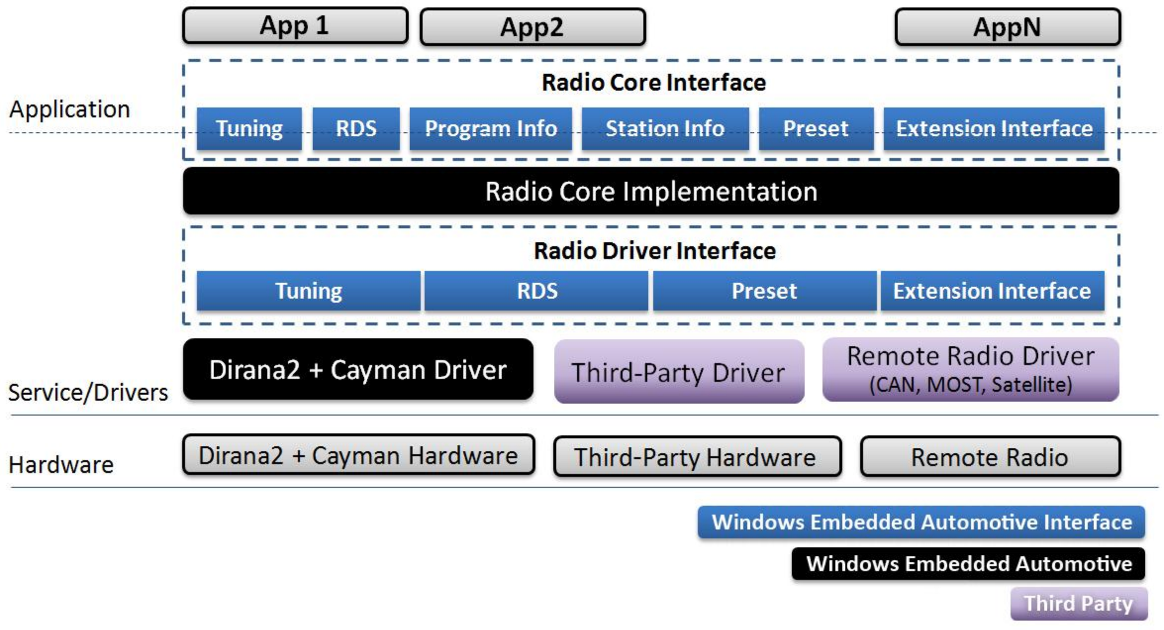


Figure 17. Radio core architecture

The radio core provides tuner support for:

- AM, FM, and HD radio.
- Multiple tuners.

[16] Note that the colors used in the figure are for readability only; they have no other significance.

- Phase diversity tuning.
- Scanning antenna diversity.

Radio information that is available through the Radio Data System (RDS) includes station information, program information, and radio text. For HD radio, there is additional information, including Main Program Service (MPS), Supplemental Program Service (SPS), Station Information Service (SIS), and Program Service Data (PSD). There are auto-fill presets and the ability to filter stations by genre, and the radio can be configured for different countries. Radio tagging is handled by the application, and the application pushes data to a media device through the Media Core API.

Device Management

In-vehicle device management:

- Technology in Windows Embedded Automotive 7 makes a range of update scenarios possible
- Includes the ability to provide new applications in addition to service updates
- Updates can be applied by the owner of the vehicle, saving time and money
- Signed images/updates ensure only authorized content is added to device

Device management (DM) is the updating of the software that resides on the system or of the system configuration itself. While automakers determine the specific device management scenarios, Windows Embedded Automotive 7 supports both the update process itself and an implementation of a USB installer service. Windows Embedded Automotive 7 supports updates through USB, SD, and CD-ROM (other mechanisms can be added as well). As part of the installer service, an OEM can ensure that only authorized (or signed) updates or applications are applied to the device.

DM relies on Image Update, which is based on a Windows Mobile® implementation. Image Update provides:

- Tools to define how the image is packaged.
- Tools to create updates (based on packages).
- Fail-safe installation of updates on platform.

Supported DM scenarios include:

- Installing application software, a complete image, or critical patches and fixes.
- Uninstalling or reinstalling application software.
- Activating or deactivating installed applications.
- Adjusting the user’s configuration.

The provisioning mechanism in Windows Embedded Automotive 7 relies on the download of standard CAB files to the device. There is a standard component available that unpacks, verifies, and installs CAB files; a limited scripting capability makes it possible to modify registry settings and other device-specific settings. The CAB unpacking mechanism is principally independent of the CAB file contents, so this mechanism can also be used to download installable content for other electronic control units that are networked to the Windows Embedded Automotive 7–based device.

The overall DM solution in Windows Embedded Automotive 7 is primarily based on core Windows Embedded Compact DM components, with some enhancements:

- **Single installation API.** A single installation API is used instead of several installer executables. The installation API can handle all types of installation packages, making for effective restart handling because information about the installation status can be exchanged more easily and installation can be paused, stopped, and resumed by using an API instead of by using an executable file. The API supports compressed CAB files for better bandwidth utilization.

  The installer (CESetup) is structured as a shell that invokes installation handlers, which handle specific types of files such as CAB files or CAB Provisioning Format (CPF) files. The installer itself is a DLL that exposes an installation API; it may be hosted in multiple processes.
- **Support for unattended uninstallation/upgrades.** By default, CAB file uninstallations are delayed until key-off. This makes the uninstallation of an application that is in use nonintrusive to the customer.
- **CAB files.** A CAB file can contain one or more system updates. Each application is typically packaged in its own CAB file.

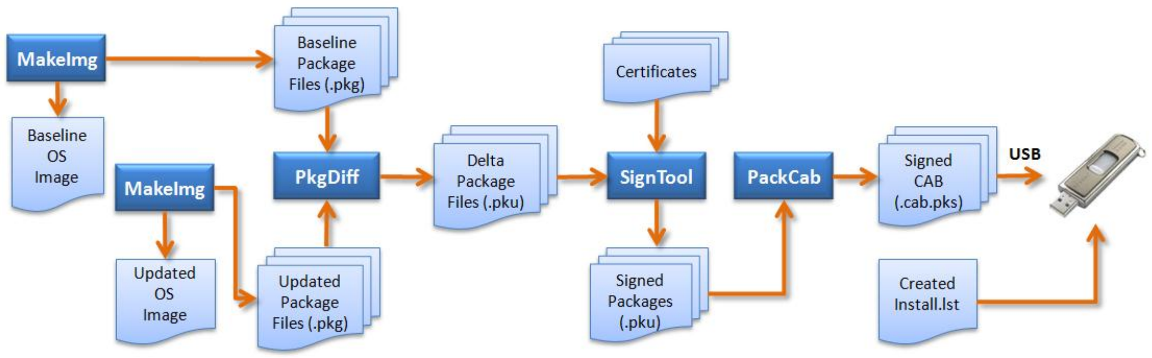


Figure 18. Creating an update

Figure 18 shows the process of creating an update. The workflow steps include:

1. Create package files (.pkg) as part of the MakeImg tool.
2. The PkgDiff tool creates the delta set of packages (.pku).
3. The delta packages are signed and combined in a CAB file (.cab.pks).
4. The CAB files are stored on a USB stick.
5. An install file (.lst) listing the CAB file is created on the USB stick.

Figure 19 shows how DM can be performed from a USB pen drive–based installation.

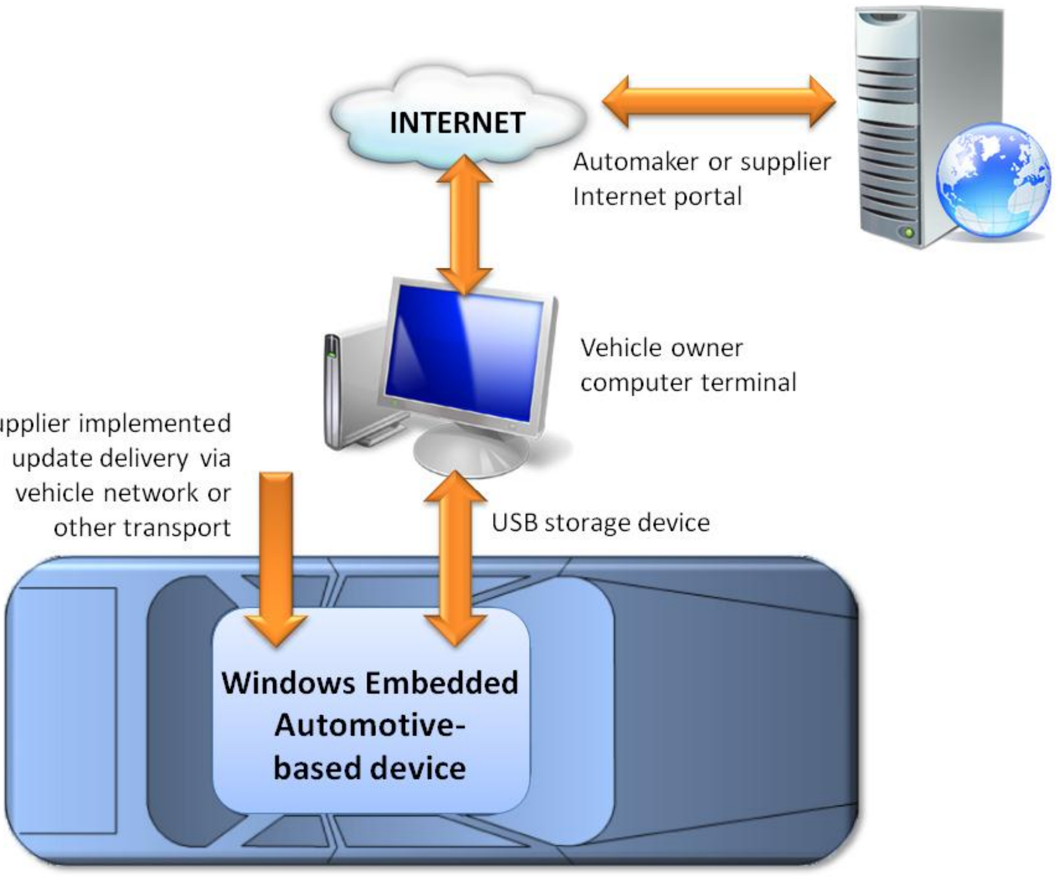

INTERNET
Automaker or supplier Internet portal
Vehicle owner computer terminal
Supplier implemented update delivery via vehicle network or other transport
USB storage device
Windows Embedded Automotive-based device

Figure 19. Device management

The device update process includes three steps:

- Obtaining the actual file (the update image) on the device. The file can be delivered on a USB storage device or by an alternate delivery method that is implemented by the supplier. To find updates, the installer looks for valid files in a USB key (such as a USB pen drive that has the correct CAB files and .lst file for Image Update) and triggers installation if it finds the files.
- Extracting and verifying the update image. Verification includes checking the signature of the update and the checksum.
- Performing the actual update by installing it. To install updates, the installer completes the instructions in the installation script.

For system-update packages, the installer triggers the Image Update mechanism. This causes a restart into the ULDR that ensures that the update can be applied; the update is the applied to the operating system. Update progress and status is carefully tracked across restarts.

Security

The security subsystem helps to ensure that Windows Embedded Automotive 7 does not accept or run executable code that is not trusted on the system. Windows Embedded Automotive 7 is engineered with adherence to the Trustworthy Computing security standards that are enforced at Microsoft. (For more information, see the Trustworthy Computing web page). The most critical security mechanism for Windows Embedded Automotive 7 is its application/operating system trust model. Authenticode® code signing technology is used to help ensure that Windows Embedded Automotive 7 does not install or run code that is not trusted.

It is possible to configure the Windows Embedded Automotive 7 platform so that all updates to the image (whether they are applications or system updates) must be signed by using Authenticode technology. For each different type of update that is possible, a different certificate store enforces this policy. Some examples include:

- **Secure image download.** At the lowest level, certificates in the secure boot downloader (SBOOT) ensure that only valid, signed images can be used to completely re-flash the system.
- **Operating system image updates.** When you apply an image update, the installer verifies the image update signature by using the corresponding public key stored in the policy stores on the device. Then, the installer triggers the Image Update mechanism.
- **Application installation.** Application CABs are signature verified upon download from the USB storage device. Certificates in the installer policy store determine whether the CAB should be accepted or not.
- **Loading operating system image and applications** The loader determines whether applications that are installed to TFAT partitions can run based on certificates in the loader policy store. The installer and loader policies only apply to applications for the TFAT partition. Anything that is loaded from the operating system/kernel partitions is considered implicitly trusted because it is a part of read-only memory (ROM).
- **Other security mechanisms.** Several additional subsystems employ mechanisms to help protect against threats that apply to them. For example:
    - Bluetooth wireless technology connection with an external device is made over a secure link.
    - Application CABs can be signed with a flag that prevents their installation from USB storage devices to limit piracy concerns.
    - Diagnostics Security Access is implemented to allow certain operations to be performed only if the client presents the necessary evidence.
    - For Windows Media DRM, Windows Embedded Automotive 7 can receive streamed DRM content from another device.
    - During Bluetooth connection with an external device over a secure link, the application can access only the phonebook of the currently connected Bluetooth phone. When multiple paired phones are in the vicinity, the user can choose the phone to use through the HMI.

Figure 20 shows the image security certificate stores.[17]

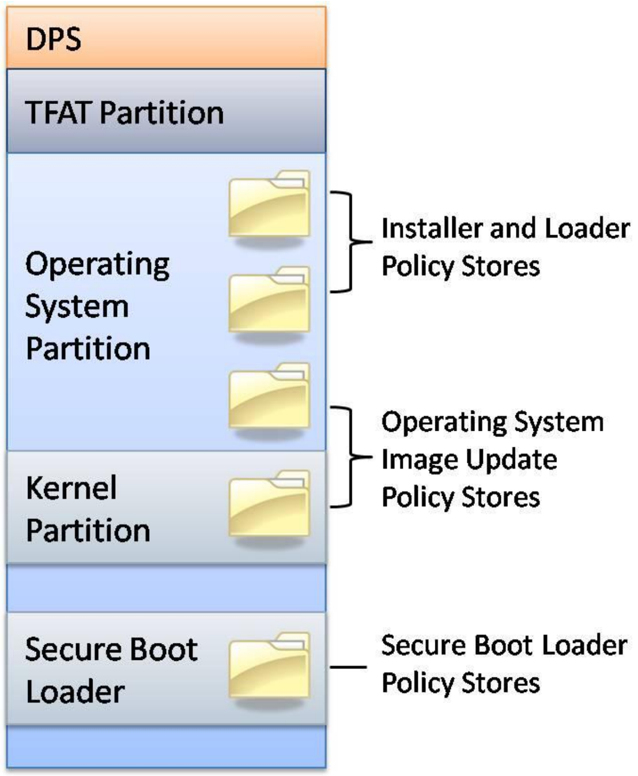

DPS
TFAT Partition
Operating System Partition
Kernel Partition
Secure Boot Loader
Installer and Loader Policy Stores
Operating System Image Update Policy Stores
Secure Boot Loader Policy Stores

Figure 20. Security subsystem

To manage certificates and encryption keys on the device, Windows Embedded Automotive 7 (through Windows Embedded Compact 7) provides the cryptographic API (CryptoAPI).

Cryptographic Keys

The device manufacturer programs cryptographic keys in the DPS of each device and a unique 64-bit product serial number (a unique identifier for the device).

When an application installation package is signed, it can be bound to a specific instance of Windows Embedded Automotive 7 by the inclusion of the product serial number in the Authenticode signature. Application authentication credentials can be created by a combination of the product serial number as the device name and a derivative of the cryptographic key as the password. Note that both the application and the remote service must use a salt to derive the actual password from the unique cryptographic key for each Windows Embedded Automotive 7 device. For example, you can create a salt from a Secure Hash Algorithm (SHA) hash that concatenates the key and a string that corresponds to the service URL.

Image Types

Windows Embedded Automotive 7 provides a set of image types based on security strength. An image with lower-strength security cannot overwrite an image higher-strength security. This scheme lets development proceed without using digital certificates; at the end of development, you can gracefully migrate to a secure image. This also helps to ensure that a development image does not inadvertently overwrite a production image on a device.

[17] Note that the colors used in the figure are for readability only; they have no other significance.

The following image types can be produced:

- **DEVTEST**. This development image type does not enforce signing. You can upgrade to this image type from a device that has a DEVTEST image by using Platform Builder or EBOOT. You can upgrade from a previous PRODTEST image by using SBOOT or JTAG to flash a complete new image
- **PRODUCT**. Once this type of image has been flashed on the device, nothing can be installed unless it is signed with a product certificate. You can upgrade a device with any Windows Embedded Automotive 7 image type to a PRODUCT image type. After you flash a PRODUCT image type to the device or board, you must use SBOOT or JTAG to flash an image type other than a PRODUCT image type.
- **PRODTEST**. Once this type of image has been flashed on the device, nothing can be installed unless it is signed with a product or test certificate. You can upgrade a device that has a DEVTEST or PRODUCT image to a PRODTEST type. If this image type exists on the device or board, it is necessary to use SBOOT and a .sec image file to flash another image type.

Reliability

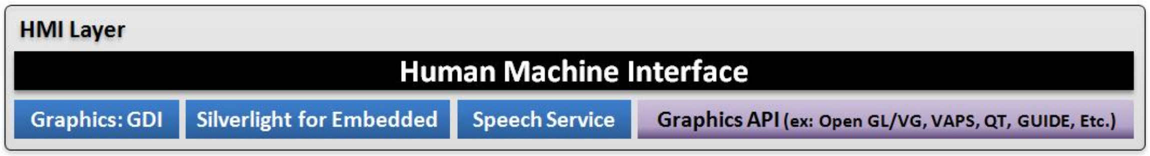
Several subcomponents in Windows Embedded Automotive 7 provide support for system reliability:

- **The hardware watchdog** ensures that the system is reset if it locks up.
- **System health monitoring** is hosted in SysHealth.exe and contains a launch monitor (which helps ensure that the set of applications that are considered to be critical for system functionality start up normally), a memory monitor (which monitors the currently available memory and schedules a deferred or immediate restart if the available memory is below a low or critically low threshold), periodic restart (which is configurable by the supplier), and RTL_ZONE logging (which collects a retail message log for the system).
- **The process monitor** provides a timer-based software watchdog; applications can request a process termination or a system restart if they hang.
- **The reliability service** provides the ability to request system restarts and enforces the logic for the system to shut down completely if recurrent restarts are detected within a short period of time.
- **The flash driver** in the MARPF1 and MARPF2 BSP implements several mechanisms to help ensure a long flash life, including wear leveling, bad block handling, main and spare-area error correcting codes, and protection against flash sector corruption because of unexpected power failure.
- **The backup installer** database becomes the primary database if the system detects any kind of corruption in the active database.

Whenever the system schedules a planned restart, it notifies applications through power management events. Planned restarts occur in situations such as installer-scheduled restarts (with application or system updates) and SysHealth restarts. Whenever the system suspends or restarts in a planned manner, it flushes data to the disk. If the system has to restart because of catastrophic conditions (such as a battery disconnection, a reset from the hardware watchdog, or a critical process failure), it is not possible to notify the applications to save data.

Windows Embedded Automotive has also ported the System Error Handler Architecture from Windows Automotive, which provides a plug-in service to trap system errors, analyze the failure(s), and take corrective action.

Human–Machine Interface



Windows Embedded Automotive 7 does not define or require a specific HMI or HMI technology. OEMs and suppliers can choose from many development tools and run-time libraries—including Silverlight for Windows Embedded—to develop HMIs that best suit their needs.

Windows Embedded Automotive 7 uses two different UI modes: speech interactions and display messages. Applications must ensure that display messages and speech messages are recognized and handled appropriately. To synchronize the speech UI and the display UI, applications handle and respond to events that are sent by the display service, by the speech service, and by the shell.

Human-Machine Interface Layer

The Windows Embedded Automotive 7 application model provides a clear separation of the HMI from the core application logic. The HMI framework, technically part of the Windows application framework, facilitates the separation of the HMI portion of the application from the computational or processing portion. This lets the core of the application be written once; then, the look and feel of the UI can be readily customized. The core applications can be updated without changes to the HMI, and vice versa.

The supplier or automaker can write applications in the HMI layer that take advantage of the application cores and middleware components. Windows Embedded Automotive 7 provides the flexibility to fit into almost any UI paradigm that the automaker may choose—the middleware offers all of the core functionality that lets the supplier present the middleware to the customer in a way that meets the requirements of each individual automaker.

The HMI sample applications that are included in Windows Embedded Automotive 7 include the media player, the phone application, the Bluetooth wireless technology pairing application, and the radio. Note that these samples are included only for the purposes of illustration—they are not intended to be used by automakers as best practice HMIs or to represent an automotive-grade UI.

Speech

- Speech Service
- Speech recognition (SR) and text-to-speech (TTS) engines from
  - Microsoft
  - Third-party, including:
    - Nuance
    - SVOX
    - Loquendo
- SAPI 5.41 (pluggable architecture for developer choice of speech engine)

The speech service that sits on top of the Microsoft Speech API (SAPI) provides the speech-based UI. The speech service lets many applications share the SAPI 5.41-compliant SR and TTS engines (from Microsoft or third parties), and provides common controls that make it easy for applications to have a rich dialog with the user with minimal application work.

The speech engines are interchangeable, which enables inexpensive support for several languages. The Microsoft speech engine languages are packed into individual CAB installation packages, which can be quickly installed to switch languages.

There are several voice recognition and TTS engine combinations included in Windows Embedded Automotive 7 for development purposes only (two sets are from Nuance [Scansoft] for both SR and TTS; another set comes from SVOX; and a final set comes from Loquendo). The supplier and automaker must choose a speech engine vendor, obtain licensing through the vendor for the engines and languages, and then tune the system before deployment.

Speech Service

The speech service provides a set of intuitive, high-level speech controls that are designed to enable rapid speech application development for non-SAPI experts. It also performs system-wide speech-related bookkeeping and management services, such as managing the grammars and arbitrating access to the speech focus.

The speech service supports sharing of speech resources:

- **Prompt Management (TTS engine) for handling dialog**. TTS engines can be configured to use the prompt engine, which can pre-process input text and look up prerecorded prompts in a prompt database, and then combine them with dynamically synthesized prompts to create a more natural-sounding interface.
- **Grammars Management (Automatic Speech Recognition [ASR] engine) for handling dialog**. The grammars (global, local, shared, system, and global exclusive) specify what the system can recognize. It is possible to have one for each application. Token requests are used to select a grammar and then transition to the appropriate local grammar.
- **Alerts and tokens.** Arbitration using alerts (priority-queued prompt requests) and tokens (priority-queued speak/listen items).
- **Recognition Pre-Processing (RPP).** ASR results arrive with confidence scores, and the RPP sets actions for different scores.

Microsoft Speech Engine

The Microsoft Speech Engine includes eight languages (U.S. English, U.K. English, Canadian French, Continental French, Mexican Spanish, Continental Spanish, German, and a preview of Korean), speech recognition, TTS, and enhanced functionality, including:

- SMS by voice.
    - Statistical Language Model–based search for flexible SMS creation/reply without full dictation (Statistical Language Model–based recognition with fuzzy search). See SMS Reply by Voice for more detail.
    - Extensible template list (consult with PST).
- Custom lexicon and text normalization.
    - Custom rules help improve pronunciation and recognition.
    - Bootstrap file of approximately 250 custom media rules included.

- Offline speaker adaptation.
    - Create individual speaker profiles for improved accuracy.
- U.S. English word support in non-English languages.
    - The SR and TTS engines support recognition and synthesis of English words in a second language. The English word pronunciation is shared between SR and TTS.
- CAB installation packages.
    - The Microsoft speech engine languages are also provided as individual CAB installation packages, which can be quickly installed for dynamic speech language addition.

SMS Reply by Voice

A new feature in the Microsoft speech engine is SMS Reply by Voice. In an automotive environment, drivers don't have access to a keyboard to reply to text messages. While drivers always have the option to call the senders to discuss the details, most urgent and common messages only require short and simple replies that could be constructed from speech.

There is not much coverage if the system provides a graphical user interface (GUI) of a small list of SMS replies, but if full SMS dictation is provided, the correction UI is too demanding to be considered for in-car usage. Windows Embedded Automotive therefore implements a middle-ground SMS voice search solution. Rather than supporting completely unconstrained replies or basic list selection, this feature matches spoken input against a series of common SMS templates using statistical language models for recognition and Text-Frequency Inverse-Document-Frequency (TFIDF) algorithms for template retrieval of the most relevant SMS replies. Using this solution, drivers do not need to precisely remember the collection of SMS reply phrases precisely; they can retrieve the exact template because of a fuzzy search post-processing step.

To implement the voice search solution, thousands of real-world SMS replies for all the supported Microsoft speech engine languages are collected and generalized into templates. For example, "see you in 5 minutes" and "see you in 10 minutes" become "see you in <D> minutes," where "<D>" is a dynamic slot. There are four types of dynamic slots supported: Time <T>, Number <D>, Place <P>, and name <N>. These dynamic slots offer flexibility on the final SMS reply.

A statistical language model is built using the collected SMS reply templates, in addition to the dynamic slots. A template search table is also built using the collected SMS reply templates. When the user speaks a reply, the reply is matched against the statistical language model and the resulting output from the speech recognition (SR) engine uses TFIDF algorithms to search against the list of available SMS reply templates in the template search table. A list of the most relevant SMS reply templates is returned to the user, with the dynamic slot content populated. The user can then pick the best-matched reply to send.

Silverlight for Windows Embedded

Brings Desktop and Web capabilities to the device:

- Optimized for Windows Embedded performance
- Includes automotive extensions
- Supports Expression Blend tooling
- Includes 2-D and 3-D hardware acceleration
- Supports C++ code behind

Today's consumers expect a compelling user experience (UX), but creating these user experiences can consume a lot of time and resources.

Typically, HMI design follows these steps:

1. Create the design using Adobe Photoshop and Microsoft® Office PowerPoint®.
2. Define the HMI behavior in Microsoft® Office Excel® worksheets and Word documents.
3. Roll the design and behavior into a rapid prototype (in flash, for example).
4. Build the prototypes into an actual HMI.

Essentially, the work is being done twice: once by the designer and then again by the developer. This system also adds delays and imposes additional risk. After the developer re-creates the UX code, the back-end code must be created as well (or at least enough of the code must be created to get the HMI running on the target). It is only at this point that the HMI can be reviewed. If there is a problem, the HMI goes back to the developer to be corrected; if the UX designer created an interface that is not practical, the cycle may need to start from the beginning.

With Silverlight for Windows Embedded, a new design/develop paradigm is possible:

1. Design the user experience in Expression Blend.
2. Develop the business logic in Visual Studio with Silverlight for Windows Embedded.
3. Run the HMI on the embedded device.

This is a significantly simpler process. Figure 21 compares current HMI design with that possible with Silverlight for Windows Embedded.

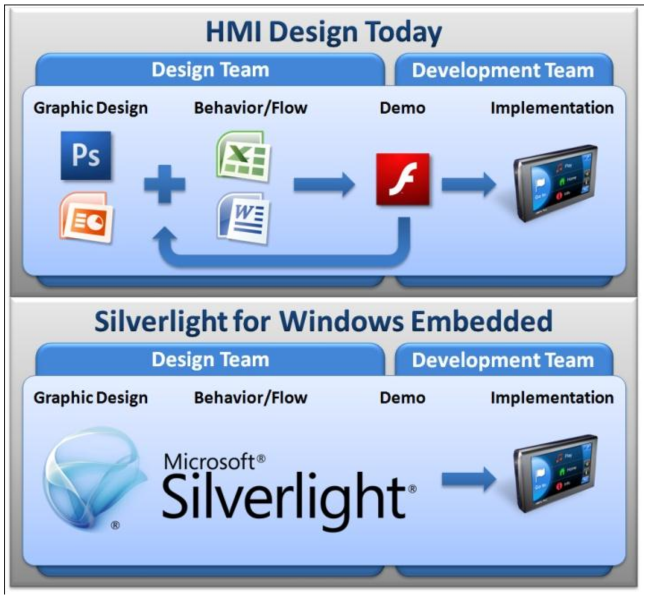


Figure 21. Comparing HMI design today with Silverlight for Windows Embedded

Using a Silverlight for Windows Embedded–based design process eliminates delays and last-minute design changes. The developer can take the UX that the designer created as is. No changes are needed—the UX can be imported directly into the developer's Visual Studio development environment. The review process is instant as well—as soon as an HMI is developed, developers can to see exactly what the HMI will look like on the device before any additional code is created. This saves time and eliminates the risk of having UX issues identified late in the design cycle.

Figure 22 shows the Silverlight for Windows Embedded workflow. Note that acronyms and terms are defined in the Glossary.

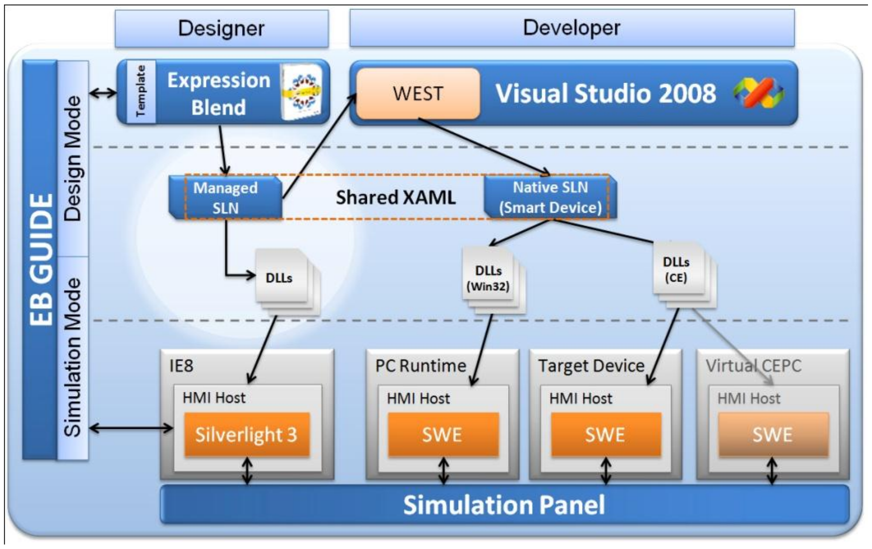


Figure 22. Silverlight for Windows Embedded workflow

Silverlight for Windows Embedded provides a subset of the Silverlight functionality for embedded devices. Unlike other versions of Silverlight, there is no managed API and no browser plug-in.

The version of Silverlight for Windows Embedded included with Windows Embedded Automotive 7 extends the version of Silverlight that is available with Windows Embedded Compact with the following key features:

- **Additional behaviors**, or self-contained, re-usable snippets of interactivity, that can be applied to any Silverlight UI object. These behaviors can support configuration options that are accessible through the property inspector.
- **Automotive button**, a specialized version of the standard UI.
- **OpenVG support**, support for graphics acceleration using the OpenVG capabilities of the graphics card.

Extensible Application Markup Language (XAML), a markup language for declarative application programming, makes it possible for designers and developers to work simultaneously. As with Silverlight on the desktop, the designer creates the visible UI elements in XAML, and the developer creates the separate code-behind files to respond to events, manipulate the elements declared in the XAML, and control the underlying business logic of the application. Unlike Silverlight on the desktop (which uses managed code like C# or Microsoft® Visual Basic®), Silverlight for Windows Embedded uses native code (C++) and does not run in a security sandbox. This means that Silverlight has access to any API and resource on the device. (This also means that desktop Silverlight applications will not run on Windows Embedded CE devices.)

To achieve smooth, responsive animations, devices can take advantage of the acceleration that is commonly provided by the Graphics Processing Units (GPUs) on their hardware. The automotive version of Silverlight for Windows Embedded provides for hardware-accelerated vector drawing through cached composition. An OpenVG1.1-based sample render plug-in is provided, and customers can modify it to use any of vector graphic APIs. (See Figure 23.)

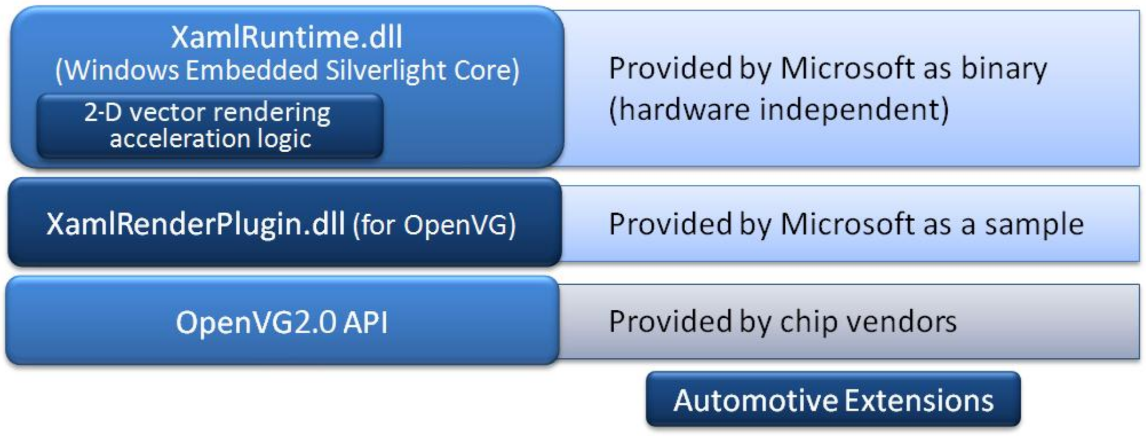

XamlRuntime.dll
(Windows Embedded Silverlight Core)
2-D vector rendering acceleration logic
Provided by Microsoft as binary (hardware independent)
XamlRenderPlugin.dll (for OpenVG)
Provided by Microsoft as a sample
OpenVG2.0 API
Provided by chip vendors
Automotive Extensions

Figure 23. Detail of Windows for Silverlight Embedded

Additionally, there may be out-of-process core applications that have graphical output, such as maps or browsers. The automotive version of Windows Embedded Silverlight has extra elements that use XAML to compose those graphics into an HMI.

Microsoft also provides a default template for an automotive solution, with a reference to the module host binary (shipped with Windows Embedded Automotive 7 or the HMI Toolkit), a bezel proxy and the outline for a media-pseudo application (a managed application that mimics a small subset of the functionality of a real application, used for design purposes and meant to be replaced by a real application when moving to native code). See the Windows Embedded Silverlight Tools section for more information.

For further information about the version Silverlight for Windows Embedded that ships with Windows Embedded Compact, see the Microsoft Silverlight for Windows Embedded web page.

Display

The ability to display text, buttons, and pictures on the display screen complements the speech-based interaction with the user—provided that the display is accomplished with consideration for avoiding driver distraction. The display screen is especially useful for scenarios in which the persistence of visual input has a higher value than auditory input (for example, if a turn announcement is missed, the driver can still look at the display to find out what to do next).

Windows Embedded Automotive 7 can use a display component to work with a remote display, which is typically accessed through the vehicle network. The use of this shared display screen by Windows Embedded Automotive 7 is managed by the display service.

The display service provides mechanisms to help ensure that applications can be written to easily accommodate variations in the display type and layouts. This benefits application developers because it lets them write code that is display agnostic; at the same time, applications can be easily adapted to multiple display types.

Applications communicate directly with the display driver for streaming information, requesting the display capabilities, and more. The display driver also needs to maintain state information with the audio subsystem because of dependencies between the audio source and a particular display control.

Display Driver

The display driver abstracts the display-specific communication that is required on the CAN bus from the higher layers. It also provides character-set mapping.

When the system starts up, the display driver determines what type of vehicle display it needs to adapt to on the basis of a CAN message broadcast by the display head unit. Until that message is received, any requests for display write operations fail. Based on this information, the display driver chooses the appropriate character-set translation map, the supported display layouts, and a display-specific CAN signal assembly.

The display driver implements two sets of IOCTLs—one for determining the supported layouts and another one for writing a message or for clearing the display. The display driver does not distinguish between the different display sections on the display screen. It does not perform any queuing of requests. It also does not perform any verification of whether the application writing to the display has permission to do so.

The display driver relies on the character-set map to perform a mapping of Unicode characters to codes that the display understands. The character codes that are sent to the display are 6-bits long and are chosen from a set of 64 characters, which includes a default character that is used to map any unknown characters. The display driver also truncates the number of lines in a display request to actual lines that the layout supports (it does not truncate characters on a line—that is the responsibility of the display head unit or of the application itself).

Native Display API

The native display API layer uses the services of the display driver after opening a handle to the display driver through a call to CreateFile on “ICD1” (the device name for the display head unit).

The handle is stored for subsequent DeviceIoControl calls. Upon success, it requests the supported display layouts by calling IOCTL_DISPLAY_GET_LAYOUTS and then caches them.

The native display API layer is a pass-through to the display driver. The native display API layer performs focus and parameter validation and packages arguments to send to the display driver. All calls into the native display API are synchronous, and there are no callback or event mechanisms.

This API layer helps ensure that the application that is making a display request has obtained the display focus. Display focus is determined by whether the application has the current Graphics, Windowing, and Events Subsystem (GWES) foreground window. If the requesting application does have display focus, an appropriate IOCTL call is issued to the display driver. Applications can also query supported capabilities of the vehicle display from this API layer.

Note that a Windows Embedded Automotive HMI is typically defined using XAML and rendered through the XAML renderer. But some applications, such as map and video, have their own graphical outputs from their processes. These graphical outputs are composited into a single HMI and presented as one UI screen from a user viewpoint. Typically, this composition is done using a hardware overlay. Automotive version of Silverlight for Windows Embedded provides a feature which blends the graphical outputs of the applications into the Windows Embedded Automotive HMI through the CompositorCore.

The Software Development Environment: Delivering a Windows Embedded Automotive 7–Based Solution

Development tools:

- Automotive System Tools
    - CPU Time Measurement—minimal overhead, report logging for analysis on a PC
    - Memory Measurement Tool—PC tools to create charts/graphs for analyzing memory usage
    - System Logging Framework—Captures logging from different sources (kernel, application, drivers, etc.)
    - System Error Handling Architecture—Framework for error handling, allows custom error handling analyzers
    - TexFAT —Transaction safe version of exFAT
- Visual Studio 2008
    - Platform Builder integrates into Visual Studio 2008
- Expression Blend, EB GUIDE

Application designers can rely on the rich, familiar development environment that is provided by Windows Embedded Automotive 7. For a "future-proof" device, Windows Embedded Automotive 7 uses the secure, standard, and stable update and installation technology from Windows Embedded Compact 7 and Windows Mobile.

Windows Embedded Automotive 7 supports a powerful API set and provides an intuitive development framework that application developers can use. The toolset and framework are familiar to the general development community (which might not have automotive software development experience).

Windows Embedded Automotive 7 comes with a set of developer tools, including the Visual Studio integrated development environment (IDE), Platform Builder (a Visual Studio plug-in), the Expression Blend tool for designers, and various operating system components, such as Internet Explorer, flash, Media Player, and DLNA, the industry-standard protocol for transferring media between devices.

Windows Embedded Automotive 7 provides easy access to various ports and peripherals during the development and testing phase—both for system developers and for application developers. This includes mechanisms in the development hardware and in the software image to enable easy developing, downloading, testing, and debugging of system images and applications. Additionally, Windows Embedded Automotive 7 enables designs to meet requirements from various testing phases, such as unit testing, functional testing, system-integration testing, and in-vehicle integration testing.

Platform Builder for Windows Embedded Compact 7

Platform Builder is a standard IDE for Windows CE–based devices. Platform Builder comes with all the development tools that are necessary to design, create, build, test, and debug a Windows CE–based platform. The IDE provides a single, integrated workspace for a software developer to work on platforms and projects.

Platform Builder ships with Windows Embedded Compact 7 and runs as an add-on to Visual Studio 2008—providing a more consistent development experience for application developers and platform developers.

Windows Embedded Automotive 7 uses Platform Builder primarily in the following scenarios:

- Developing (editing, compiling, and debugging) of native C and C++ code and managed Microsoft® .NET Compact Framework code on the platform.
- Flashing a development board with new software images.
- Monitoring the debug trace output from a running system.
- Using remote tools to measure and monitor various parameters of the device.
- Executing some automated tests of the device using the Windows Embedded test kit.

Platform Builder Catalog

In Platform Builder, a catalog item is the smallest piece of functionality that a software developer can select and add to an operating system design. Managing, modifying, and customizing these items makes up a large part of the operating system design development process. Windows Embedded Automotive 7 provides access to the catalog through command-line tools and configuration files.

The items that the Platform Builder catalog contains range from BSPs and core operating system functionality, such as applications, networking, or security, to transport layers and device drivers. It is possible to add items to the catalog by using catalog item (.pbcxml) files to import the metadata about the items. These items can be items that the developer or a third party has created.

For more information, see the Platform Builder User’s Guide web page.

Platform Customization

Customizing a Windows Embedded Compact 7–based platform can involve adding or removing features from the list of available options in the Platform Builder catalog, adding projects, adding a BSP, adding a device driver, creating an OAL, creating a boot loader, localizing the platform, and exporting an SDK.

In a typical platform development scenario, a software developer first builds a basic operating system image, downloads it to the hardware development platform (see the available platforms in the Hardware Design section earlier in this document), and then refines and debugs the platform.

During the process of refining and debugging the platform, the software developer can customize the platform by adding or removing features and localizing the platform to adapt the operating system for a specific international market or locale. The software developer can add features that are provided with Platform Builder or that are designed by the developer (called user features).

The types of projects that can be created include applications, transport layers, static libraries, and dynamic-link libraries, such as device drivers. A device driver links the operating system and a device, making it possible for the operating system to recognize the device and to expose the device's services to applications. A transport layer is used to communicate between a host computer and a connected device; this is necessary to export a custom SDK.

After refining and debugging the platform, the software developer adapts it for a custom target device. After the platform has been completed, support for the development of additional applications for the platform can be provided by using Platform Builder to create an SDK. Application developers can import the SDK into Visual Studio to create, debug, and run custom applications.

For more information, see the Platform Builder User's Guide web page.

Visual Studio 2008

Visual Studio 2008, the standard development suite for Windows® desktop operating systems development, features an extensive set of tools, configuration samples, and guidelines that can help improve productivity, from the initial design to the final testing and tuning. To improve application performance, Visual Studio 2008 offers code analysis tools, including code metrics that can identify inefficient areas or other problems in code. An integrated build system also features multithreading support for both building and debugging.

Visual Studio 2008 is required for hosting Platform Builder 7.0. Using the Platform Builder mode, developers can create and debug the actual operating system platform on which the HMI applications will run. Developers can use Visual Studio in application mode to develop and debug applications that are running on top of the platform image that is already running on the device.

For more information about Visual Studio 2008, see the white paper "An Overview of Microsoft Visual Studio 2008."

Development Hardware

The development and testing for Windows Embedded Automotive 7 is typically done on prototype hardware rather than through software emulators. See the Board Support Package and Systems section for more information.

Windows Embedded Automotive 7 does provide a hardware reference design—for details, see Appendix 2: Windows Embedded Automotive 7 Base Components and Hardware Reference Design. You can purchase a Windows Embedded Automotive Development Kit from Qualnetics Corporation. This kit includes the Windows Embedded Automotive Reference hardware platform with the Freescale IMX35 processor, digital AM/FM/HD radio, CD player, six-channel audio output, Bluetooth for audio streaming and hands-free phone, DVI video output with touch-screen support, USB and Wi-Fi. For more detailed product information, see the Qualnetics web site.

Additionally, a board is being developed by Renesas. See the Renesas web site for more information.

Platform Development Kit

Windows Embedded Automotive 7 ships as a PDK that installs on top of Windows Embedded Compact 7; it is used in conjunction with Visual Studio 2007 and Platform Builder 7.0 for development of the complete software stack for a specific device.

The Windows Embedded Automotive 7 PDK includes:

- Sample BSPs for the MARP-F2 that is based on the Freescale i.MX35, and for the

Renesas Pilsner, which is based on a dual-core SH4 processor and is SMP enabled. The MACEPC–based hardware platform provides a sample BSP for iA.

- Binaries for the processor architectures Intel iA, ARM, and SH4.
- Binaries for the Windows Embedded Automotive 7 middleware components.
- Documentation.
- The command-line tools necessary to create, modify, and extend a Windows Embedded Automotive 7–based device image.
- Sample applications that run on a prototype board and demonstrate how to use various APIs.

For more information about the samples, see Appendix 5: Windows Embedded Automotive 7 Samples.

HMI Tools

The HMI automotive tools include:

- Windows Embedded Silverlight Tools (WEST).
- HMI simulation panel.
- Expression Blend templates.
- PC Runtime for Silverlight for Windows Embedded.
- Elektrobit guide (EB GUIDE) interface.
- Sample application ("Projekt2").

Windows Embedded Silverlight Tools

Silverlight for Windows Embedded is a native code (C++) UI framework that helps to dramatically improve the UI on devices while letting suppliers reduce the time and cost needed to deliver a differentiated experience through a new designer/developer paradigm.

Designers can quickly and easily design rich, compelling UIs for embedded devices on the Windows Embedded CE platform using familiar tools like Expression Blend (using XAML). At the same time, developers implement the code-behind (code that is joined with markup-defined objects) with Visual Studio and Platform Builder. Windows Embedded Silverlight Tools reviews the XAML for incompatibility, and stubs out function headers in C++.

EB GUIDE Interface

Expression Blend works with EB GUIDE Studio by using an XAML plug-in, so that graphical XAML elements from Expression Blend can simply and easily be controlled by EB GUIDE Studio state charts. The integration of the tools is bidirectional: Expression Blend also has access to system data and events from EB GUIDE Studio.

Support and Maintenance

Windows Embedded Automotive tests hundreds of phone and media players each year to help ensure that the platform works with the latest devices. Each release of the Windows Embedded Automotive platform contains test reports and feature compatibility statements for each supported device. In addition, Windows Embedded Automotive maintains a sustained engineering team that can address issues with Quick Fix Engineering (QFE) service packs.

Partner Solutions Team

In addition to standard product support, Windows Embedded Automotive 7 also provides a Partner Solutions Team, which lets customers secure Windows Embedded Automotive-managed engineering resources that are dedicated to their projects.

The PST will work on any tasks that the customers and Microsoft agree are appropriate, especially project elements that are technically complex and difficult to implement. The PST engineers can help accelerate a project's time-to-market and ensure that the project satisfies all requirements.

Device Lab

The Device Lab began testing mobile phones and media players in 2005; the Device Lab currently tests around 200 devices per year. Detailed test results are regularly updated, and are available to customers (contact inthecar@microsoft.com for access).

The Device Lab selects the devices to be tested based on various criteria, including OEM and customer requests, complaints from users, and sales figures. If an issue is uncovered, the Device Lab informs the device manufacturer and makes a reasonable effort to find a workaround. In 2009, the Device Lab reported 363 fixes/workarounds.

SUMMARY

Windows Embedded Automotive Infotainment Vision

- Build innovative solutions on an award-winning software platform to get to market faster with lower costs.
- OEMs need to meet evolving customer expectations for high fidelity UI experience along with differentiated HMI user experience.
- Customers need devices that seamlessly connect to PCs, Media, online services, and personal content.

Windows Embedded Automotive 7 provides a proven, highly reliable, and extensible software platform and hardware reference design on which automakers can distinguish themselves by building innovative solutions to help drive sales and customer loyalty. It provides automakers, suppliers, and developers with the building blocks that they need to quickly and reliably create a broad range of advanced in-vehicle solutions that meet the growing needs of automotive consumers and set them apart from the rest of the field.

A formal partner program brings new partners onboard and broadens the exposure of existing partners. A wide range of system integrators, software vendors, silicon suppliers, and training partners provide scalable and reliable ways to deliver innovative solutions that are based on Windows Embedded Automotive 7.

Windows Embedded Automotive 7 adds features that are designed to improve the overall car infotainment experience. From effective, software-based acoustic echo cancelling and noise reduction and support for rich speech HMI development, to dedicated support teams for vehicle OEMs and tier-one suppliers, Windows Embedded Automotive 7 offers the best choice for next-generation in-vehicle infotainment solutions.

For the latest information, visit Windows Embedded Automotive on the Web:
www.windowsembedded.com/auto

Related Links

The following web pages provide additional information.

- For more information about Windows Embedded Automotive 7, visit the following web site:
  www.windowsembedded.com/auto

- For more information about Windows Embedded Compact 7, visit the following web sites:
  http://msdn.microsoft.com/en-us/library/bb159115.aspx
  http://www.microsoft.com/windowsembedded/en-us/products/windowsce/default.mspx

- For more information about the Qualnetics development kit, visit the following web site:
  http://www.qualnetics.com/WE-ADK/

APPENDIX 1: WINDOWS EMBEDDED AUTOMOTIVE 7 FEATURES

Windows Embedded Automotive 7, the latest release, includes a host of exciting features, including the following. (See Table 3.) Note that acronyms are defined in the Glossary.

Table 3. Windows Embedded Automotive features.

Hardware and System Specifications

PROCESSOR SUPPORT:

- ARM-based processors, such as Freescale's i.MX35
- Intel iA-based processor (x86 or Atom)
- Renesas SH4-based processors, such as dual-core SH4A

BSPs

In-the box:

- MARP-F2 with Freescale i.MX35
- Renesas Pilsner/Weisse with dual-core SH7786
- Windows CE PC–based hardware platform (CEPC)

BSP, third party

- Intel iA processor support included (CEPC)

BSP test kit, documentation, and tests for self-verification of Windows Embedded Automotive–compatible BSP available in the PDK

DEVELOPMENT HARDWARE

- Qualnetics WE-ADK
- Renesas Weisse with dual-core SH7786

OPERATING SYSTEM:

Based on Windows Embedded Compact 7

Based on the Windows Embedded CE 7.0 kernel, with 2 GB of virtual memory per process and 32 K simultaneous processes

Kernel cache improvements

Heap improvements

Driver optimizations

CPU support:

- ARMv7 support
- NEON, VFP
- x86
- MIPS
- SMP
- SIMD, L2

Browsing:

- Internet Explorer 7, panning/zooming, customizable XAML UI, TAB browsing
- Thumbnail view, auto fit, search integration, flash 10.1 platform
- >512-MB addressable physical RAM

Operating system connectivity:

- Windows Networking Stack (NDIS 6.1/5.0)
- LLTD Technology
- Wi-Fi, WPS Technology
- Connection Manager, works with CellCore and BT
- Zune Wi-Fi setup
- Bluetooth 2.1 protocol support for improved performance
- Profile updates, Simple Pairing
- Improved CellCore/RIL

User experience:

- Silverlight 3.0–based UI development framework (3-D transformations, Pixel/Shader effects)

Expression Blend tools

Multimedia playback:

- WM-DRM 10.02
- CODECs
- WMP
- Customizable XAML UI
- MPEG-4 re-architected pipe-line
- MTP
- Media library
- DLNA 1.5
- Camera API
- HTTP streaming
- Buffer filter
- Server-side playlist

BOOT TIMES:

Boot loaders

- Fast cold boot
- Snapshot boot
- Performance

Performance:

- First drivers: 440 ms
- Radio: 680 ms
- Minimum shell: 1.4 seconds
- Full (sample) applications: ~5 seconds (as tested on the Freescale i.MX35-based
- development platform)

SYSTEMS AND NETWORKING:

- AEC/NS
- Audio management (arbitration, mixer, and more)
- IEEE 1394/IPC
- CAN

Phone

Phone Support:

Bluetooth 2.1 + EDR (with SSP)

Message Access Profile (MAP) 1.0

- Send and receive text messages
- Audio streaming over Bluetooth
- HFP 1.5 and 1.0 support
- Data connection using DUN
- Twice-yearly device compatibility updates

Bluetooth Profiles:

- Generic Object Exchange Profile(GOEP) 1.1
- Object Push Profile (OPP) 1.1
- Serial Port Profile (SPP) 1.1
- Phonebook Access Profile (PBAP)-Phone Book Client Equipment (PCE) 1.0
- Advanced Audio Distribution Profile (A2DP)-SNK 1.2
- Audio/Video Remote Control Profile (AVRCP)-Controller 1.4
- Hands-Free Profile (HFP)-HFP 1.5 (backward compatible to HFP 1.0)
- Dial-Up Networking Profile(DUN)-DT and GW 1.1
- Message Access Profile (MAP) 1.0
- SIM access Profile 1.1
- Device ID profile 1.3
- Human Interface Device Profile 1.0
- Personal Area Network Profile 1.0

Simplified extensibility model for new Bluetooth profiles

CellCore

- Embedded cellular
- 3G
- RIL: AC75i & HC25

Phonebook, Calendar, and Task Management

Phonebook download using PBAP, SyncML, GSM AT, and OBEX

- Flashfile
- POOM

Calendar download from mobile phone: vCal 1.0 / iCal 2.0 supported formats

Messaging

- AT-SMS
- MAP email and SMS

Media, Radio, and Speech

AUDIO FILE TYPES:

- Playlists: WPL, ASX, M3U, Zune, iPod, MTP
- Media files: WMA, MP3, MP4, AAC, WAV
- Codecs: WMA, MP3, AAC, PCM, WAV

Extension interface for additional file types

MEDIA SUPPORT:

Support for DLNA media device integration

Acts as a DMP attached to a DMS or M-DMS with LPCM, MP3, or AAC audio streaming

- Media Core Browse API support
- Device identification for future device interoperable behaviors
- Playlist support

DEVICE CATEGORIES

- iPod/iPhone
- Zune
- MTP
- USB MSD
- Audio CD
- Data CD
- DLNA, DMS, and M-DMS

iPhone/iPod Touch Firmware 3.x support

Multi-application access to media index

- Fourth-generation and newer iPod, iPhone, and iPod Touch support, 1-wire and 2-wire
- Video browsing and playback supported on iPod and iPhone devices and Microsoft Zune (1.0, 2.0, and Zune HD devices)
- Devices certified for Windows Vista® or compatible with Windows® 7 (includes popular players from Creative, SanDisk, Philips, Samsung, Archos, and others)
- Album art across player types
- “Tag to Purchase” support for iPod/iPhone
- CDs, data CDs, and DVDs
- CD ripping to local storage, including plug-in interface for metadata database

CONNECTION MANAGER

- CellCore Cinterion AC75i support, Cinterion formerly Siemens module, SAP and 2.5G Auto-grade solution
- CellCore 3G support, Cinterion HC25; Auto-grade version of this chip available soon

RADIO SUPPORT:

- AM, FM, and HD Radio
- Data from RDS-TMC
- HD Radio support for MPS, SPS, SIS, and PSD
- Multiple tuners, Phase Diversity tuning, and Scanning Antenna diversity
- Extensible architecture for additional radio types

SPEECH SUPPORT:

SAPI 5.41 support creates a pluggable architecture for choice of speech recognition and text-to-speech engines, including:

- Speech service
- Microsoft speech engines, as well as third-party engines, such as:
  - Nuance
  - SVOX
  - Loquendo

Additional Features

SPECIAL PAIRING FEATURES AND SSP:

Secure Simple Pairing Support:

- Tuned through testing against more than 150 shipping SSP devices and at worldwide Bluetooth UPF events
- Numeric Compare Pairing
- Out-of-Band Pairing
- Just Works Pairing
- Supports public and private OBEX phonebook strategies
- Fully tested with devices for complete “legacy” compatibility
- Multi-number handling

PHONEBOOK IMPLEMENTATIONS:

Phonebook SyncManager:

- Utilizes both Pocket Outlook Object Model and a cache mechanism to improve contact download performance
- Contact number and memory budgeting

Software Development Environment

- Silverlight for Windows Embedded
- Visual Studio 2008
- Expressions Blend/Windows Embedded Silverlight Tools
- Watson/device tools
- Device Management

Auto System Tools (AST)

- Advanced Logging
- CPU Time Measurement Tool
- Memory Measurement Tool

Appendix 2: Award Highlights

Windows Embedded Automotive 7 builds on a history of success, as demonstrated by the following awards. (See Table 4.)

Table 4. Industry Awards

2010	• CES Innovation Award 2010 for Ford Work Solution and SYNC connectivity systems powered by Windows Embedded Automotive
2009	• Telematics Detroit 2009 Car Manufacturer Innovation Award for Ford SYNC • Telematics Detroit 2009 Industry Newcomer Award for Ford SYNC • 2009 Automotive PACE Award for Information Technology and Services for Microsoft Auto • Platform Category Award, Automotive Telecommunication Technology, Tokyo
2008	• Bluetooth SIG Best of CES 2008 for Ford SYNC • Global Car Manufacturer Innovation Award, Telematics, Detroit, 2008 • Best New Technology Award, Automobile Journalist Association of Canada
2007	• CNET Editor's Choice Award 9.0/10.0 for JVC's Windows Automotive powered KD-NX5000 • CNET Car Tech Awards - Best Tech for your Beater
2006	• J.D. Powers & Associates—Customer Satisfaction Award for Alpine's Windows Embedded Automotive –powered device • Best Telematics Solution, Detroit Telematics Show for the Windows Mobile for Automotive platform

	• Car of the Year, *MotorTrend* • Excellence in Technology Award, Frost & Sullivan • Best of CES 2006 Award – Car Tech Category for the Pioneer AVIC-Z1 • Cabrio of the Year, 2006, Comite Cabriolet • Innovation 2006 in Design and Engineering Awards, Vehicle Audio Category, Consumer Electronics Association

For more information, see the Windows Embedded Automotive Awards web page.

APPENDIX 3: WINDOWS EMBEDDED AUTOMOTIVE 7 BASE COMPONENTS AND HARDWARE REFERENCE DESIGN

User functionality and features—such as management of device power states and transitions, management of NAND flash, support for data transfer over USB ports, and the time service of the system—rely on some fundamental primitives that are supported by the base system components.

- **Power management.** The power management framework provides the ability to operate the system at a predefined set of power consumption levels and provides a way to define various power-state transitions. There are two main goals for power management: to ensure that adequate power is available to various modules in a specific operating mode and to prevent draining the vehicle battery.
- **Flash driver.** The flash driver provides support for the primitives that are required for reading and writing to NAND flash to various system components (such as file systems and low-level boot loaders). The flash driver is responsible for handling bit errors and bad blocks, for ensuring power-safe operation of the flash memory, and for distributing flash writes evenly to maximize flash life. The file system partitions also reside in flash memory. The flash memory is accessed by using the Flash Abstraction Layer (FAL)/Flash Media Drive (FMD) interface.
- **USB components.** The Windows Embedded Automotive 7 system contains a USB device port and a USB host port. The goal of the USB software components that manage these ports is to provide a robust mechanism for data transfer over the USB ports. The USB device port is used for development-time connectivity and production image programming. The USB host port is used for plugging in mass storage devices (such as a USB storage device that belongs to a service technician or a media storage device that belongs to the user).
- **Device parameter store.** The DPS provides a mechanism for storing device information to persistent memory. The DPS can be accessed by system-level components, such as the boot loader, and by application-level components. The Windows Embedded Automotive 7 DPS implementation is a tool for BSP and application developers to store and access device-specific data.
- **Windows Embedded Automotive 7 IEEE 1394 drivers.** The Windows Embedded Automotive 7 IEEE 1394 driver is a high-performance serial bus interconnect that enables high-speed data transfer. A maximum of 63 nodes can be connected to the network. The driver is self-configuring and uses two types of thin serial cables that do not need terminators or device identifiers.

DEVELOPMENT HARDWARE

Figure 24 shows a schematic for the F2 development platform.[18]

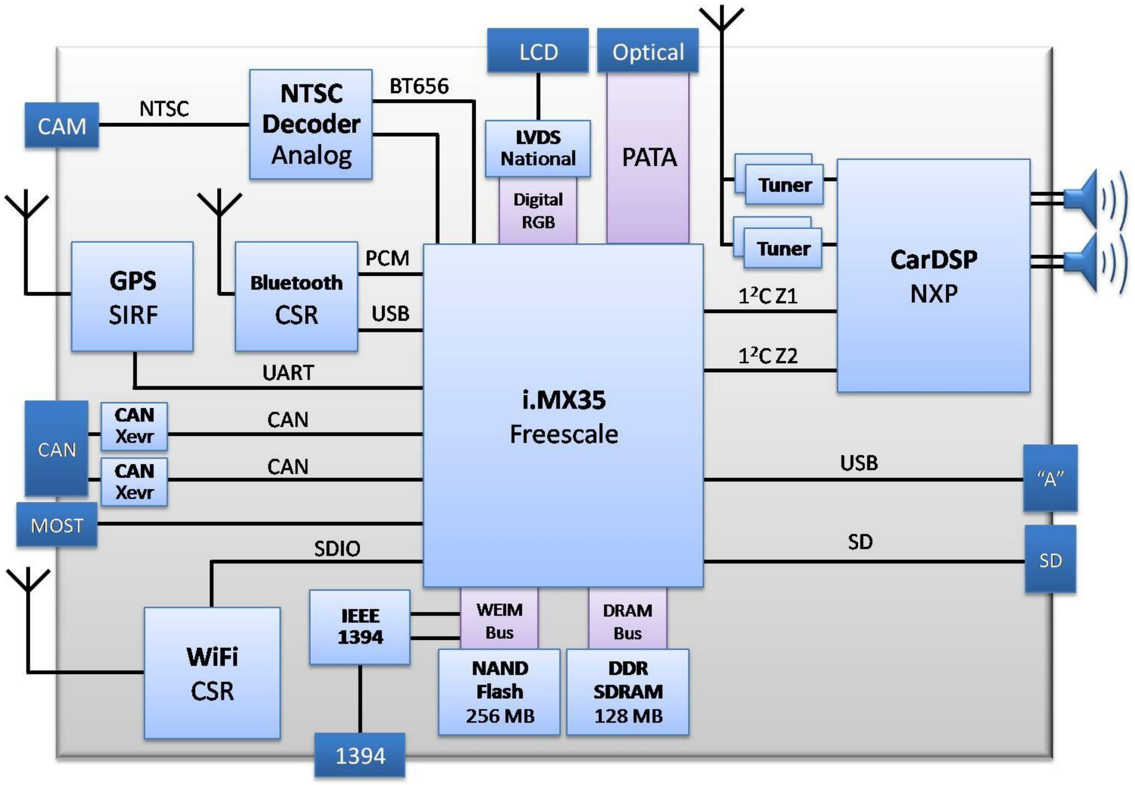


Figure 24. F2 development platform

Table 5 provides details of the Windows Embedded Automotive 7 development hardware.

Table 5. Development hardware

Function	Characteristics
Processor	• Freescale i.MX35, 16-bit/32-bit RISC microprocessor • ARM1136FJ-S core • 90 nm CMOS technology • Multi-Layer 6*5 AHB Smart Speed Crossbar Switch allowing up to five bus transactions in parallel • Separate 16-KB instruction and 16-KB data cache • 128 KB second-level cache • Smart Power Management • Enables simultaneous MPEG4 (SP) encoding and decoding • Support for real-time video decode ◦ MPEG4 SP (simple profile) ◦ H.264 ◦ WMV ◦ RV ◦ MPEG2 ◦ DivX

[18] Note that the colors used in the figure are for readability only; they have no other significance.

	• Video and image data pre-processing/post-processing support in hardware • 400 (533) MHz CPU clock, SDRAM bus at 100 (133) MHz, DDR at 200 (266) MHz • 2-D/3-D graphics support (i.MX31 only) • Camera sensor support • High-speed USB 2.0 interface: • OTG: high speed • Host1: high speed • Host2: full speed • Flexible Audio Interconnect Module allows for programmed flexible connection of the various audio ports (I2S) • Security features • Memory management unit • Security controller including secure RAM and security monitor • Random number generator accelerator • Secure JTAG controller (with optional disabling) • Universal unique identification • Real-time integrity checker • High-assurance boot • Tamper detection • Enhanced direct memory access (DMA) • Timers • Real-time clock • Enhanced periodic interrupt timers • General purpose timer • Watchdog timer • Pulse Width Modulation (PWM) module • I2C • Two Server-Side Includes (SSIs) • Three CSPIs • Five-channel UART • Finite impulse response (FIR) module • GPIO
Memory	• 256-MB NAND flash 8-bit/16-bit • 128-MB DDR SDRAM
CAN	• The CAN controllers can be connected to a vehicle CPU that in turn is connected to the main CPU by a serial connection
Ethernet	• 100 megabits per second (Mbps) Ethernet port for fast connection to development environment or other purposes
USB 2.0 OTG port	• High-speed (480 Mbps) USB 2.0 port available for both device and host connections (On the Go)
USB 2.0 host port	• High-speed (480 Mbps) USB port available for external E-Call module connection and (with an additional USB 2.0 hub) connection to future extension modules • Optional—replaceable by a RS485 UART connection
USB full-speed port	• Full-speed (12 Mbps) USB port for connection to the Bluetooth 2.0 + EDR module
Bluetooth	• Bluetooth 2.1+ compatible • Supports Enhanced Data Rate (EDR) • Connection to processor through USB
Wireless 802.11 module	• Optional 802.11 module can be connected to CPU Secure Digital Input/Output (SDIO) port

APPENDIX 4: BOOT TIMES

In the Windows Embedded Automotive 7 hardware reference design, the software is started from a flash memory instead of from a fixed disk. This greatly reduces the boot time. Figure 25 and Table 6 show the results that Microsoft obtained on Windows Embedded Automotive 7 development hardware.

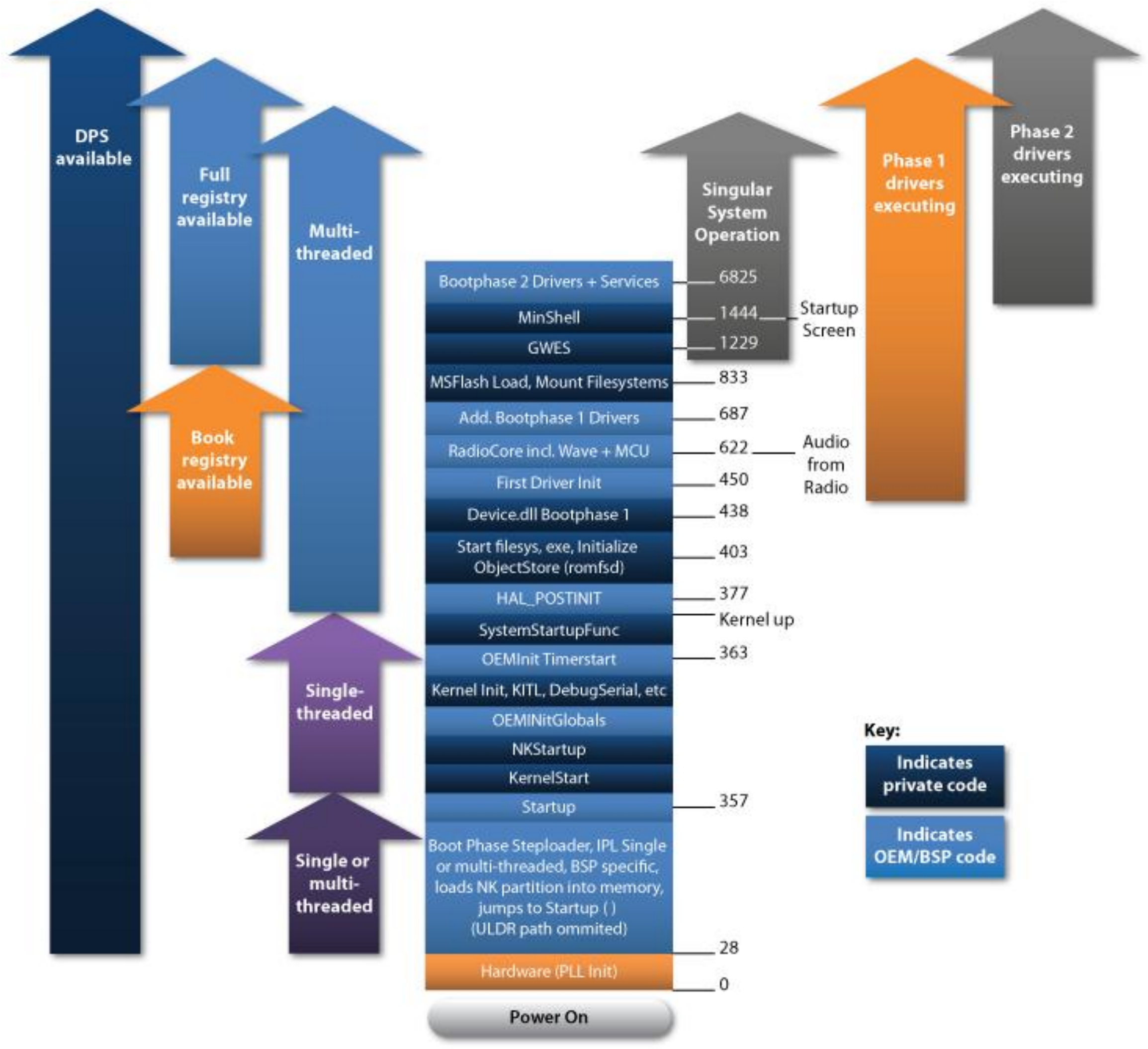


Figure 25. Boot times measured by Microsoft on Windows Embedded Automotive 7 development hardware

Table 6. Boot times on Windows Embedded Automotive 7 reference hardware

Operation	Time (ms)
Startup, initial program loader (IPL), and NK load*	357
OEMInit timestart	363
HAL_POSTINIT	377
Start Filesys.exe, initialize ObjectStore	403
Radio	680
Microsoft flash load, mount file systems	833
Min shell	1444
Full sample applications	~5000

**Interrupts can be activated during that period, but no operating system functionality is available; also, a proprietary IRQ handler is necessary.*

Appendix 5: Media Device Services

Table 7 details the media device services. Acronyms and terms are defined in the Glossary.

Table 7. Media device services

Mass storage devices	Playback	• DirectShow® • Playlists: WPL, ASX, and M3U • Media files: WMA, MP3, MP4, AAC, and WAV • Codecs: WMA, MP3, AAC LC, random selection of files available to play while indexing
	Browsing	• MSD content is immediately accessible via browse interface
	Indexing	• FAT12, FAT16: Standard file access through the file system • FAT32: Proprietary fast access method • Indexing performance optimized for content changes
	Album art	• Support for retrieval of album art once indexing is complete
	Indexing performance	• Varies with the connected device (12–40 ms per song)
	Direct MSD rough performance	• Direct MSD rough performance ○ Highly optimized indexer for FAT32 volumes ○ Indexing twice as fast as MSD
iPod	Platform	• One-wire and two-wire support ○ Digital audio on one wire, analog audio on two wires ○ Not all features are available with two wires (technical limitations: video and tagging) • iAP authentication support ○ Support for 2.0B authentication chips ○ Partner must acquire authentication chips from Apple • Support for new technologies in 3.x firmware ○ iDPS ○ Extended Application Framework (EAF)
	Playback	• Support for FairPlay-protected content ○ Decryption performed on iPod • Play control ○ Support for all common play control commands (play, pause, stop, fast forward, rewind, shuffle, and repeat) ○ Play control commands are executed locally on the iPod

	Browsing	• iPod content is immediately accessible via browse interface
	Indexing	• Uses iPod index instead of local index • Uses iPod playlist
	Album art	• Support for the "now playing" item
	Radio tagging (HD and FM radio)	• Application obtains tagging data • Application uses the Media Core API to push the data to the device
	Indexing performance	• Dynamic indexing is used
MTP/Zune	Playback	• Zune security handshake ○ Customer must acquire a valid production certificate from the Zune team • Windows Media DRM ○ Customer must apply for a Windows Media DRM license and meet system robustness requirements • Play control ○ Support for all common play control commands (play, pause, stop, fast forward, rewind, shuffle, and repeat)
	Browsing	• Audio content is immediately accessible via browse interface prior to index completion
	Indexing	• Support for incremental index ○ Free/used spaces, thumbprint of N songs ○ Tracks added/removed doesn't require full index • Playback while indexing if ○ GetPartialObject is supported ○ Or, indexing is paused if playback is requested
	Album art	• Support for retrieval of album art for any item upon index completion
	Indexing performance	• Varies with the connected device (15–300 ms per song)
A2DP/AVRCP	Command and control	• AVRCP for play control—next/previous/play/pause • Browse functionality with metadata support in AVRCP v1.4
	Playback	• A2DP using SBC or MP3 codec
	Performance	• No indexing is supported

	indexing	
CD	Playback	• Redbook Audio and Compact Disk File System (CDFS) support • Support for all common play control commands (play, pause, stop, fast forward, rewind, shuffle, and repeat)
	Browsing	• Content is accessible via the media core • Common browser interface enumerates track content ○ CDFS can only be navigated through the file system structure until indexing is complete
	Indexing	• Metadata support for CDTEXT • Other metadata sources (for example, Gracenote, WMIs) provide field values through metadata lookup API
	Album art	• Audio CD: no album art • CDFS: same as MSD
	Ripping	• Redbook Audio and CDFS support • Ripped CD content is stored on fixed, local storage ○ Ripped content is accessible via MSD service • Play ripped content while ripping is in progress • Optimized WMA encoder is provided in platform • Encoder can be replaced by a new encoder or an encoder DirectX Media Objects (DMO) and a matching container
Local store	Local store content accessible via MSD source	• Browse by category or by file system • Folder structure is managed by the media core • Support for album art • Can add any root folder (\\my album\music) to the local store
	API to add content to local store from CD or MSD	• Support for metadata editing • Add/remove files • MediaCompactSource API for resolving file system changes

Appendix 6: Windows Embedded Automotive 7 Samples

The Windows Embedded Automotive 7 PDK includes a set of sample applications that run on a prototype board and demonstrate how to use various APIs. These sample applications are described in the following tables.

Every time that you run a full build, all of the sample applications build into libraries and system generate into executables based on the image settings. Each built sample is automatically copied to the release directory on your development workstation.

By using Platform Builder, you can run a sample application on the prototype board after it is copied to the release directory. For samples that run at the command line, run the sample application on the prototype board by typing **S** in the target control window to start the process for that sample. The parameter for the process name is typically the name of the .exe file. You can also include other command-line arguments that are used by the sample application. To find out whether a sample application offers additional command-line arguments and which arguments are available, use the **-h** switch to view help information for the sample.

Media Samples

Table 8 describes media samples.

Table 8. Media samples

Sample	Description	Location
Bluetooth Audio/Video	Demonstrates the use of the Bluetooth Audio/Video (BTAV) Service. The sample registers for BTAV messages, detects paired BTAV devices, and then connects to the first A2DP device in the paired device list. Then, it plays music from the device. When the device stops playing music, it disconnects and closes the device handle.	`<`*`installation_point`*`>\public\autophone\Oak\Samples\A2dp\BtavSvc`
iPod	Demonstrates the use of the iPod Service. The sample waits for a message that indicates that an iPod is attached and then connects to the first available iPod. Then, the sample demonstrates several types of iPod service functionalities, including autoplay, listing the first five playlists on the iPod device, playing a specified song (using a command-line argument), playing the second genre on the iPod device, and skipping to the next song in a playlist.	`<`*`installation_point`*`>\public\automedia\Oak\Samples\Ipod\Sample`
GUI—Media Player	Demonstrates how to use the media core to browse and play media files from portable media devices. Supported media devices include iPod devices, mass storage devices such as USB, and Media Transfer Protocol devices. In addition, if PairingSampleApi.dll	`<`*`installation_point`*`>\public\Autoshell\Oak\Samples\Media\MediaPlayer`

	is available, Bluetooth streaming device support is enabled.	
MTP	Demonstrates the use of the Media Transfer Protocol Service. The sample connects to the first available MTP device, searches all storage, folders, and subfolders on the device for songs, lists the first 30 songs found in WMA format, lists the first playlist it finds, and if a playlist is found, lists the first 30 songs in MP3 format in the playlist.	`<`*installation_point*`>\public\Automedia\Oak\Samples\Mtp\Indexing`

Telephony and Data Communications Samples

Table 9 describes telephony and data communications samples.

Table 9. Telephony and data communications samples

Sample	Description	Location
GUI—Bluetooth Pairing	Demonstrates the use of the Bluetooth Pairing Core to pair with, manage, and view information on Bluetooth devices by setting the Windows Embedded Automotive 7 device in discovery mode to find either the specified Bluetooth-enabled device or the first discovered Bluetooth-enabled device. This application is designed to be consumed by the Media Player sample and the Phone sample as a module to handle pairing and selecting Bluetooth devices to initiate connection.	`<`*installation_point*`>\public\Autoshell\Oak\Samples\Phone\Pairingsample`
GUI—Phone Application	Demonstrates the use of phone and phonebook management to make and receive phone calls and to manage phonebook entries and call history. This sample also demonstrates basic SMS support and Speech Service Reference support.	`<`*installation_point*`>\public\Autoshell\Oak\Samples\Phone\Phonesample`
Bluetooth Pairing Service	Demonstrates the use of the Bluetooth Pairing Service. The sample first puts the Windows Embedded Automotive 7 device in discovery mode to find either the specified Bluetooth-enabled device or the first discovered Bluetooth-enabled device. If a device is discovered, the sample initiates the pairing procedure with the discovered device.	`<`*installation_point*`>\public\Autophone\Oak\Samples\BtPair\Sample`
Hands-Free Phone	Demonstrates the use of the Hands-Free Profile Service. The sample connects to a specified Bluetooth-enabled phone by dialing a phone number. It can also display hands-free profile information, such as the	`<`*installation_point*`>\public\Autophone\Oak\Samples\Phone\Hfp`

	calling number, the battery level, the signal strength, the phone carrier, the service state, and all call list and phonebook entries.	
SMS Reader	Demonstrates the use of the Hands-Free Profile Service Reference and the SMS Router Reference. The sample triggers downloading all SMS messages from a connected Bluetooth phone. All downloaded SMS messages go through the SMS router to the SMS Reader sample. The sample application displays the details of the SMS messages to the output window. When the SMS download is complete, the application closes.	`<`*`installation_point`*`>\public\Autophone\Oak\Samples\Sms\SmsReader`
SMS Send	Demonstrates the use of the Hands-Free Profile Service Reference and the SMS Router Reference. The sample sends an SMS message to the native SMS router.	`<`*`installation_point`*`>\public\Autophone\Oak\Samples\Sms\SmsSend`

Installer Samples

Table 10 describes installer samples.

Table 10. Installer samples

Sample	Description	Location
CE Setup	Demonstrates the use of the CESetup DLL, which enables you to perform custom actions during application installation and uninstallation. This sample application is not a command-line sample. After this sample is built and a platform developer installs it on the prototype board, the sample installs an application and performs a custom action: It creates a file that is called NewFile.txt in the root folder of a plugged-in USB storage device, and then it writes text to the file.	`<`*`installation_point`*`>\public\Autocomp\Oak\Samples\DevMgmt\MsgLog`
Installer	Demonstrates the use of the CESetup APIs and can be used to install a .cab file onto the prototype board from the command-line build window. It provides several example files that are used to create two test .cab files: RebootRequired.cab and NoReboot.cab.	`<`*`installation_point`*`>\public\Autocomp\Oak\Samples\DevMgmt\Installer`
MakeUnload	Creates an output file that contains information for a .cab file uninstallation. This tool is a desktop sample that works with CabWiz. Use the same .inf file with this tool to create a .cab file and an uninstallation file.	`<`*`installation_point`*`>\public\Autocomp\Oak\Samples\DevMgmt\UnloadFile`

HMI Sample

Table 11 describes an HMI sample.

Table 11. HMI sample

Sample	Description	Location
Projekt2	Provides a Silverlight for Windows Embedded UI demonstrating the sorts of interactions that are supported by Silverlight. The UI covers three key areas of an Automotive UI: Phone, Radio, and Media playback. Some/all of these are hooked up to the corresponding Phone, Radio, and Media cores, allowing the user to place and receive phone calls, listen to the radio, and browse and play songs in her media library.	`<installation_point>\public\Autoshell\Oak\Samples\Projekt2\SWE`

Speech	Demonstrates the use of the Speech Service Reference. The sample is modeled after a trivia game and asks the user a series of trivia questions with possible answers of "A," "B," "C," or "D." When one runs this sample application, ensure that the .wav files are built into the image.
	
	Speech is also integrated into the phonesample and mediapalyersample.
	
	By pushing the Push-To-Talk (PTT) button in the phonesample, one can voice dial any contact in the dynamically downloaded phonebook with the "Call <contact_name>" command.
	
	By pushing the Push-To-Talk (PTT) button in the mediaplayersample, you can voice search any media track in her media device with the "Play Track <track_name>" command.
		`<installation_point>\public\Autocomp\oak\samples\Speech/trivianative`
		
		`<installation_point>\public\Autoshell\Oak\Samples\Phone`
		
		`<installation_point>\public\Autoshell\Oak\Samples\Media`

Glossary

A2DP—Advanced Audio Distribution Profile. A2DP defines how high-quality audio (stereo or mono) can be streamed from one device to another over a Bluetooth wireless technology connection.

AAC—Advanced Audio Coding. AAC is a standardized, lossy compression and encoding scheme for digital audio that is designed to be the successor of the MP3 format. AAC generally achieves better sound quality than MP3 at many bit rates.

AEC/NS—Acoustic echo cancellation/noise suppression. AEC/NS is the process of removing noise and echo from a voice communication in order to improve voice quality on a phone call.

Alert—A notification from the system to the user. Alerts are initiated by the system and are associated with a priority. Alerts that have higher priority are played before alerts that have lower priority. One example of an alert is a stock application notifying the user that a stock has reached a certain price.

Alert token—A representation of a system-initiated dialog between the system and the user. Alert tokens are used when the system has to notify the user and receive a response from the user. One example is a navigation application that needs to inform the user that he or she missed a turn and wants to know whether he or she needs new directions. Alert tokens have a priority associated with them.

API—Application programming interface. An API is a source code interface that an operating system or library provides to support requests for services to be made by computer programs.

ARM—Advanced RISC Machine or Acorn RISC Machine. ARM is a 32-bit RISC processor architecture that is widely used in many embedded designs. Power-saving features make ARM CPUs dominant in the mobile electronics market, where low power consumption is critical.

ASX—Advanced Stream Redirector. One of the three Windows Media metafile formats (ASX, WAX, and WVX). The ASX file is a metafile (a file that contains data about another file).

ATCI—AT command interpreter. ATCI enables the Windows Embedded Automotive 7–based device to be used as a modem. ATCI receives AT commands through an input serial port and then parses and interprets them into TAPI, Ex TAPI, or RIL calls. The responses are then converted to AT response codes and returned through the output serial port handle. Typically, the input and output serial ports are the same port. ATCI is used with DUN over Bluetooth.

AT commands—The Hayes command set, also called the AT (for attention) command set, is used by dial-up modems. The command set consists of a series of short strings that combine together to produce complete commands for operations such as dialing, hanging up, and changing the parameters of the connection.

Authenticode—Internet Explorer uses Authenticode technology and its underlying code signing mechanisms to help address security concerns for users who want to download code from the Internet. Authenticode does not guarantee bug-free code. But, Authenticode identifies the publisher of signed software and verifies that the software hasn’t been tampered with before users download software to their PCs.

AVRCP—Audio/Video Remote Control Profile. AVRCP is designed to provide a standard interface to control devices to let a single remote control be in control of all of the audio/visual equipment to which a user has access.

BB—Baseband. The baseband is the physical layer lying on top of the Bluetooth radio layer in the Bluetooth stack. It manages physical channels and links, apart from other services such as error correction, data whitening, hop selection, and Bluetooth security. The baseband protocol is implemented as a Link Controller, which works with the link manager for carrying out link-level routines such as link connection and power control. The baseband also manages asynchronous and synchronous links, handles packets and does paging and inquiry to access and inquire Bluetooth devices in the area. The baseband transceiver applies a time-division duplex (TDD) scheme.

Bluetooth wireless technology—An industrial specification for wireless personal area networks. Bluetooth enables connection and information exchange between devices, such as mobile phones, laptops, personal computers, printers, digital cameras, and video game consoles over a secure, globally unlicensed, short-range radio frequency.

Bluetooth version 2.0 + EDR introduced an Enhanced Data Rate (EDR) of 3.0 megabit/sec (basic signaling rate; the practical data transfer rate is 2.1 megabit/sec).

Boot loader—A program that loads software that is used to start the operating system.

CAB file format—The Windows native compressed archive format. CAB supports compression and digital signing and is used in a variety of Microsoft installation engines, including Setup API, Device Installer, AdvPack (for the installation of ActiveX® components from Internet Explorer) and Windows® Installer.

CAN—Controller Area Network. CAN is a computer network protocol and bus standard that is designed to enable microcontrollers (microprocessors designed for high integration, low power consumption, self-sufficiency, and cost-effectiveness, in contrast to a general-purpose microprocessors) and devices to host computer.

Codec—A device or a program that is capable of encoding and decoding a digital data stream or signal. Windows Embedded Automotive 7 only provides production-licensed decoders for Windows Media Audio and a development license for MP3.

CPF file—CAB Provisioning Format file. A CPF file is a CAB archive that contains only the _setup.xml file.

CDFS—CD-ROM File System driver. The CD file system driver for Windows platforms.

CryptoAPI—Cryptographic Application Programming Interface. CryptoAPI is an API that is included with Windows operating systems and that provides services so that developers can help secure Windows-based applications by using cryptography. It is a set of DLLs that provides an abstraction layer that isolates programmers from the code that is used to encrypt the data.

CSA 1—Core Specification Addendum 1. An addendum added to prevent corruption to the Bluetooth specifications v2.1/v2.0 +EDR.

CSP—Connection Service Provider. A CSP provides connection information to the Connection Manager application, writes provisioning information that is received from the service providers to the registry, and binds connection requests to the NDISUIO (NDIS User-Mode I/O) Driver.

DHCP—Dynamic Host Configuration Protocol. DHCP is a network application protocol that is used by devices (called *DHCP clients*) to obtain configuration information for operation in an IP network.

DirectShow—A multimedia framework/API produced by Microsoft. Software developers can use DirectShow to perform various operations with media files or streams; DirectShow is based on the Windows Component Object Model (COM) framework and provides a common interface for media across many programming languages. It is an extensible, filter-based framework that can render or record media files on demand.

DAB—Digital Audio Broadcasting. DAB is a digital radio technology for broadcasting radio stations, used in several countries, particularly in Europe.

DLL—Dynamic-Link Library. DLLs are implementations of the shared library concept in the Windows and OS/2 operating systems, and they have the file extension DLL, OCX (for libraries containing ActiveX controls), or DRV (for earlier system drivers). DLLs can contain code, data, and resources, in any combination.

DLNA—Digital Living Network Alliance. A standard for moving movies, photos, music, and other media from one device to another. DLNA servers can store media in one location stream the media to DLNA-compliant players without any setup or configuration.

DMA—Direct Memory Access. DMA enables hardware subsystems within the computer to access system memory for reading and writing independently of the CPU. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards, and sound cards. DMA channels can transfer data to and from devices with much less CPU overhead.

DMP—Digital Media Player. Any home theater system or game console that plays audio or video material and/or displays photos.

DMS—Digital Media Server. Software that makes computer files available on the network.

DRAM—Dynamic Random Access Memory. DRAM is a type of random access memory (RAM) that stores each bit of data in a separate capacitor within an integrated circuit. The capacitor charge is refreshed periodically, as opposed to static memory.

DRM—Digital Rights Management. DRM refers to the access control technologies that are used by publishers and copyright holders to limit usage of digital media or devices.

DSP—Digital Signal Processing. DSP is the representation of the signals by a sequence of numbers or symbols and the processing of these signals; it includes audio and speech processing.

DUN—Dial-Up Networking profile. DUN provides a standard to access the Internet and other dial-up services over Bluetooth wireless technology. DUN can be used to access the Internet from a laptop by dialing up wirelessly on a mobile phone.

EB Guide—Elektrobit Guide. EB Guide is a product family of HMI design tools.

ESAI—Enhanced Serial Audio Interface. ESAI is an integration of the multi-capable Enhanced Synchronous Serial Interface (ESSI) common to many of Motorola's DSP56300 processors and the Serial Audio Interface (SAI) designed specifically for consumer and professional audio applications on Motorola's dedicated audio processors.

EDR—Extended Data Rate. Bluetooth version 2.0 + EDR introduced an Enhanced Data Rate (EDR) of 3.0 megabit/sec (basic signaling rate; the practical data transfer rate is 2.1 megabit/sec).

Executable—A file the contents of which are meant to be interpreted as a program by a computer.

Ex TAPI—Extended Telephony API. Ex TAPI provides extended wireless functionality that TAPI does not support, such as network services (operator selection and available operators), call barring functions (locking, unlocking, and passwords), High-Speed Circuit Switched Data, mute functionality, Unstructured Supplementary Service Data, send caller ID, and call waiting control (enable/disable).

FairPlay—A digital rights management (DRM) technology that was created by Apple Inc.

FAT—File Allocation Table. FAT is the primary file system for various operating systems. A TFAT is a Transaction Safe FAT.

FIR—Finite Impulse Response. A FIR filter is a type of a digital filter, an electronic filter that works by performing digital mathematical operations on an intermediate form of a signal (in contrast to older analog filters).

Flash memory—Non-volatile computer memory that can be electrically erased and reprogrammed.

FPGA—Field-Programmable Gate Array. An FPGA is a semiconductor that contains programmable logic components (logic blocks) and programmable interconnects.

Gateway—A computer or a network that enables or controls access to another computer or network.

GDI—Graphics Device Interface. GDI is one of the three core subsystems (along with the kernel and the Windows API for the user interface of Windows). GDI is an interface for representing graphical objects and transmitting them to output devices.

GOEP—Generic Object Exchange Profile. The GOEP provides a basis for other data profiles and is based on OBEX.

GPIO—General Purpose Input/Output. GPIO devices provide a set of I/O ports that can be configured for either input or output.

GPS—Global Positioning System. GPS utilizes at least 24 satellites that transmit precise microwave signals, enabling a GPS receiver to determine its location, speed, direction, and time.

***GSM*—**Global System for Mobile Communications. The most popular standard for mobile phones in the world.

GWES—Graphics, Windowing, and Events Subsystem. Windows Embedded CE combines the Microsoft Win32 API, UI, and GDI libraries into the GWES module. GWES is the interface between the user, your application, and the operating system.

HAL—Hardware Abstraction Layer. A HAL is an abstraction layer, implemented in software, between the physical hardware and the software that runs on a computer. It hides the hardware differences from most of the operating system kernel, so that most of the kernel-mode code does not need to be changed to run on systems that have different hardware.

HCI—Host Controller Interface. An HCI is a basic interface to Bluetooth hardware, responsible for controller management, link establishment, and maintenance.

Head unit—A component of a stereo system, either in a vehicle or in a home cinema system, that provides a unified hardware interface for the various components of an electronic media system.

HFP—Hands-Free Profile. HFP is commonly used to enable automotive hands-free kits to communicate with mobile phones in the car.

HMI—Human-Machine Interface. The HMI is the means with which users can interact with the system, including input and output capabilities.

IMGFS—Image File System. IMGFS is the main Windows Embedded CE image with the TFAT partitions included.

IOCTL—Input/Output Control. A part of the user-to-kernel interface of a conventional operating system, IOCTLs are typically used to enable userspace code to communicate with hardware devices or kernel components.

IPC—Inter-Process Communication. IPC is a set of techniques (message passing, synchronization, shared memory, and remote procedure calls) for exchanging data among multiple threads in one or more processes that are running on one or more networked computers.

IPL—Initial Program Loader. IPL is a mainframe term for the loading of the operating system into the computer's main memory.

JTAG—Joint Test Action Group. JTAG refers to the IEEE 1149.1 standard for test access ports that are used for testing printed circuit boards using boundary scan.

Kernel—The central component of computer operating systems that manages the system resources (communication between hardware and software components).

L2CAP—Logical Link Control and Adaptation Layer Protocol. The L2CAP is layered over the baseband protocol and resides in the data link layer. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

LIN—Local Interconnect Network. LIN is a single-wire serial communications protocol based on the common SCI (UART) byte-word interface. UART interfaces are available as low-cost silicon modules on almost all microcontrollers and they can also be implemented as software equivalents on pure state computers for ASICs.

LMP—Link Management Protocol. LMP manages logical links and logical transports on the baseband and physical layers within a Bluetooth physical channel that exists between two Bluetooth-enabled devices.

Log—The process that logs speech activities and performance, depending on registry settings.

M3U—Moving Picture Experts Group Audio Layer 3 Uniform Resource Locator (also MP3 URL). M3U is a computer file format that stores multimedia playlists.

MAC address—Media Access Control address. A MAC address is a unique identifier that is assigned to most network adapters or NICs by the manufacturer to be used for identification (used in the Media Access Control protocol sublayer).

MAP—Message Access Profile. MAP defines a set of features and procedures to exchange messages between devices.

M-DMP—Mobile Digital Media Player.

M-DMS— Mobile Digital Media Server.

Middleware—Computer software that connects software components or applications. Middleware consists of services that enable multiple processes that are running on one or more computers to interact across a network.

MMU—Memory Management Unit. An MMU is a hardware component that is responsible for handling access to memory requested by the CPU.

MOST—Media Oriented System Transport. MOST is a serial communication system for transmitting audio, video, and control data through fiber-optic cables. This multifunctional, high-performance multimedia network technology is based on synchronous data communication and requires professional software tools and hardware interfaces.

MP3—MPEG-1 Audio Layer 3. MP3 is a digital audio encoding format that is used to create a file to store a single segment of audio so that it can be organized or easily transferred between computers and other devices.

MPEG—Moving Picture Experts Group. MPEG is a working group of the International Organization for Standardization/International Electrotechnical Commission that is charged with the development of video and audio encoding standards. MPEG has standardized compression formats and ancillary standards.

MSD—Mass Storage Device. MSDs are used to store data. MSDs use a set of computing communications protocols defined by the USB Implementers Forum that run on the Universal Serial Bus.

MTP—Media Transfer Protocol. MTP is a set of custom extensions to the Picture Transfer Protocol (PTP) from Microsoft. MTP supports the transfer of music files on digital audio players and movie files on portable media players. MTP is closely related to Windows Media Player.

NAND flash memory—NAND flash memory forms the core of the removable USB interface storage devices (USB storage devices) and most memory card formats.

NDIS—Network Driver Interface Specification. NDIS is an API for network interface card (NICs). NDIS is a Logical Link Control (LLC) that forms the upper sublayer of the Open Systems Interconnect (OSI) data link layer and acts as an interface between the data link layer and the network layer. The lower sublayer is the Media Access Control (MAC) device driver.

NIC—Network Interface Card. A NIC is a hardware component that is designed to let computers communicate over a network. It is both an OSI layer 1 (physical layer) and layer 2 (data link layer) device, because it provides physical access to a networking medium and a low-level addressing system through the use of MAC addresses.

OAL—OEM adaption layer. A layer of code between the Windows CE kernel and the hardware of the target device.

OBEX—Object Exchange. OBEX is a communications protocol that facilitates the exchange of binary objects between devices. Many PDAs use OBEX to exchange business cards, data, and applications.

OPP—Object Push Profile. OPP defines the requirements for the protocols and the procedures to be used by the applications that are involved in the pushing and pulling of data objects between Bluetooth devices.

PBAP—Phone Book Access Profile. PBAP enables the exchange of Phone Book Objects between devices. It can be used between a car kit and a mobile phone to let the car kit display the name of the incoming caller.

PCE—Phone Book Client Equipment role. The PCE role is for the device that retrieves phone-book objects from the phone book server.

PCM—A term for data that is encoded as Linear Pulse Code Modulation (LPCM). LPCM is a method of encoding audio information digitally.

PDA—Personal Digital Assistant. PDAs are handheld (or palmtop) computers. Newer PDAs also have color screens and audio capabilities, which enables them to be used as mobile phones (smartphones), Web browsers, or portable media players.

PDK—Platform Development Kit. Refers to a collection of tools, documentation, and code that lets a platform developer (likely a supplier) create, extend, and customize an image for the device. Platform developers have access to all low-level operating system and device-hardware-specific APIs, in addition to application-level public APIs. Platform developers can write new device drivers and modify an image to support new hardware.

ping—A computer network tool that is used to test whether a particular host is reachable across an IP network; it is also used to test the network interface card of the computer or it used as a latency test.

PLL—Phase Lock Loop. A PLL is a control system that generates a signal that has a fixed relation to the phase of a "reference" signal.

PMP—Portable multimedia player. PMPs are consumer electronics devices that are capable of storing and playing digital media. The data is typically stored on a hard drive, microdrive, or flash memory. Mobile phones are also sometimes referred to as PMPs because of their playback capabilities.

POOM— Pocket Outlook Object Model, a Microsoft Component Object Model (COM)-based library that provides programmatic access to Microsoft® Office Outlook® Mobile Personal Information Management (PIM) data items and container objects, for phonebook storage.

PPP—Point-to-Point Protocol. PPP is a member of the TCP/IP protocol suite and it is designed to facilitate communication between two computers through a serial, network, or infrared interface in a dynamically changing network.

Primitives—Low-level objects or operations from which higher-level, more complex objects and operations can be constructed.

PTT—Push-To-Talk. PTT is the process that enables a user to start a dialog with the system by pressing a button and verbally issuing a command. Any speech process that is executing is paused, and the system switches to listening mode. When the system finishes listening, any process that was paused resumes.

PWM module—Pulse Width Modulation module. The purpose of the PWM module is to enable time-critical waveform operations to be handled by the hardware instead of by software.

RDS—Radio Data System. A communications protocol standard for embedding small amounts of digital information in conventional FM radio broadcasts. The RDS system standardizes several types of information transmitted, including time, station identification, and program information.

Remote layer—The layer that enables speech service to be invoked remotely on the Windows Embedded Automotive 7 device.

RFCOMM—Serial Cable Emulation Protocol. RFCOMM is Bluetooth's adaptation of the TS07.10 protocol. It serves as a base for COM port emulation facilities and derived point-to-point protocols (PPP). Multiplexing and flow control between device and application are also implemented in RFCOMM.

RIL—Radio Interface Layer. RIL provides a uniform radio interface API that can interface with a diverse set of radio modules and standards in the wireless industry. The RIL makes port communication easier by providing a uniform API, because not all radio interfaces that use an AT interface use the same command set.

RPP—Recognition Pre-Process. RPP determines a speech recognition confidence score based on user audio input. The confidence score enables the speech service to determine the best match between user audio input and the current grammar.

RTC—Real-Time Clock. A computer clock, usually in the form of an integrated circuit, that keeps track of the current time.

SAPI—Speech API. SAPI is an API that was developed by Microsoft for speech recognition (converts spoken words to machine-readable input) and text-to-speech (the artificial production of human speech) within Windows-based applications.

SAT—Satellite radio. A satellite radio or subscription radio (SR) is a digital radio signal that is broadcast by a communications satellite, which covers a much wider geographical range than terrestrial radio signals.

SBC—Sub-band Codec. SBC is an audio encoder and decoder used to connect to the Internet, in addition to Bluetooth high-quality audio devices, like headphones or loudspeakers.

SBP2—Serial Bus Protocol 2. SBP2 is a transport protocol that is defined within Serial Bus, IEEE Standard 1394-1995 (also known as FireWire or i.Link), developed by T10.

SCO—Synchronous Connection Oriented protocol. SCO is a type of communications link that is used within the Bluetooth wireless communications standard for small electronic devices.

SCSI—Small Computer System Interface. SCSI is a set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, and electrical and optical interfaces.

SD card—Secure Digital card. An SD card is a flash (nonvolatile) memory card format that was developed by Matsushita, SanDisk, and Toshiba. SD cards are used in portable devices, such as digital cameras, handheld computers, PDAs, mobile phones, and GPS units.

SDIO—Secure Digital Input/Output. Devices that support SDIO (typically PDAs, laptops, or mobile phones) can use small devices that are designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth adapters, modems, Ethernet adapters, or other mass storage media, such as hard drives.

SDP—Service Discovery Profile. SDPs are network protocols that enable automatic detection of devices and services offered by the devices on a computer network. (For example, the Bluetooth SDP is a profile that is used to find out which Bluetooth services are offered by the remote device.)

SDRAM—Synchronous Dynamic Random Access Memory. SDRAM is a dynamic random access memory (DRAM) that has a synchronous interface, meaning that it waits for a clock signal before it responds to control inputs, and is therefore synchronized with the computer's system bus.

SH—SuperH. SH is a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA) that was developed by Hitachi. It is implemented by microcontrollers and microprocessors for embedded systems.

SLN—Visual Studio solution extension.

SMS—Short Message Service. API and Router support the SMS services that are available on mobile phones. SMS is a communications protocol that enables the interchange. The SMS API talks to the SMS router to implement most of the short text messages between mobile telephone devices. SMS technology has facilitated the development and growth of text messaging.

Speech Service Client API—The software abstraction that enables applications to use the speech service to interact with the user by using a speech-based interface.

S/PDIF—Sony Philips Digital Interface. S/PDIF is a data link layer protocol and a set of physical layer specifications for carrying digital audio signals between devices and components over either optical or electrical cable.

SPP—Serial Port Profile. The SPP emulates a serial cable to provide an easily implemented wireless replacement for existing RS-232-based serial communications applications, such as familiar control signals. It provides the basis for other profiles, such as DUN, Headset Profile (HSP), and AVRCP profiles.

SR engine—Speech recognition engine. The SR engine is the process that handles speech recognition. The speech recognition engine must be compliant with SAPI 5.41. The engine can run as an in-proc recognizer or an out-proc recognizer.

SSI—Server-Side Includes. SSI is a simple server-side scripting language that is used for the web primarily for dynamically including the contents of one file into another file that is served by a Web server.

SWE—Silverlight for Windows Embedded. Silverlight for Windows Embedded is a native (C++ based) user interface (UI) development framework for Windows Embedded CE powered devices and is based on Microsoft Silverlight 2.

SyncML—Synchronization Markup Language. SyncML is a platform-independent information synchronization standard.

SYSGEN—System Generation. The installation of a new or revised operating system. It includes selecting the appropriate utility programs and identifying the peripheral devices and storage capacities of the system that the operating system will be controlling.

System grammar—The grammar that is associated with the system. The system grammar is always active when the speech system is in listening mode, even if no application has the token. The system grammar recognizes the commands “Help,” “Repeat,” and “Cancel.”

TAPI—Telephony API. TAPI abstracts call-control functionality to allow different, and seemingly incompatible, communication protocols to expose a common interface to applications through its support of telephony service providers (TSPs). It exposes telephony operations, such as call waiting, call forwarding, conference calling, and call holding.

TDI—Transport Driver Interface. An interface in the Windows Embedded CE operating system that serves as an adaptation layer to Winsock-based user APIs. It isolates the highly asynchronous callback-based architecture of the stack that is presenting a Windows Sockets Specification 1.1 interface.

TFAT—Transaction-safe FAT. A TFAT file system is a file system that is designed specifically to provide transaction safety for data that is stored on a disk. TFAT requires a hardware-specific driver that is designed for the type of media on which the TFAT volume resides.

TFTP—Trivial File Transfer Protocol. TFTP is a file transfer protocol that can be implemented in a very small amount of memory.

TLB—Translation Lookaside Buffer. A TLB is a CPU cache that memory management hardware uses to improve virtual address translation speed.

TMC—Traffic Message Channel. TMC is digitally encoded traffic and travel information (typically encoded using FM-RDS).

Token—A representation of a user-initiated dialog between the user and the system. There is only one token available in the system, and an application must request the token to start the dialog with the user. If an application requests the token while another application has the token, the new application is granted the token, and the old application is notified that it lost the token through the SSN_LOSTTOKEN event, unless the token’s style is set to system modal. In this case, the token is not released until the original application releases it.

TSP—Telephony Service Provider. A TSP is included in Windows Embedded CE (the Unimodem service provider) and supports AT-command-based modems.

TTS engine—Text-To-Speech engine. TTS is the process that converts a text phrase into a generated voice audio. The TTS engine must be complaint with SAPI 5.41.

UART—Universal Asynchronous Receiver/Transmitter. UART is computer hardware component (an individual or a part of an integrated circuit) that translates data between parallel and serial forms. UARTs are now commonly included in microcontrollers.

UPL—Update Loader.

USB—Universal Serial Bus. USB is a serial bus standard for interface devices that is designed to let peripherals be connected by using a single standardized interface socket, improving plug-and-play capabilities because devices can be connected and disconnected without restarting the computer (called hot swapping).

vCard—A file format standard for electronic business cards. vCards are frequently attached to email messages but they can also be exchanged on the World Wide web. vCards can contain name and address information, telephone numbers, URLs, logos, photographs, and audio clips.

VPN—Virtual Private Network. A VPN is a computer network in which some of the links between nodes are carried by open connections or by virtual circuits in some larger networks (such as the Internet), as opposed to running across a single private network.

VSP—Virtual Serial Port. A VSP is a redirector without network software support that is usually used to create a pair of back-to-back virtual COM ports on the same computer.

WAV—Waveform Audio Format. WAV is a Microsoft and IBM audio file format standard for storing an audio bitstream on a computer. It is a variation of the RIFF (Resource Interchange File Format, a generic meta-format for storing data in tagged chunks) bitstream format method for storing data in “chunks” and it is the main format used on Windows for raw and typically uncompressed audio. The default bitstream encoding is the Microsoft Pulse Code Modulation (LPCM) format.

WEST—Windows Embedded Silverlight Tools. WEST let you take an Expression Blend 3 project and create a C++ project in Visual Studio with many of the required initialization functions and event handlers automatically stubbed out.

Wi-Fi—Wireless Fidelity. Wi-Fi is a wireless technology that promotes standards for the interoperability of wireless local area network products based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards. Common applications for Wi-Fi include Internet and voice over IP (VoIP) phone access, gaming, and network connectivity for consumer electronics.

Win32 API—Windows API. WinAPI is the core set of application programming interfaces (APIs) that are available in the Windows operating systems—Win32 is the 32-bit API. The API consists of functions implemented in system DLLs (Kernel32.dll, User32.dll, and Gdi32.dll).

Windows Embedded CE—A Windows operating system that was developed for embedded systems. Windows CE has a distinctly different kernel, not a trimmed-down version of desktop Windows (it should not be confused with Windows® XP Embedded, which is based on the Microsoft® Windows NT® Server operating system). Windows Embedded Compact 7 is supported on Intel x86 (and compatible processors), MIPS, ARM, and Hitachi SuperH processors. Windows Embedded Automotive 7 ships with support for only two ARM-based processors.

Winsock—Windows Sockets API. A technical specification that defines how Windows network software should access network services, especially TCP/IP.

WMA—Windows Media Audio. WMA is an audio data compression technology. WMA can refer to the audio file format or its audio codecs.

WPL—Windows Media Player Playlist. WPL is a computer file format that stores multimedia playlists.