A Technical Companion to Microsoft Auto 4.0

Proven technology adapted for the automotive industry

Published: September 2009

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

Abstract

Microsoft, working in partnership with the automotive industry, has developed Microsoft® Auto 4.0, an industry-leading platform for communication, entertainment, navigation, and connected services.

Microsoft Auto 4.0, the newest generation of embedded operating systems from Microsoft, 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 includes a large set of integrated, tested, and flexible middleware components, as well as the hundreds of components that are available with Windows® Embedded CE 6.0 R2. These components enable Microsoft Auto–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

Today's customers want to stay connected to information and entertainment sources while they travel in their vehicles. They want their mobile devices—such as mobile phones, portable navigation devices, and portable music players—seamlessly integrated into their vehicles. They want access to the connected services that they have come to rely on—such as navigation and live traffic reports—readily available. Customers expect high-quality audio and an array of information and entertainment options, from immediate access to information services (such as help in finding a nearby restaurant) to the capability to automatically capture and replay a wide variety of music.

Microsoft is at the forefront of this trend, partnering with the automotive, mobile, and consumer electronics industries to provide technologies that can help these industries quickly bring feature-rich, innovative, and cost-effective solutions to the market. The Microsoft® Auto 4.0 platform is a development platform that includes an extensive software framework and a hardware reference design that are specifically engineered for the automotive industry. Microsoft Auto 4.0 empowers application designers to develop a variety of integrated solutions that can help customers get to their destination while keeping them connected to the people, information, services, and digital entertainment that matter to them.

This white paper, "A Technical Companion to Microsoft Auto 4.0," introduces Microsoft Auto 4.0 and the many benefits that it provides to automakers and tier 1 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 Microsoft Auto 4.0, discusses the challenges that automakers and suppliers face in bringing integrated solutions to market. It also explains how the Microsoft Auto 4.0 platform helps 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 Microsoft Auto 4.0, provides a close look at many of the features of Microsoft Auto 4.0.
- The third section, Delivering a Microsoft Auto 4.0–Based Solution, describes the tools that are included in Microsoft Auto 4.0 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 Microsoft Auto 4.0 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 Auto, visit the following Web site:
http://www.microsoft.com/auto/ma.mspx

The Business Case for Microsoft Auto 4.0

The market is ripe for a new generation of in-vehicle systems. Consumers increasingly 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 e-mail, Web browsing, and calendaring), and communication (including conferencing and calling)—all seamlessly integrated, as if the vehicle was 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. Research from the U.S. Department of Transportation and the National Highway Traffic and Safety Administration estimates that Americans spend more than 500 million commuter hours per week in their vehicles and that 73 percent of mobile phone users talk on their phones while they are driving. Consumers report that they would frequently use services such as location-based search and digital audio entertainment and that these services would substantially affect their willingness to switch car choice or mobile operators.[1]

Hands-free driving legislation has been adopted in a number of countries and U.S. states. This has lead to significant pressure on OEMs to create cost-effective solutions for the provisioning of hands-free devices and has led to a rise in consumer demand for Bluetooth® wireless technology. The European Union is considering making eCall a mandatory service in all new cars, potentially propelling European telematics to the forefront and driving the inclusion of speech and Bluetooth technology in in-vehicle systems.

The market for “green” solutions is also growing, as organizations continue to explore new ways to address the increasing market interest in preserving the environment and to comply with environmental legislation. Environmental efforts at Microsoft are focused on using technology to help solve environmental challenges; Microsoft sees it as a key responsibility to help enable OEMs to build their environmentally friendly concepts on the Microsoft Auto platform. For example, with the Microsoft Auto–powered Blue&Me, Fiat Auto Group was able to build the eco:Drive solution, which helps consumers monitor their cars' behavior and therefore become more environmentally considerate.

[1] ABI Research 2008

Introducing Microsoft Auto

Microsoft Auto aims to bring the vehicle into today's digital lifestyle—Microsoft Auto delivers entertainment and communications features, and it provides the opportunity to drive new revenue through remote services. Microsoft Auto also lets automakers and their partners move closer to their customers, enabling them to accurately match mobility products and services to those who demand them. (See Figure 1.)

Figure 1. Microsoft Auto: Bringing the digital lifestyle into the car

Microsoft Auto is based on a vision to enrich the in-vehicle experience for drivers and passengers by bringing the mass market an industry-leading platform and by enabling services for communication, entertainment, navigation, and information.

- **An enriched in-vehicle experience for drivers and passengers.** The technology in Microsoft Auto is designed to enable easy-to-use, appropriate experiences for the front and rear vehicle seats, facing drivers and passengers alike. To stay relevant, the platform provides update capabilities; these help ensure state-of-the-art support for modern consumer electronic devices through device updates and service content.
- **An industry leading platform and services.** Microsoft Auto reduces the cost and time-to-market of telematics and infotainment solution development by providing a platform with integrated communication and entertainment features and by providing a connectivity platform for navigation and information service features. It provides the tools for creating unique solutions that help automakers and suppliers set themselves apart from the competition.

- **Integrated communication, entertainment, navigation, and information.** The integrated components in Microsoft Auto help automakers and tier 1 suppliers connect drivers with a wide range of devices, services, and technology, including hands-free Bluetooth phone communication, media and mobile device integration, rich content that is delivered through connected services and high-fidelity digital entertainment. These technologies are not delivered in silos, separated from each other as individual components; instead, they are integrated to provide a seamless experience for customers. Microsoft Auto does not define or require a specific operation concept or human–machine interface (HMI). Therefore, the infotainment system can match the look-and-feel of the automaker's or tier 1 supplier's target customer.
- **For the mass market.** The Microsoft Auto platform lets suppliers reduce design and engineering costs so that they can create devices at a lower per-device cost. Microsoft Auto provides the ability to 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.

Microsoft Auto is committed to providing:

- *Integrated features*—Including high-quality state-of-the-art communication, navigation, entertainment, and service features at low cost.
- *A robust platform for infotainment*—Providing a platform to reduce risk, shorten time to market, and drive down the overall cost for suppliers and automakers.
- *Rich tools*—Using industry-leading tools for application and HMI development by engineers and designers.
- *Relevance*—Upgradable to evolve with consumer trends and technology, which improves customer satisfaction and loyalty.
- *Services*—Enabling connectivity to Web-based services content to deliver real-time information, entertainment, and security for drivers.

Benefits of Microsoft Auto

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 application programming interfaces [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 2.

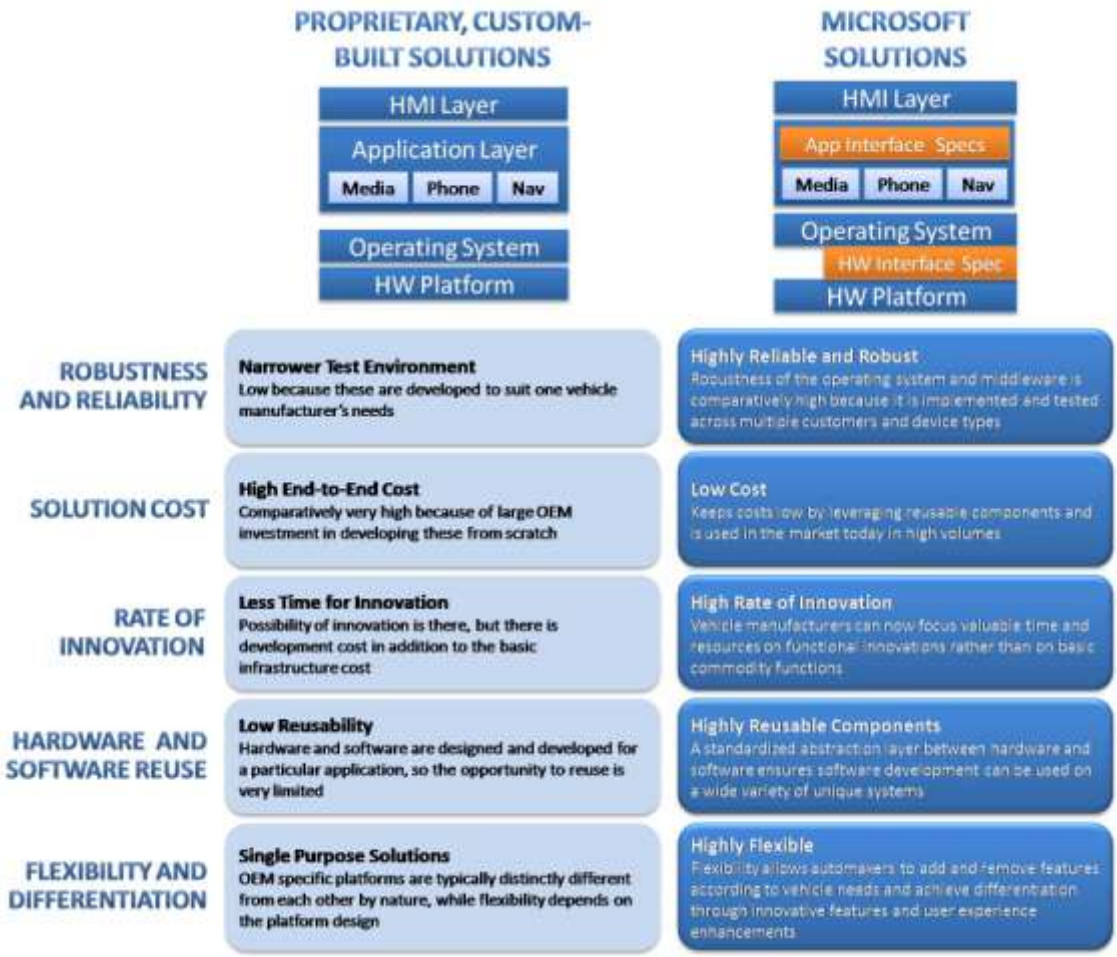


Figure 2. Benefits of Microsoft Auto

According to the 2008 Best Global Brands List by Interbrand, Microsoft is the third strongest brand worldwide, with a brand value of over 59 billion U.S. dollars. This is a unique asset that no other player in the automotive supplier environment is able to offer.

Microsoft Auto Service Center and Module

Microsoft Auto 4.0 provides the ideal platform for creating compelling, connected experiences for in-vehicle consumption. Solutions that satisfy customer demand for connected services can be prototyped with the production-ready connectivity and networking platform technology that is provided by the Microsoft Auto Service Center and the Microsoft Auto Service Module. These use many Microsoft assets (including MSN®, TellMe, MSN Direct, and Bing™ maps) as well as Microsoft partners.

The Microsoft Auto Service Module provides an environment for prototyping connected service-based customer scenarios in a very rich and stable environment. The services that are available for development purposes include:

- **Local business listings, weather, and gas prices**. This information, which is licensed by Microsoft for redistribution, is provided to customers through a Microsoft-hosted infrastructure. This use of an existing content and delivery infrastructure affords

economies of scale across aggregated service delivery and benefits services-based business models.

- **A solution framework for aggregating location-based services**. The solution framework is composed of adapters: a business adapter provides the infrastructure for reporting and attaching business models, a license adapter manages device and partner access to services, and a protocol adapter optimizes a delivery-based integration model. This streamlines the introduction of new services to customers.
- **Microsoft Auto client components**. Microsoft Auto Services exposes the Win32 API to developers, which reduces the cost of service integration. Microsoft Auto Services also supports existing in-vehicle HMI models to ease adoption.

To use the Microsoft Auto Service Module, you can build an application that enables users to query the search provider and to view the search results on a Microsoft Auto–powered device. You can also integrate point-of-interest (POI) data with a separate third-party navigation application so that users can view the location of the POI on a map, together with the POI name, the address, the telephone number, and other useful information. With the POI plotted on a map, a third-party navigation application can generate turn-by-turn directions to the POI from the current Global Positioning System (GPS) location of the Microsoft Auto–powered device.

Microsoft: The Right Partner for You

Microsoft Auto brings the advantage of partnering 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 if they are not financially supported, they will not be there tomorrow, which potentially leaves the responsibility for ongoing maintenance, support, and technical evolution to others.

Key reasons why Microsoft is the right partner for you:

1. We deliver state-of-the-art software platforms.
2. We apply our software assets for lower project implementation risk and faster time to market.
3. We employ a collaborative approach to achieve solution fit and knowledge transfer.
4. 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 3.

Figure 3. Microsoft assets

About Us: Microsoft Automotive Business Unit and Ecosystems

Microsoft created the Microsoft Automotive Business Unit (ABU) in 1995. The ABU—a multidisciplinary group that is composed of product developers and business leaders in North America, Japan, and Germany—is a dedicated partner to the automotive industry. It provides innovative technologies and flexible software—including Microsoft Auto 4.0—to help deliver reliable, easy-to-implement, and cost-effective in-vehicle infotainment solutions.

The ABU has offices in all the major geographies for automotive development, as shown in Figure 4.

Figure 4. Teams located near the main centers of automotive development

Microsoft Auto 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 Microsoft Auto development ecosystem, which includes Microsoft engineering and the Partner Response Team, 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 Microsoft Auto system. Located in the new "Studios West" campus in Redmond, Washington, this engineering team is co-housed with the engineering teams of most of the components that are provided together with Microsoft Auto, including Windows® Embedded CE, Zune®, and Windows Mobile®. This proximity enables close collaboration between the engineering teams and provides a unique opportunity to stay well connected to the communication and entertainment trends in the consumer electronics world. This tight relationship helps keep Microsoft Auto at the leading edge of innovation.

Partner Response Team

In addition to the standard product support that Microsoft offers in several payment models, Microsoft Auto also provides access to a team of highly skilled resources called the Partner Response Team (PRT).

The PRT options give customers the opportunity to secure ABU-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 PRT engineers is to accelerate the project's time-to-market and to make sure that the project satisfies all of the necessary requirements.

Automotive Business Unit Partner Ecosystem

The ABU partner ecosystem connects Microsoft, tier 1 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 ABU 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 Microsoft Auto partner program. (See Figure 5.)

Figure 5. ABU partner ecosystem

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

- **For automakers**: 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's ongoing commitment to the automotive industry helps 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: Microsoft Auto on the Road

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."

Today, drivers and passengers can experience Microsoft automotive technology through Microsoft Auto in more than 80 vehicle models worldwide. Figure 6 describes some of the solutions that use the technology built into Microsoft Auto to satisfy evolving customer needs.

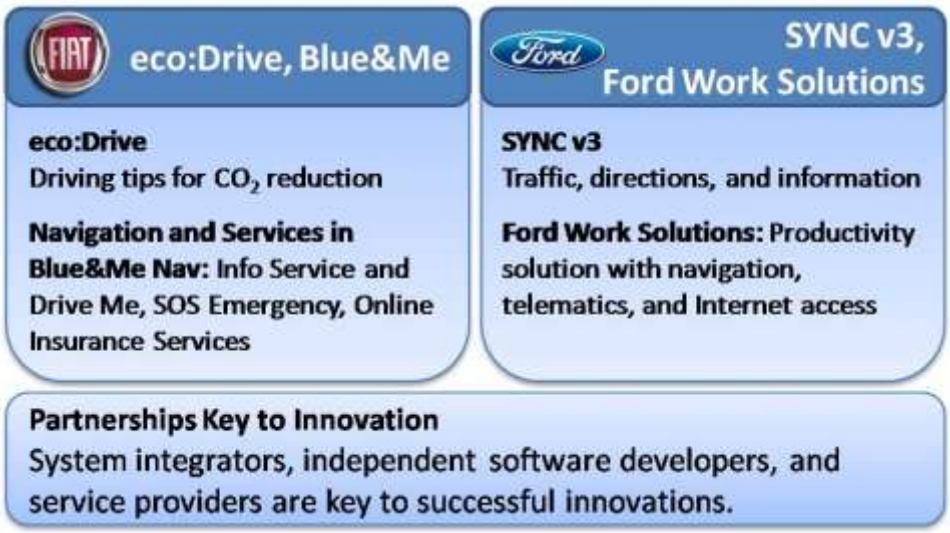


Figure 6. Microsoft Auto success stories

Fiat Blue&Me

Fiat Auto Group and Microsoft jointly developed the infotainment system Blue&Me, which empowers customers to connect their personal mobile devices with the integrated solution in various vehicle models from Fiat, Alfa Romeo, Lancia, and Fiat Light Commercial Vehicles. This infotainment concept was new and innovative, and the solution was completed in less than two years. As of March 2009, over 700,000 Fiats equipped with the Microsoft-powered Blue&Me have been sold.

This competitively priced infotainment package is voice controlled and includes Bluetooth wireless technology and Universal Serial Bus (USB) connectivity, which allows drivers to connect to a large number of mobile phone models and media players. The Blue&Me system is based on a modular structure and can therefore be easily updated to support different services. For example, customers can download language packs from the Fiat Web site and update the system to support a language that was not originally installed. Or, customers can add a new application as soon as it becomes available.

- **Blue&Me Nav** extends the phone and media functionalities with a GPS 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 emergency, information, 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. With an original design that was developed together with Magneti Marelli specifically for the Fiat 500, the device represents a new frontier in the portable navigation systems market.
- **Fiat eco:Drive** is another step in creating innovative applications. It 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 drivers plug in to a computer. The eco:Drive system presents drivers with the detailed environmental performance of the car, including the CO_2 emission level for each trip. It analyzes the drivers' driving style and provides tips and recommendations for modifications to reduce CO_2 emissions and to save money on fuel.

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

Ford SYNC is a factory-installed, fully integrated in-vehicle communications and entertainment system that was developed by Microsoft and the Ford Motor Company. SYNC 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, Zune, PlaysForSure-verified devices, and most USB storage devices. Supported audio formats include MP3, Advanced Audio Coding (AAC), Windows Media® Audio (WMA), and Waveform Audio Format (WAV).

SYNC is based on an Advanced RISC Machine (ARM) 11 processor that has 64 megabytes (MB) of dynamic random access memory (DRAM) and 256 MB of flash memory. Customers can use the USB port to update the software to work with the newest personal electronic devices. This is an important advantage, because customers tend to change devices more frequently than they change vehicles. SYNC debuted in the fall of 2007 on 12 different 2008 models of Ford, Mercury, and Lincoln vehicles. By the end of 2009, Ford will install SYNC on all vehicle models.

Automakers and tier 1 suppliers can ensure that their systems stay current with the latest devices and services thanks to the inherently flexible design and built-in software update mechanisms of Microsoft Auto. Unique solutions and customized user interfaces can also be easily built on top of the existing platform.

- **Ford Work Solutions** is an in-dash computer that was developed by Ford and Magneti Marelli and that is powered by Microsoft Auto. It provides full high-speed Internet access over the Sprint Mobile Broadband Network and navigation by Garmin. This system allows customers to print invoices, check inventories, and access documents that are stored on their home or office computer networks—right from the job site.
- **SYNC with Traffic, Directions, and Information** expands Ford's connectivity leadership by providing 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, on the Microsoft Auto software platform. It 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.

Continental Multi Media Platform

Continental uses Microsoft Auto for the Continental Multi Media Platform (MMP), which provides powerful, secure, flexible, and easy-to-update in-vehicle multimedia systems. The MMP software architecture clearly separates vehicle functions from functions that are related to entertainment features, which makes it possible to quickly react to future innovations and market trends. In the MMP hardware design, Continental uses a scalable concept to ensure top performance at high integration. For high-end systems, another CPU and graphics processor provides additional power for online services and three-dimensional graphics applications.

The hardware is equipped with standard consumer electronics interfaces, so that mobile devices (such as USB storage devices, iPod devices, or Secure Digital [SD] cards) can be easily networked with the MMP. The MMP also provides a Bluetooth wireless technology interface, so that consumers can use mobile phones for hands-free phone calls or mobile data services.

Awards for Microsoft Auto

Microsoft Auto has a history of success, as demonstrated by the following awards.

Table 1. Awards for Microsoft Auto

2009	*Four industry awards including:* • **Automotive News PACE Award** for Microsoft Auto • **Popular Mechanics CES People's Choice** for SYNC
2008	*Three industry awards including:* • **Bluetooth SIG—Best of CES 2008** for SYNC
2007	*Ten industry awards including:* • **CNET Editor's Choice Award 9.0/10.0** for JVC KD-NX5000 (Windows Automotive–powered device
2006	*Seven industry awards including:* • **J.D. Powers & Associates—Customer Satisfaction Award** for Alpine's Windows Automotive-powered device

Deep Dive into Microsoft Auto 4.0

Microsoft Auto 4.0 is an ideal application development platform because it provides a rich programming environment that empowers application software developers to be able to add their own functionality. The flexible Microsoft Auto 4.0 platform targets a wide range of devices, including connectivity gateways, connected radios, multimedia devices, and navigation head units.

This section of the paper takes a more detailed look at the components of the Microsoft Auto 4.0 platform. First, the paper discusses the individual components of the platform. Then, the paper describes how these components work together to provide a foundation for a range of in-vehicle devices.

Microsoft Auto 4.0 Components: The Basic Building Blocks

The following diagram gives an overview of the various building blocks that make up the Microsoft Auto 4.0 system. Some are provided by Microsoft, and others are provided by partners.

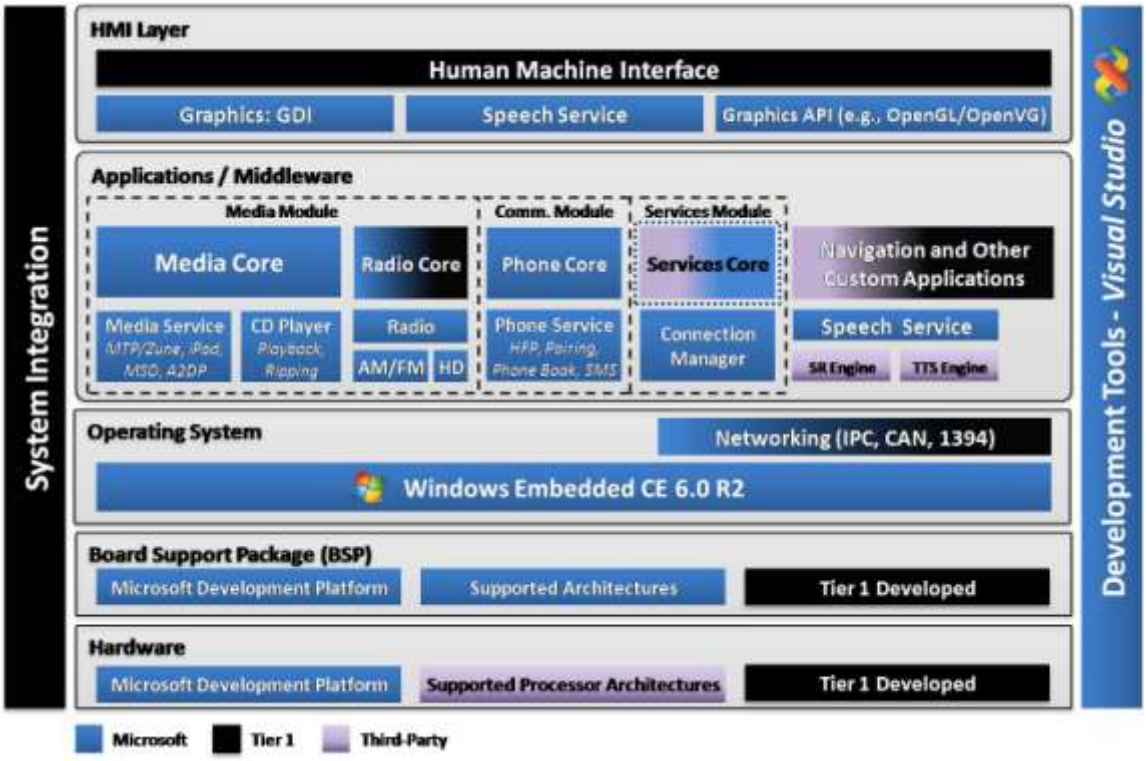


Figure 7. Microsoft Auto 4.0

Microsoft Auto 4.0 includes the following components:

- **Hardware**. The Microsoft Auto development platform is a hardware implementation of all Microsoft Auto features that facilitates rapid prototyping. This development reference 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 (BSPs)/Drivers**. BSPs and drivers are available in the Microsoft Auto 4.0 Platform Development Kit (PDK) and through hardware suppliers. The Microsoft Auto 4.0 PDK includes BSPs for Microsoft's development reference platform based on Freescale i.MX31 and i.MX35, Texas Instruments Jacinto EVM, and Renesas SDK 7785. Intel plans to have an Intel Atom-based Microsoft Auto BSP available in the autumn of 2009.
- **Microsoft Auto base operating system.** Microsoft Auto 4.0 is built on Windows Embedded CE 6.0 R2 and was written from the ground up to provide a component-based, real-time operating system for embedded devices.
- **Microsoft Auto middleware**. Microsoft Auto 4.0 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.
- **Microsoft Auto 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 made productive more quickly. The applications have been designed so that the HMI is easily separable; 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 chose from many development tools and runtime libraries, including OpenGL/OpenVG and Graphics Device Interface (GDI), to develop two-dimensional or three-dimensional 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 Design

Hardware options:

- Support for Intel iA86 processors
- Support for Renesas SH4-based processors (SH7785)
- Support for Freescale i.MX 31 and i.MX35
- Support for Texas Instruments Jacinto
- Microsoft Auto can run on processors that use the ARM v4, SH4 , and x86 instruction sets

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 hardware reference design.

Microsoft Auto 4.0 supports a hardware reference design, MARPF2, that 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

hardware reference 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 reference platform also comprises a flash disk drive that has a data transfer rate of 22 MB/sec.

The Microsoft Auto 4.0 software is also compatible with other ARM-based and SuperH (SH)-based microprocessors and now supports Intel architecture processors, including the newest options for the Intel Atom Z5xx series processors that offer an industrial temperature range. Microsoft Auto 4.0 also includes a BSP for the Texas Instruments Jacinto EVM processor and the Renesas SDK7785.

The hardware reference design uses 256-MB NAND flash memory to store the operating system image, the applications, the speech engines, and additional application data. 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 Initial Program Loader (IPL), the Update Loader (UPL), the device parameter store, and the field-programmable gate array (FPGA) configuration modules. The IPL is only updateable through Joint Test Action Group (JTAG) programming. The Secure JTAG feature of the CPU prevents unauthorized access to this region.

Note that the development hardware reference design is not mandatory to build devices that are based on Microsoft Auto 4.0. Suppliers can choose one of the supported processor platforms that ship with Microsoft Auto 4.0, or they may choose to develop a Windows Embedded CE 6.0 BSP for another processor family. Microsoft provides documentation that describes the requirements for transforming a standard Windows Embedded CE 6.0 BSP into one that can fully support the functionality of the Microsoft Auto 4.0 platform. Because of the layered architecture, the operating system, middleware, and applications should not be significantly affected when a standard Windows Embedded CE 6.0 BSP is repurposed into one that supports Microsoft Auto 4.0.

For additional details about the development hardware reference design and for a schematic of the development platform, see Appendix 2: Microsoft Auto Base Components and Hardware Reference Design.

Board Support Package and Systems

Included board support packages:

- Renesas SDK-7785 system development kit
- Microsoft MARPF1 and MARPF2 development hardware based on Freescale i.MX31 and i.MX35 processors
- Texas Instruments Jacinto processor and the EVM (TMS320DRA446) prototyping circuit board

Boot times measured by Microsoft on the Microsoft Auto hardware reference design:

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

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

Microsoft Auto 4.0 starts 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 3: Boot Times.

The OAL includes the lowest-level components that initialize the CPU, the peripherals, and the other hardware modules. The OAL includes code to handle interrupts, timers, power management, and input/output controls (IOCTLs). The OAL interacts directly with the hardware and abstracts the specifics of the hardware from the upper layers. The OAL is typically built when the BSP is developed; it is built as a library (HAL.lib) and linked into the overall kernel executable (Nk.exe).

The hardware-specific device drivers in Microsoft Auto 4.0 include boot loaders, Bluetooth, USB 2.0 Host, NAND flash, and networking drivers. Most 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.

Boot Loaders for Microsoft Auto 4.0

Microsoft Auto includes the following boot loaders:

- **Step Loader**. The Step Loader is the initial software that is loaded from storage and is usually of a fixed size (2 KB on the MARPF2). This module must be placed on the first block of the NAND flash.

  The Step Loader performs the following tasks:

  1. It initializes and configures the CPU (for example, disables interrupts and configures the memory management unit [MMU], the cache, the coprocessor, the ARM peripheral, and the synchronous dynamic random access memory [SDRAM]).
  2. It relocates itself into random access memory (RAM).
  3. It copies the IPL into RAM and then jumps to the IPL.

  Platforms that do not support starting from NAND flash must support starting from NOR flash, and a NOR-based Step Loader must be created. In this scenario, a dummy Step Loader is added to the image that is flashed to NAND.

- **Initial Program Loader (IPL).** The IPL module determines the device mode (normal boot, update, or development mode).

  The IPL performs the following tasks:

  1. It decides which image to start 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. It locates the image by reading the device parameter store (DPS) and by retrieving the data for the location (on the flash memory), the image size, and the jump address on the SDRAM.
  3. It copies the image from NAND to SDRAM and then jumps to the image.

- **Ethernet Boot Loader (EBoot)**. The EBoot is loaded by the IPL; it enables the launching of resident images or the downloading and flashing of new images onto the board. 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). Note that the EBoot is not part of a final released product image.

  The EBoot performs the following tasks:

  1. It initializes the CPU. (If the CPU is not already initialized by the Step Loader, the EBoot enters a supervisor mode and then clears the instruction and data caches and Translation Lookaside Buffers [TLBs], clears or masks interrupts, and then initializes the phase lock loops [PLLs].)
  2. It creates and populates the reserved memory area to be shared with the operating system.
  3. It sets up the debug Universal Asynchronous Receiver/Transmitter (UART) and provides a user menu for configuring loader options.
  4. It initializes the Ethernet controller.
  5. It obtains the IP address for the target from a Dynamic Host Configuration Protocol (DHCP) server or assigns a static IP address.
  6. 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. 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 itself does not provide any menu options. As soon as the SBoot is launched, it tries to connect to the desktop USB downloader application to initiate the image download over USB.

  The SBoot contains security features that lets it flash only verified images. It downloads .sec images, which consist of the regular image, its signature file, and a simple header.

After a successful download, the SBoot verifies that the image is signed by a trusted source by using the MINCRYPT library. Then, the SBoot starts the flashing process.

The SBoot is included on both development and production images, which require security features for flashing the new images.

- **Hardware Test Utility (HWTU)**. The HWTU is a suite that is used to test and verify several hardware features. It is loaded and launched by the EBoot as a menu option. Like the EBoot, the HWTU is a development utility and is not shipped as part of the production image.
- **Image update file system (IMGFS)**. The IMGFS is the main Windows Embedded CE image that has the TFAT partitions included. It is loaded and launched by the IPL. The IMGFS has its own mechanism to define individual NAND flash blocks and is not aware of DPS. Boot loader components and DPS regions are marked READ-ONLY and RESERVED on the flash memory to make sure that they are not overwritten. A distinct advantage of the IMGFS is the ability to update the components that are stored in the IMGFS by using the UPL.
- **Update Loader (UPL)**. The Image Updater is a functional subcomponent of the Device Management Service, which provides reliable updates to the Windows Embedded CE image. The Image Updater uses the Windows Embedded CE UPL 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**. The DPS access library, also referred to as the Auto DPS Library, consists of two main components:
  - One component is used to access the DPS at the BSP level, such as in the boot loaders.
  - Another component is used to access the DPS from within Win32 applications.

  The first component is provided as a library that gets statically linked with a BSP module during the build process. The second component is provided as a library that gets dynamically linked with an application during the build process.

  Accessing the DPS by using these libraries follows the same philosophy in that these APIs require the developer to provide a segment identifier number; an offset, in bytes, from the start of the segment to the desired data; and a size, in bytes, of the desired data. All the APIs perform validation checks to help ensure that accesses do not cross segment boundaries. The library makes no assumption about the structure of the data within a segment; it is entirely up to the application that accesses the DPS to interpret the results. Any failure that is encountered when the DPS is accessed results in a FALSE value being returned by the access APIs. A success results in a return value of TRUE.

Power Management

The main goals for power management in in-vehicle devices are to help ensure that adequate power is available to various modules in a specific operating mode and to prevent draining the vehicle battery.

The power management framework in the Microsoft Auto hardware reference design provides the ability to operate the system at a predefined set of power consumption levels. It also provides a way to define various power state transitions. Microsoft Auto 4.0 implements a custom platform-dependent driver (PDD) for the Windows Embedded CE 6.0 Power Manager (PM) that helps enforce power policy.

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

- **A set of defined 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 CE and have the same meaning for all devices.

- **Defined conditions or triggers.** These triggers cause transitions between the power states. Microsoft Auto 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 suspend current for the MARPF2 development platform is approximately 1.5 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 Arbitration and Management

Microsoft Auto 4.0 supports multiple audio sources and multiple audio outputs. 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). Note that the mono out line plays the left channel only.

Also, depending on the source and destination of the audio stream, 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 and mixing requirements. Audio management functionality is responsible for these requirements and considerations.

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

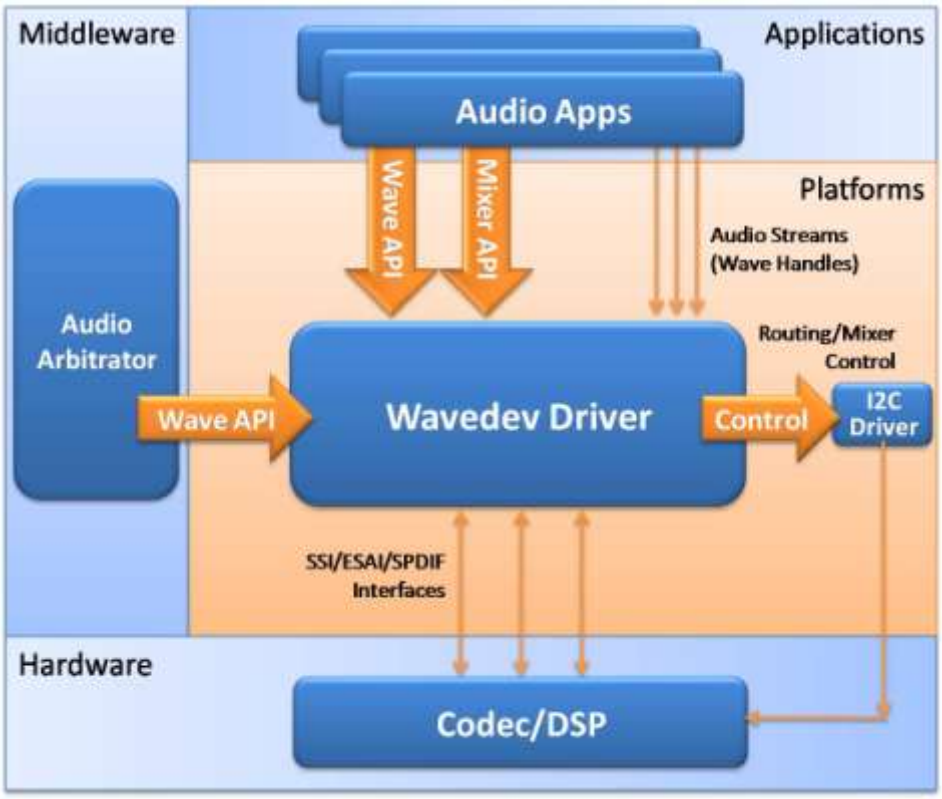


Figure 8. Audio system architecture

The Microsoft Auto audio system comprises 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, as well as 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 Microsoft Auto Wavedev driver supports the standard Windows CE Wavedev APIs, as well as 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 digital signal processing (DSP) and hardware codecs. Note that the Wavedev 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 CE Wave API (as documented in the Windows CE Help files that are shipped with Platform Builder). Each Wave API request is handled by the Audio Device Manager (which manages audio devices and the kernel software mixer). The Audio Device Manager in turn makes a request to the Wavedev driver through the Wave Device Driver Interface.

In the Microsoft Auto 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 or recording of Pulse Code Modulation (PCM) audio, 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

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 Microsoft Auto to play over the car's speaker system.

Networking

- IEEE 1394 support
- Enables automakers to develop navigation

Microsoft Auto 4.0 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.

Although vehicle network stacks are not part of Microsoft Auto 4.0, two different models to integrate vehicle networking into a Microsoft Auto–based device have been used in current production vehicles. A direct vehicle connection stack runs on the infotainment CPU and implements all the vehicle networking protocols. An indirect vehicle connection runs on a secondary CPU, often referred to as the vehicle CPU, 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 Microsoft Auto. Third-party products can also be used for other networks, such as Local Interconnect Network (LIN) or FlexRay, if desired.

IEEE 1394 Bus Architecture

The IEEE 1394 bus in Microsoft Auto 4.0, a high-performance serial bus that was specifically designed for entertainment and communication applications, can be used to transfer data at high speeds.

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

The IEEE 1394 bus driver acts as a Power Manageable Device Driver (PMDD). The IEEE 1394 bus driver supports the Intelligent Transportation Systems Data Bus (IDB)-1394 and utilizes the power management capabilities of IDB to efficiently manage power for devices.

When the Windows CE power management driver (Pm.dll) sends any request to change the power state, IOCTL_POWER_SET is called. Figure 9 shows how an application can use the IEEE 1394 to interact with the Microsoft Auto hardware. Terms and acronyms are defined in the Glossary.

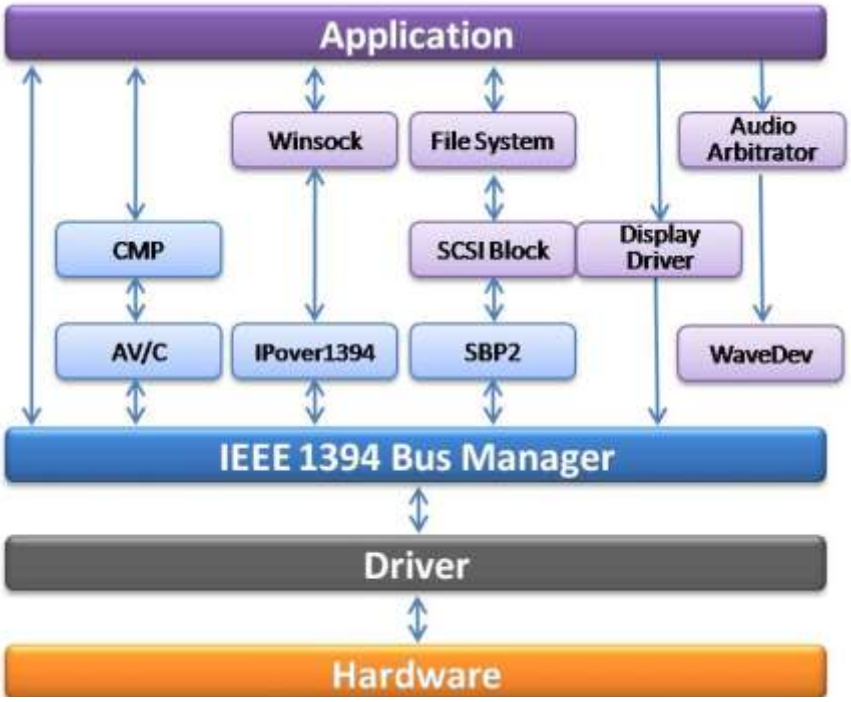


Figure 9. IEEE 1394 architecture

The 1394 features enable:

- Uploading navigational data to a navigational device
- Downloading games, music, and video over the Internet on the rear-seat entertainment devices
- Performing updates to obtain new applications or operating system updates
- Streaming location information on the rear-seat entertainment devices
- Controlling devices through speech-based interaction
- Playing media from a USB storage device
- Streaming multi-channel audio
- Streaming video

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

Microsoft Auto 4.0 is built on top of the Windows Embedded CE 6.0 R2 operating system and adds automotive-specific features. Windows Embedded 6.0 R2 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 CE 6.0 R2 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 CE 6.0 R2 offers a competitive advantage through faster, more efficient design cycles and an extensive array of features.

Functions of the Base Operating System Layer

The Microsoft Auto 4.0 platform builds on a specific set of Windows Embedded CE 6.0 R2 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 Microsoft Auto 4.0 system. Device makers that are using Microsoft Auto can customize the Windows Embedded CE 6.0 R2 and Microsoft Auto 4.0 components that are included in their specific device image.
This base operating system layer includes support for:

- File systems
- Windowing and focus management
- Speech API components that expose the chosen speech recognition (SR) and TTS engines
- 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 CE 6.0 R2 kernel exposes core system functionality and provides the infrastructure through which the rest of the system software can interface with the OAL. It is primarily responsible for process and thread management, predictable thread scheduling, memory management, interrupt handling, and support for system calls.

Features of Windows Embedded CE 6.0 R2

Windows Embedded CE 6.0 R2 provides a portfolio of hundreds of carefully tested operating system components, along with innovative tools and features that help enhance performance, security features, compatibility, and flexibility. These innovative tools include:

- **GPS.** Microsoft Auto 4.0 supports GPS positioning of the vehicle and communicates the position to the applications that need the information. Microsoft Auto 4.0 uses the Windows Embedded CE 6.0 R2 GPS Intermediate Driver (GPSID) architecture. A developer can create a driver that feeds the GPSID with GPS data (either from a GPS chip onboard the device, from GPS that is available on the vehicle network, from a Bluetooth wireless technology connection that is feeding GPS signals, or from another source). Since the developer controls the functionality of the source of the GPS data, that data can be raw feeds directly from a GPS chip, or it can be corrected GPS data based on dead-reckoning algorithms. The GPSID architecture then offers multiplexed application access to the GPS data through a standard interface at the top edge of the GPSID component.
- **Wi-Fi support.** There is rich networking support for Wi-Fi in Windows Embedded CE 6.0 R2, including:

- DNS, TCP/IP version 4 and version 6, and WinSock
- Firewall support, Internet Connection Sharing (ICS)
- Network Driver Interface Specification (NDIS), network utility support (such as `ping` and `ipconfig`)
- Local Area Network (LAN) IEEE 802.3 and 802.5, Wi-Fi IEEE 802.11

Additional features of Windows Embedded CE 6.0 R2 that are used by Microsoft Auto 4.0 are shown in Table 2.

Table 2. Features of Windows Embedded CE 6.0 R2

Architecture	Built-in tools, a layered architecture that features advanced real-time operating system technologies, and integrated middleware components that accelerate core development allow software integrators and tier 1 suppliers to spend more time focusing on differentiating infotainment devices in the competitive automotive marketplace.
Unified kernel	The Windows Embedded CE 6.0 kernel handles more than 32,000 simultaneous processes, each with 2 gigabytes (GB) of virtual memory space. The file system supports larger storage media and file sizes (up to 4 GB) and removable media encryption.
Security features	A redesigned, one-tier security model feature is Secure Development Lifecycle (SDL) compliant and helps ensure that only authenticated applications can run on an embedded device.
Device performance	Improvements to the kernel architecture reduce the overhead of system calls between base operating system services, which improves operating system performance.
Microsoft Internet Explorer® support	Internet Explorer Web Browser for Windows CE (based on Internet Explorer 6), with an OEM-replaceable user interface is included. Render Web sites with Rich Text Editing (RTE); manage cookies, user history, and temporary files more efficiently. Through a redesigned Internet Explorer 6, these features, available with the R2 update, offer better security, better performance, and easier use.
Source code access	Debug, test, and make changes to an operating system image with access to Windows Embedded CE 6.0 Shared Source. Plus, make modifications and create differentiated features while maintaining control over intellectual property—without sharing any code.
Real-time operating system	Power high-performance embedded devices to manage time-critical responses with: • Nested interrupts • Per-thread quanta • 256 levels of thread priority
Multimedia support	Comprehensive multimedia support for a variety of formats, such as WMA, MP3, and DVD, is provided through the Microsoft DirectShow® API.

For more information about Windows Embedded CE 6.0 R2, see the Related Links section later in this document.

Middleware and Services

The next layers in the software stack are the Microsoft Auto 4.0–specific middleware and services. These components define the heart of the Microsoft Auto 4.0 platform; they distinguish this platform from the standard Windows Embedded CE and from the other platforms that are based on Windows Embedded CE technology.

A detailed view of the system building blocks that make up the software platform components for Microsoft Auto 4.0 is shown in Figure 10.

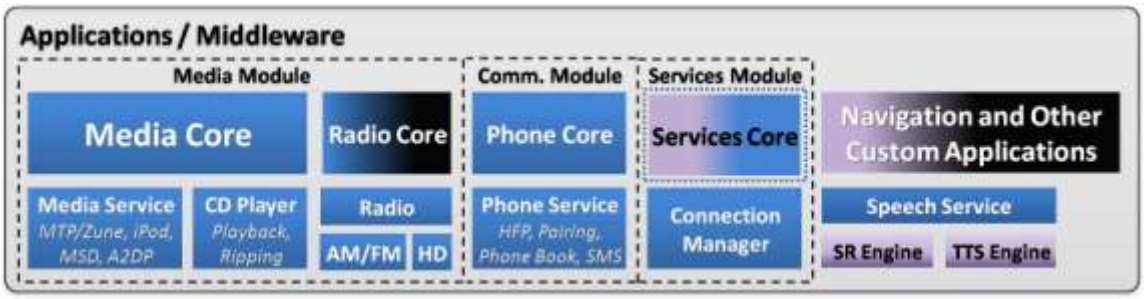


Figure 10. Platform components

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 Microsoft Auto 4.0 can use a broad array of C/C++ APIs. The base Windows Embedded CE 6.0 R2 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

Microsoft Auto 4.0 includes a set of services for:

- Hands-free telephony (HFP)
- Bluetooth profiles
- Phone management
- Data connectivity
- SMS

Applications that rely on telephony and data communications reside at the topmost layer of the telephony and data communications architecture and use Microsoft Auto 4.0 middleware services and Bluetooth profiles, as shown in Figure 11. Terms and acronyms are defined in the Glossary.

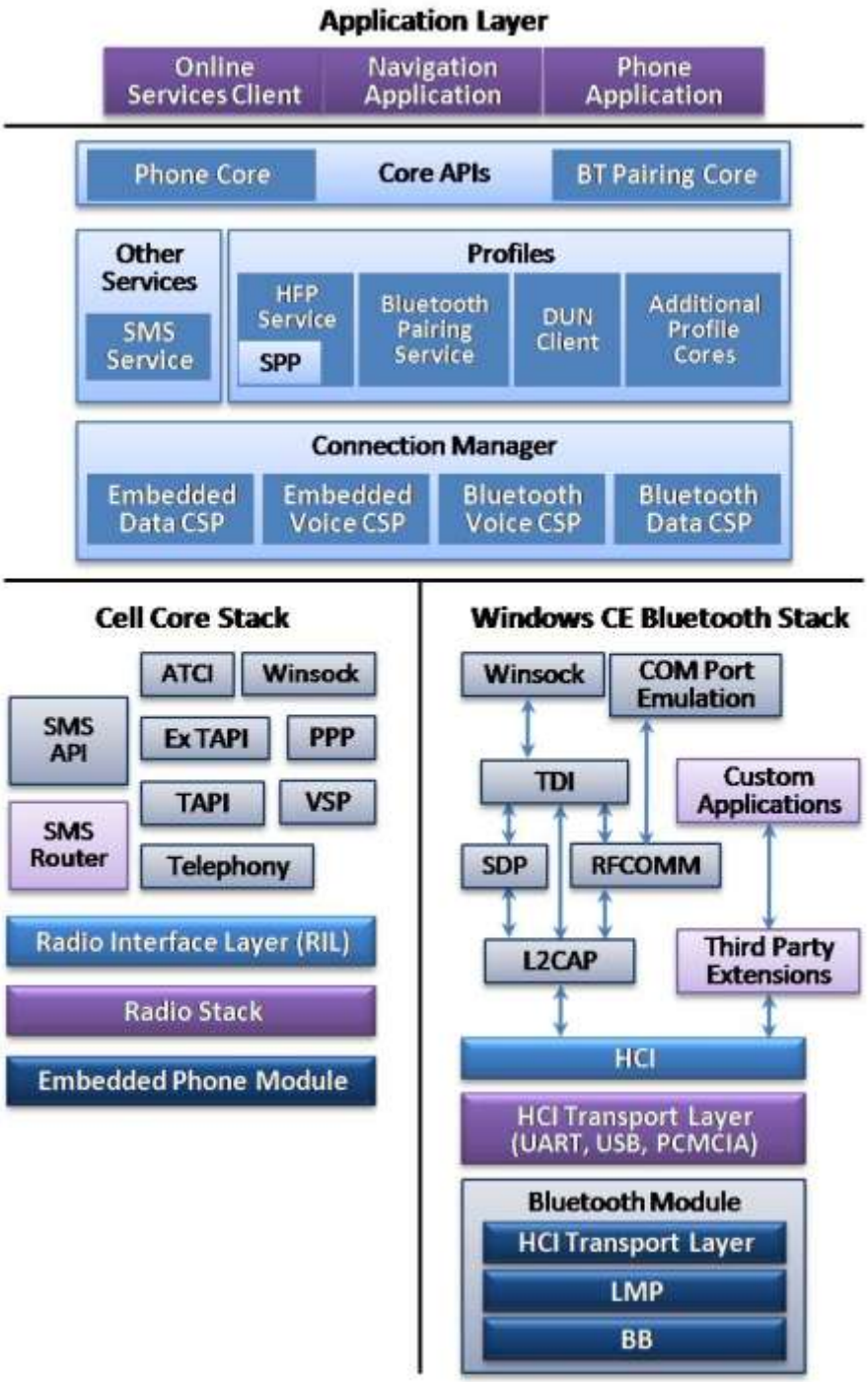


Figure 11. Telephone and communication architecture

For applications that use Bluetooth wireless technology for data communications, a profile is applied to describe how to exchange specific types of data. For example, 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 Microsoft Auto–based device using a Bluetooth gateway (a paired 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. (CellCore is a set of services that enables the creation of wireless-connection oriented services on a device, including the Radio Interface Layer [RIL], SMS support, Wireless Application Protocol Support [WAP], an Extended TAPI (Ex TAPI) and telephony service provider (TSP), Subscriber Identity Module [SIM] card support, an AT command interpreter [ATCI], and a connection manager.) Other interesting features—such as the ability to use Microsoft Auto 4.0 as a Bluetooth Dial-Up Networking (DUN) profile gateway—are also supported.

Besides telephony support, Microsoft Auto 4.0 also supports the data connectivity that is required for various system and application scenarios, such as device management and access to Internet-based services.

Data connections that are established with other data sources through the external mobile phone are managed by the connection manager (see the section about the Connection Manager). The connection manager centralizes and automates the establishment and management of these connections for Microsoft Auto 4.0 applications by handling the details of each connection. The connection manager can establish and manage network connections to communicate when IP packets are flowing through various pathways (such as Bluetooth and CellCore).

Bluetooth Software Stack

The Microsoft Bluetooth wireless technology stack implementation is a modular, general-purpose Bluetooth 2.0 + EDR–compatible software stack that is built on the Windows Embedded CE 6.0 Bluetooth stack; it is linked in the default Microsoft Auto 4.0 configuration. 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, USB, and Personal Computer Memory Card International Association [PCMCIA]) to the Bluetooth wireless technology chip.

Deployed Microsoft Auto 4.0–based systems that make use of the Microsoft Auto 4.0 Bluetooth solution may be updated in the field with the device management subsystem. The platform supports a pairing service, hands-free functionality, dial-up networking client, Object Push Profile (OPP)/Object Exchange (OBEX), streaming audio using A2DP, and the serial port protocol.

Windows Embedded CE 6.0 R2 includes additional Bluetooth wireless technology profile implementations that have not been explicitly tested with Microsoft Auto 4.0 but could provide a basis for additional solutions.

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

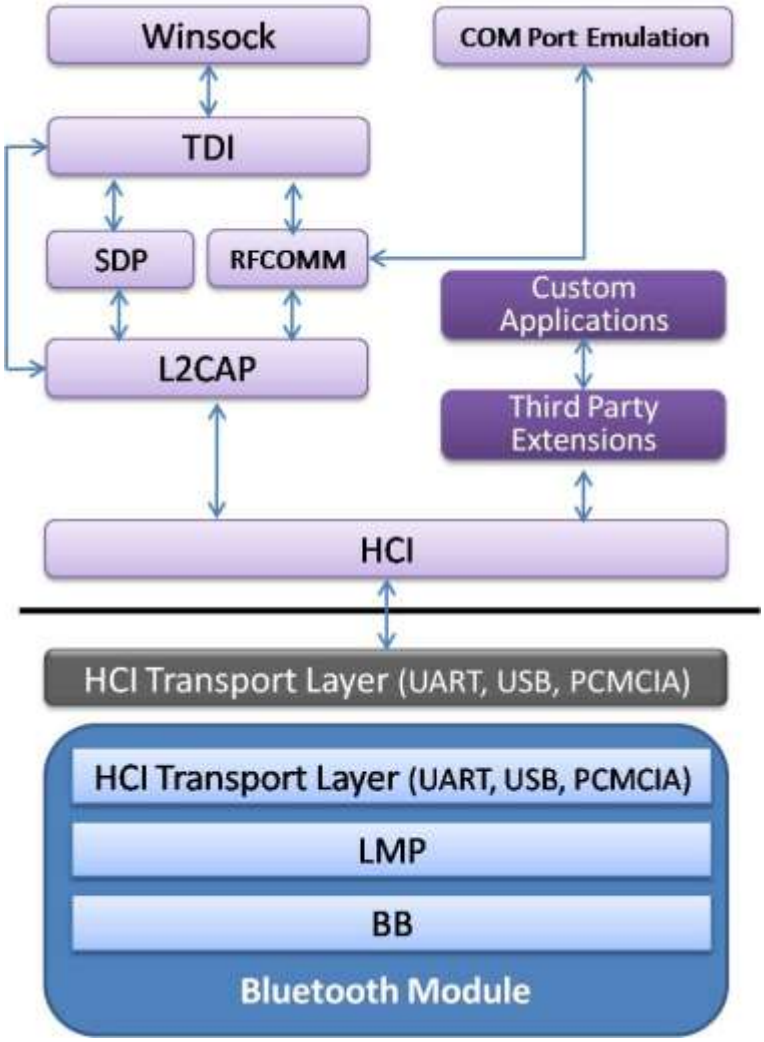


Figure 12. Bluetooth software stack

Bluetooth Connectivity

Microsoft Auto 4.0 provides the following Bluetooth 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

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
- Hands-Free Profile (HFP)-HF 1.5 (backward compatible to HFP 1.0)
- DUN Profile-DT and GW 1.1
- Message Access Profile (MAP) 1.0

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.)

The Bluetooth Pairing Service itself is not power aware. Because it is a service that is designed to be used by other applications in the Microsoft Auto 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 enabled, the pairing service waits for a pairing event from the Bluetooth stack. Upon receipt of this event and successful PIN negotiation, the device Bluetooth address (BT_ADDR) is checked against the existing set of known devices. If an existing record is found, the Bluetooth link key is obtained again, and the device record is updated. If the device is not found, a link key exchange is initiated, and, once paired, the device is queried for a set of profiles that Microsoft Auto 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, as well as with other components of the telephony and data communications stack.

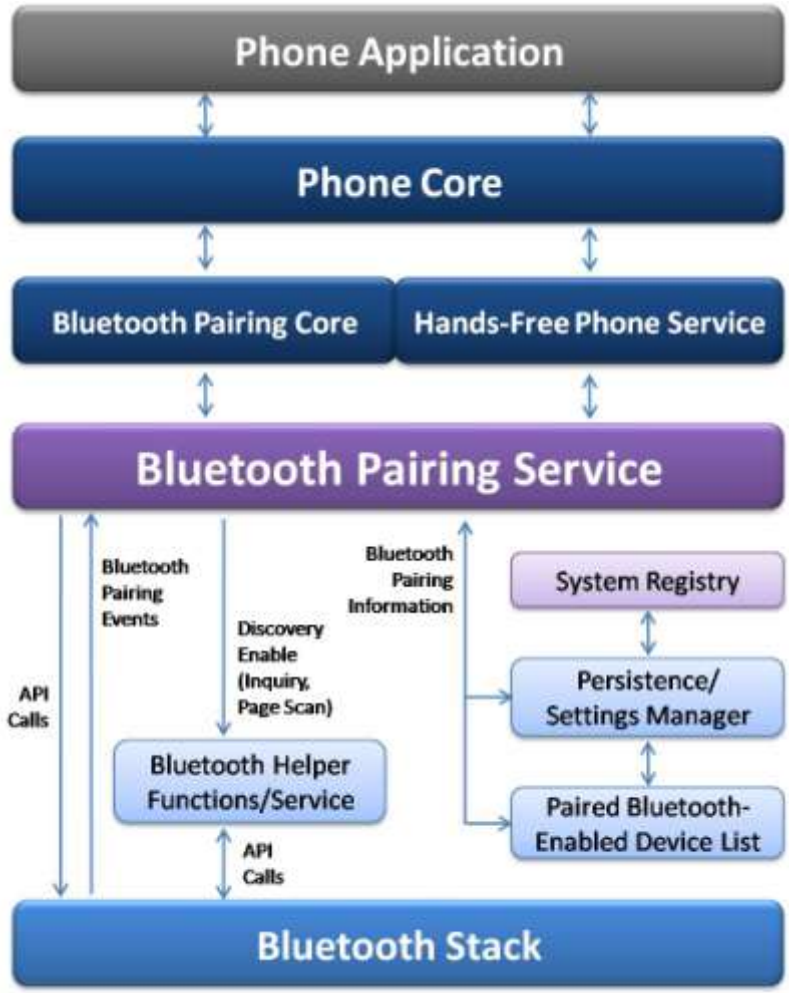


Figure 13. Bluetooth Pairing Service architecture

Bluetooth Pairing Core

The Bluetooth Pairing Core is one of the application cores that are used to provide core services to Microsoft Auto 4.0 applications.

Specific pairing applications may be developed that use the Bluetooth Pairing Core API to use the capabilities of the underlying Bluetooth Pairing Service to pair to and communicate with Bluetooth-enabled devices. Figure 13 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-proc DLL, a provision of straight APIs that wrap the 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. The phone core fills the gap between a phone application HMI layer and the Microsoft Auto 4.0 HFP service. The phone core is responsible for phone connection, call handling, call history, and phonebook management.

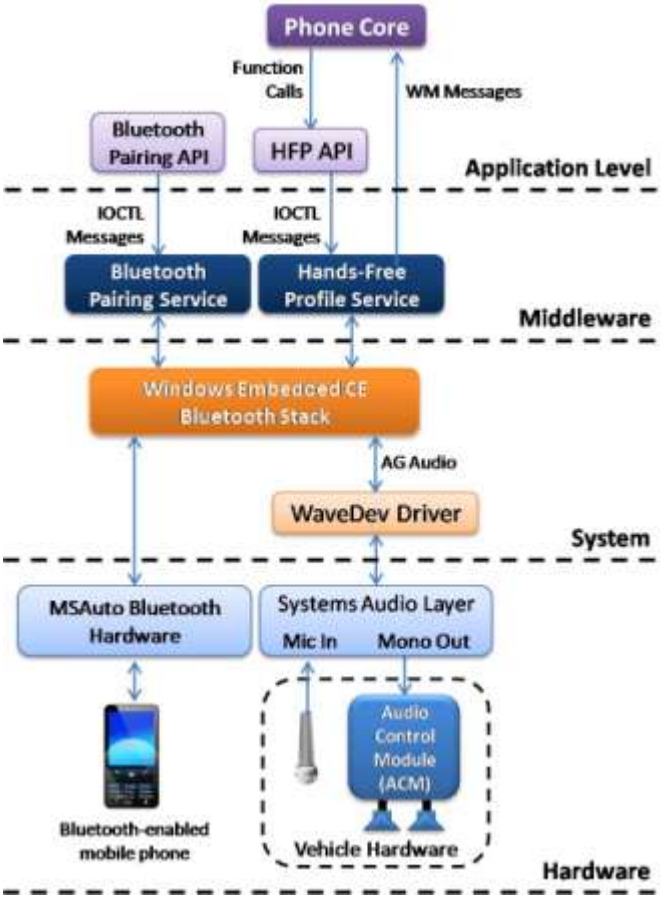


Figure 14. HFP architecture

Phone Core

Phone features:

- Support for hundreds of mobile phones, media players, and USB devices that were tested to help ensure broad market compatibility
- Semiannual 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 and that is customizable for additional data types
- Support for the new Bluetooth Phone Book Access Profile (PBAP), a preliminary implementation of Message Access Profile (MAP)*, and new high-quality ringtones

**Bluetooth SIG adoption of MAP was not final at publication of this white paper*

Microsoft Auto includes a set of services for HFP, Bluetooth profiles, phone management, data connectivity, and SMS. Microsoft Auto 4.0 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 Microsoft Auto–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 enables the application to 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.** Microsoft Auto 4.0 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 Management

The phone core utilizes the HFP API to download the phonebook from mobile phones that support the functionality through Bluetooth wireless technology. 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. Once each 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 (PBAP), by Synchronization Markup Language (SyncML), or by AT commands. The phonebook can also be filled through an OPP/OBEX vCard object that is push initiated by the user.

The system developer may also choose to use the legacy synchronization method, which enables Microsoft Auto's standard download logic with respect to protocols and timing to be used when it connects to a Bluetooth phone. The legacy method stores only names and phone numbers. However, it stores them in a manner that enables very efficient retrieval and provides APIs for maintenance and manipulation of these records.

Another alternative for the system developer is to choose to modify the SyncManager sample that is included in Microsoft Auto 4.0. This sample presents a framework for using the separate

services for downloading the phonebook directly and saving the results into the Pocket Outlook Object Model (POOM). This method allows the OEM to retrieve all the relevant vCard fields (such as contact pictures, street and e-mail addresses, and a variety of phone numbers with locations) for processing.

Note that a compelling hybrid method which combines the tested logic and retrieval efficiency of the legacy method and the extended fields of the SyncManager sample method is planned for delivery in the next Microsoft Auto Device Update Pack.

The Pocket Outlook Object Model

The phone service can use the POOM, a Microsoft COM-based library that provides programmatic access to Microsoft Office Outlook® Mobile Personal Information Management (PIM) data items and container objects. 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 Microsoft Auto 4.0 version of POOM provides an optimized search algorithm and specialized fields to enable convenient storage of records for each connected phone.

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 (described earlier in this document) and that can be used by a hands-free phone application that is provided by the carmaker or supplier in order to expose the functionality of these services to the user.

Short Message Service Support

Microsoft Auto supports access to SMS messages that are received by a connected Bluetooth phone and sending of SMS messages by a connected Bluetooth phone.

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 Microsoft Auto SMS support architecture.

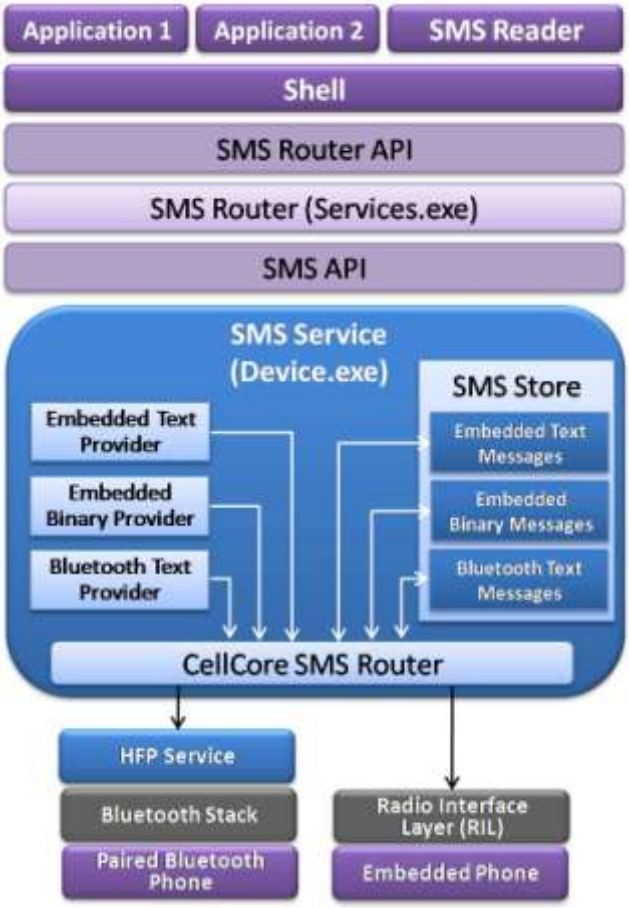


Figure 15. SMS support architecture

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 is all configurable by the developer.

Connection Manager

The connection manager is the central component for managing connections on the Microsoft Auto 4.0 platform. The connection manager provides an API to let applications request connections, specify priorities, and close connections after use. It manages platform network connections including embedded-connected and Bluetooth-connected phones, wireless connections (WLAN), and Wi-Fi.

When an application requests a network connection, the connection manager first retrieves all the possible connections from a set of connection service providers (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.

A supported connectivity feature is the firewall module. Windows Embedded CE 6.0 R2 brings an integrated firewall module to Microsoft Auto 4.0 that is configured by using registry settings; with this module, Microsoft Auto 4.0 is flexible enough to support a wide range of firewall scenarios and applications.

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

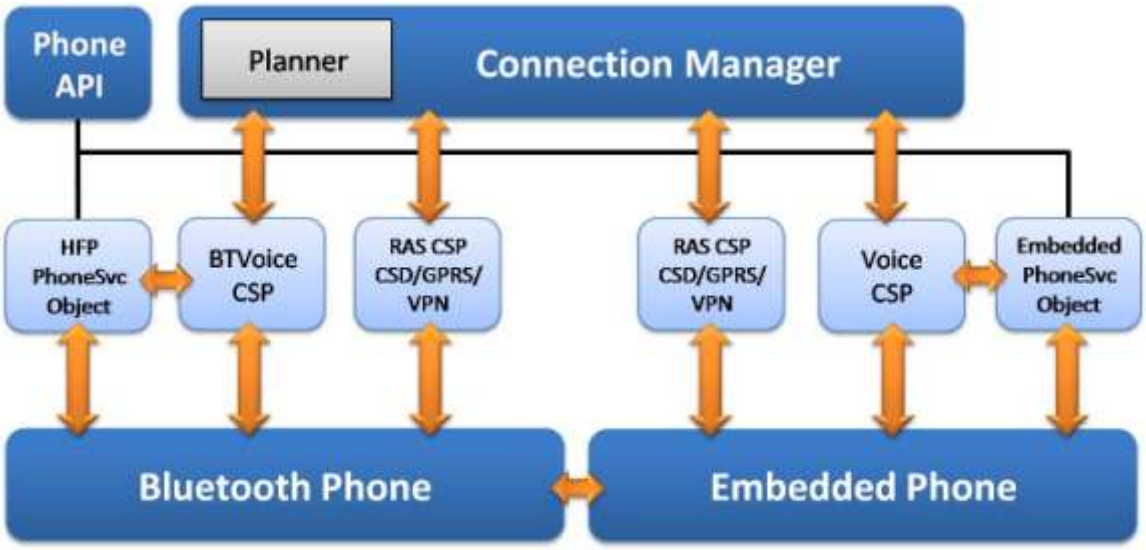


Figure 16. Connection manager

The Microsoft Auto 4.0 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, this 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

Microsoft Auto 4.0 provides entertainment access through the radio and media cores.

Radio Core

Standard programming interface for controlling the radio in the car:

- Support AM, FM, and HD radio
- Data from Radio Data System (RDS)
- HD radio support for Main Program Service (MPS), SPS (Supplemental Program Service), 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. The unified API is also extensible for additional radio types (such as Digital Audio Broadcasting).

Figure 17 shows the radio core architecture.

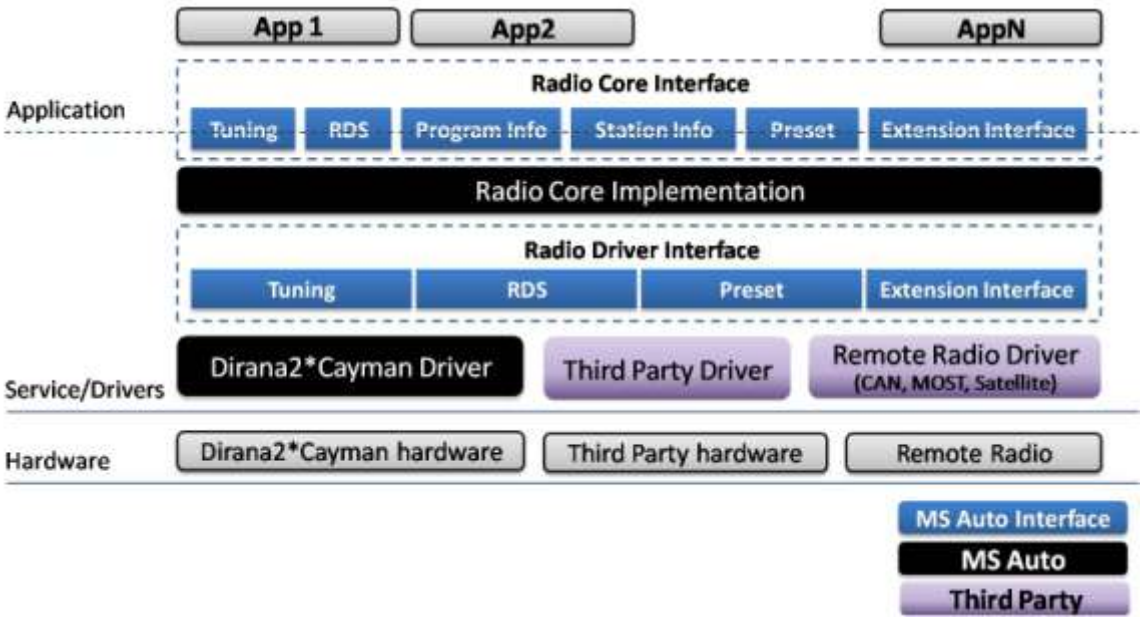


Figure 17. Radio core architecture

The radio core provides tuner support for:

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

Media Core

Media features:

- Supported media classes:
  - 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

The media core is the primary subsystem of the Microsoft Auto 4.0 platform that media applications use to abstract and manage device-specific interactions, such as indexing and playing.

By using the media core, developers can work with any of the wired device types, as well as locally stored media, through a single standard interface and can 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 4: 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**. Microsoft Auto 4.0 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. MTP-based devices from companies such as Sansa, Creative, and iRiver are also supported, 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 popular 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 CE 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 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 as well as 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.

Device Management

In-vehicle device management:

- Technology in Microsoft Auto 4.0 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

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, Microsoft Auto 4.0 supports both the update process itself and an implementation of a USB installer service. Updates can be delivered by USB key, by SD card, or by custom mechanisms.

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 Microsoft Auto 4.0 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 could also be used to download installable content for other electronic control units that are networked to the Microsoft Auto 4.0–based device.

The overall DM solution in Microsoft Auto 4.0 is primarily based on core Windows Embedded CE 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. This enables 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 CPF files. The installer itself is a DLL that exposes an installation API; it may be hosted in multiple processes.

  The installer is hosted either by the installer process (WCELoad.exe launched for speech language updates), by the DM client service (DMService.dll in Services.exe), or by the Microsoft Auto shell (ConvSh.exe for USB-based application installation).

- **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.

- **Super CAB files.** Super CAB files (PKS) bundle packages together into one larger package. These may involve a set of application CAB files or a set of system updates.

Figure 18 shows how DM can be performed from a USB pen drive–based installation.

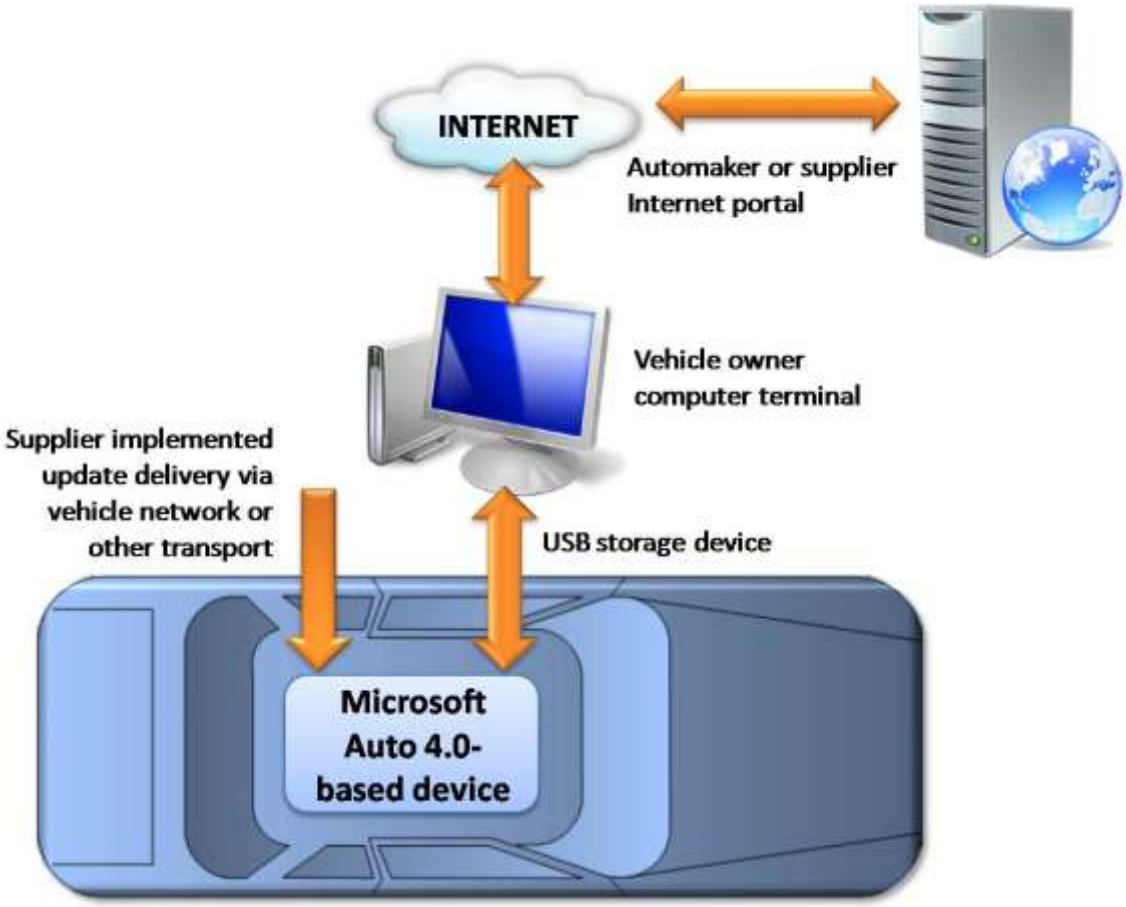


Figure 18. Device management

The device update process comprises two 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 INF file) and triggers installation if it finds the files.
- Performing the actual update by verifying and extracting the update image and 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 UPL, which verifies and applies updates to the operating system. Update progress and status is carefully tracked across restarts.

Security

The security subsystem helps to ensure that Microsoft Auto 4.0 does not accept or run executable code that is not trusted on the system. Microsoft Auto 4.0 is engineered with adherence to the Trustworthy Computing security standards that are enforced at Microsoft. (For more information, see http://www.microsoft.com/mscorp/twc/default.mspx.) The most critical security mechanism for Microsoft Auto is its application/operating system trust model.

Authenticode® code signing technology is used to help ensure that Microsoft Auto does not install or run code that is not trusted.

It is possible to configure the Microsoft Auto 4.0 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 image downloader help ensure that only certain signed images can be used to completely re-flash the system.
- **Operating system image updates.** Each of the image update packages has a certificate store (stored in a binary large object that is called Device Side Manifest [DSM] in the package). When you apply an image update, the installer verifies the image update package signature by using the policy stores (DSM) inside of the corresponding package on the flash memory. Then, the installer triggers a start into the UPL. The UPL verifies the signature of the update package again before it applies the update to the system read-only memory (ROM).
- **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.

  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 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 ROM.
- **Device authentication.** A service credential object is provided to applications for enabling device authentication.
- **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, Microsoft Auto 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 19 shows the image security certificate stores.

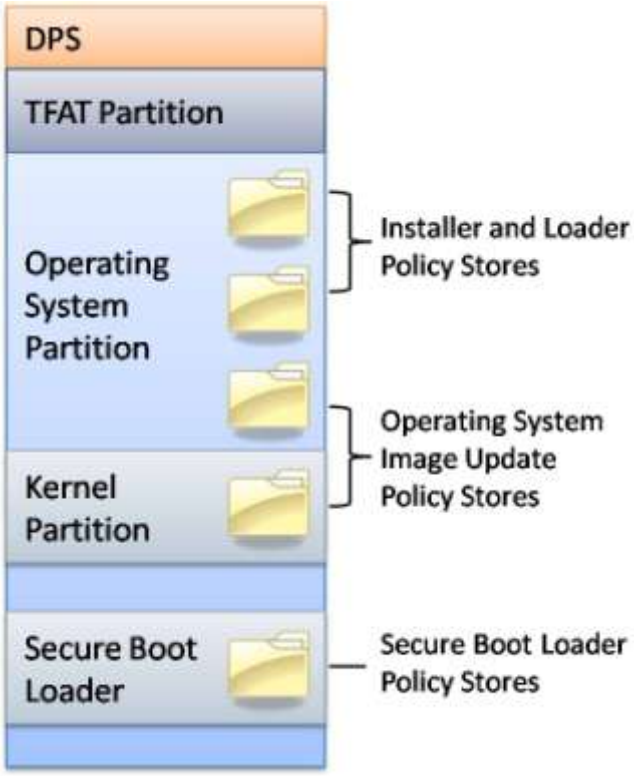


Figure 19. Security subsystem

To manage certificates and encryption keys on the device, Microsoft Auto 4.0 (through Windows Embedded CE 6.0 R2) provides the cryptographic API (CryptoAPI).

Cryptographic Keys

The device manufacturer programs a set of 128-bit (symmetric) cryptographic keys in the DPS of each device. The device manufacturer also programs a unique 64-bit product serial number, a unique identifier for the device, to the DPS.

When an application installation package is signed, it can be bound to a specific instance of Microsoft Auto 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 a 128-bit key as the password. Note that both the application and the remote service must use a salt to derive the actual password from the 128-bit unique cryptographic key for each Microsoft Auto 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

The following types of images are examples of image types that can be produced that enable development to continue without the need to produce digital certificates and then migrate to a secure image toward the end of development. An example organization of image types is as follows:

- PRODUCT. Nothing runs on this type of image unless it is signed with a product certificate. There are no boot loaders except the SBoot. You can upgrade a device with

any Microsoft Auto image type to this image type. After you flash this image type to the device or board, you must use JTAG to flash an image type other than a PRODUCT image type.

- PRODTEST. Nothing runs on this type of image unless it is signed with a product or test certificate. There are no boot loaders except the SBoot. You can upgrade a device that has a DEVTEST or PRODTEST image to this image type. If this image type exists on the device or board, you must use a secure downloader and a .sec image file to flash another image type.
- DEVTEST. This development image type does not enforce signing. It contains all types of boot loaders. You can upgrade to this image type from a device that has a DEVTEST image by using Platform Builder or from a previous PRODTEST image by using Secure Downloader.

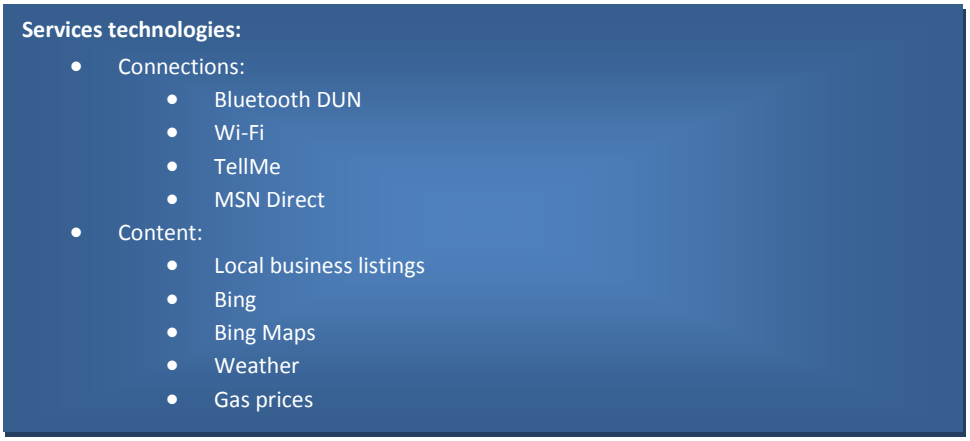
Reliability

Several subcomponents in Microsoft Auto 4.0 provide support for system reliability:

- **The hardware watchdog** helps ensure that the system is reset if it appears to be locked.
- **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 is switched to 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.

Microsoft Auto Services



The Microsoft Auto Service Center and the Microsoft Auto Service Module are Microsoft Auto Services components that enable prototyping of scenarios for providing the Microsoft Auto–powered device with connected information and services. This extensive development environment enables functions such as delivering location-based search (from a search provider) to a vehicle that has GPS. Other information, including weather and gas prices, is available to provide additional content for further development.

Microsoft Auto Services consist of the following components:

- **The Microsoft Auto Service Module application.** An application on the navigation device that provides an intuitive user interface (UI) and uses the Microsoft Auto Service Module API to retrieve search results from a search provider.

- **The Microsoft Auto Service Module API.** An API that is used to implement the Microsoft Auto Service Module protocol over HTTP. The Microsoft Auto Service Module application calls into the Microsoft Auto Service Center to retrieve search results from the search provider. The Microsoft Auto Service Module API is a native (C/C++) API.

  The Microsoft Auto Service Module API takes advantage of the following capabilities:
  - The ability to return telephone book listings based on the latitude, the longitude, and the radius of the search
  - The ability to change the ranking (relevance) of search results based on the latitude, the longitude, and the radius that are provided in the search request
  - The ability to interpret keywords
  - The ability to sort search provider results by distance (which is the default sort order) or by relevance
  - The ability to find the geographic center of cities, towns, and subdivisions within select regions
  - The ability to find current weather conditions, as well as weather forecast information
  - The ability to find gas station location along with the current gas prices

- **The Microsoft Auto Service Module transport layer.** A Microsoft Auto Service Module component that takes the request that is encoded by the Microsoft Auto Service Module protocol, the Device ID, and the Device Group ID to form an HTTP request

packet. Then, the transport layer sends the packet to the Microsoft Auto Service Center and receives the response.

- **The Microsoft Auto Service Center.** A remote server that receives the HTTP requests from the Microsoft Auto Service Module over an IP-based network. In order for the Microsoft Auto Service Center to receive these requests, the Microsoft Auto–powered device must be connected to a paired cell phone that uses GPRS to connect to an IP-based network either by using Bluetooth DUN or by using another mechanism that enables the IP connection (such as a built-in phone module or Wi-Fi radio).

  When the Microsoft Auto Service Center receives a request, it logs the requester Device ID and Device Group ID that are included in the request header. Communication between the Microsoft Auto Service Module and the Microsoft Auto Service Center occurs by using the Wireless Application Protocol Binary XML (WBXML) protocol over HTTP. If the request is permitted to pass through the Microsoft Auto Service Center, it is parsed and translated into a request that is compatible with the search provider. If the Microsoft Auto Service Module API call is not in the proper format, the Microsoft Auto Service Center returns status code 400 (Bad/Invalid Request). The Microsoft Auto Service Center might also cancel the request. (See Figure 21.)

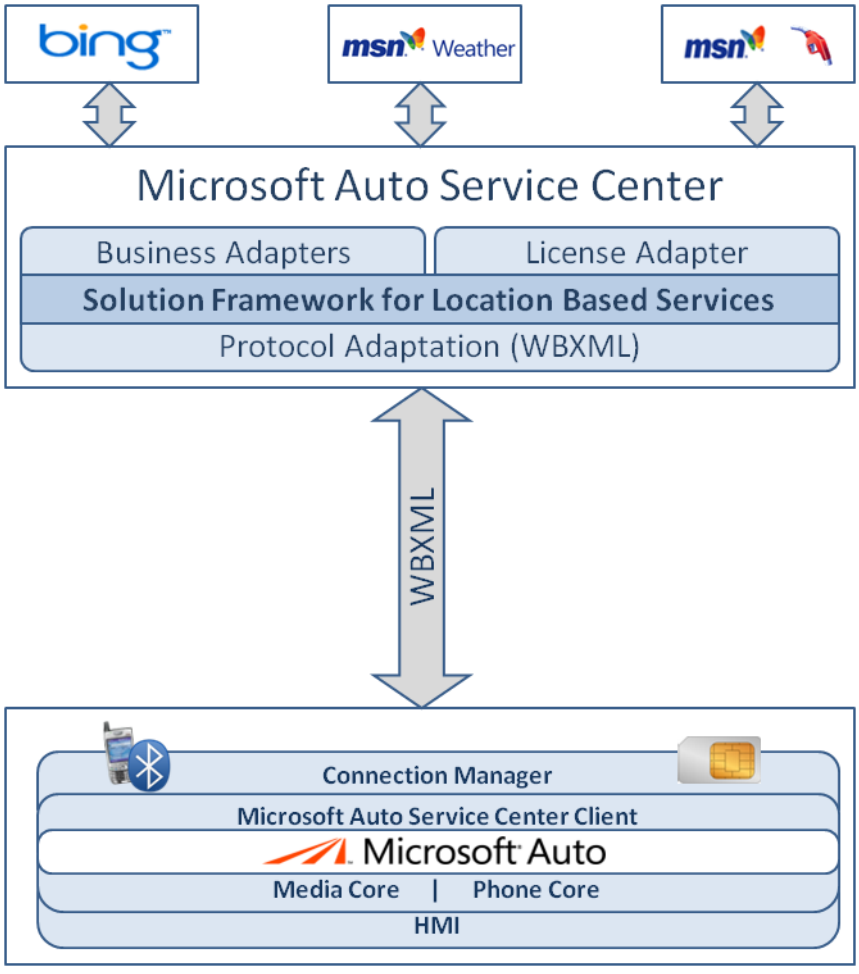


Figure 20. Microsoft Auto Service Center and module architecture

Additionally, the Microsoft Auto PDK provides examples that show how to connect to a service center. For example, the Microsoft Auto PDK describes how to connect to TellMe, a North American Microsoft subsidiary. MSN Direct Services can also be enabled, providing a variety of up-to-date broadcast data services.

Human–Machine Interface

Microsoft Auto 4.0 uses two different UI modes: speech interactions and display messages. Applications must ensure that display messages and speech messages are recognized and are appropriately handled. 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 Microsoft Auto application model enables 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. Microsoft Auto 4.0 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 it to the customer in a way that meets the requirements of each individual automaker.

The HMI sample applications that are included in Microsoft Auto 4.0 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 Service

Rich speech experiences:

- Speech recognition and text-to-speech engines from:
  - Nuance
  - SVOX
- Supported with SAPI 5.41
- Pluggable architecture for developer choice of speech engine

The speech service lets many applications share the SR and TTS engines and provides common controls that make it easy for applications to have a rich dialog with the user with minimal application work. The speech-based UI is enabled by the speech service that sits on top of the Microsoft Speech API (SAPI). The speech service can host a variety of SAPI 5.41-compliant SR and TTS engines. The speech service also uses a speech prompt engine, which, when combined with prerecorded and dynamically synthesized prompts, creates a more natural user experience. The speech engines are interchangeable, which allows inexpensive support for several languages.

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 global grammar and arbitrating access to the speech focus.

Language availability is limited only by the SR and TTS vendor's language catalog. There are three voice recognition and TTS engine combinations included in Microsoft Auto 4.0 for development purposes only (two sets are from Nuance [Scansoft] for both SR and TTS; another set comes from SVOX). 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.

Display

The ability to display text, buttons, and pictures on the display screen complements the speech-based interaction with the user—provided that it is accomplished with due consideration to 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).

Microsoft Auto 4.0 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 Microsoft Auto 4.0 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 allows them to write code that is display agnostic, and 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. This 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 it 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. It 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.

Delivering a Microsoft Auto 4.0–Based Solution

Application designers can rely on the rich, familiar development environment that is provided by Microsoft Auto 4.0. For a "future-proof" device, Microsoft Auto 4.0 uses the secure, standard, and stable update and installation technology from Windows Embedded CE 6.0 R2 and Windows Mobile.

Microsoft Auto 4.0 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 necessarily have automotive software development experience).

Microsoft Auto 4.0 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, Microsoft Auto 4.0 enables designs to meet requirements from various testing phases, such as unit testing, functional testing, system-integration testing, and in-vehicle integration testing.

Development tools:

- Platform Builder for Windows Embedded CE 6.0 R2
- Microsoft Visual Studio 2005
- Development hardware
- Microsoft Auto 4.0 Platform Development Kit
- Source code and binaries for the supported processor platforms
- Binaries for Microsoft Auto middleware components
- Documentation
- Sample code
- Command-line tools

Platform Builder for Windows Embedded CE 6.0 R2

Platform Builder is a standard integrated development environment (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 in which a software developer can work on platforms and projects.

Platform Builder ships with Windows Embedded CE 6.0 R2 and runs as an add-on to Visual Studio 2005—providing a more consistent development experience for application developers and platform developers.

Microsoft Auto 4.0 uses Platform Builder primarily in the following scenarios:

- Developing (editing, compiling, and debugging) of native C and C++ 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

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. Microsoft Auto 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 about the Platform Builder catalog, visit the following Web site: http://msdn.microsoft.com/en-us/library/aa913751.aspx.

Platform Customization

Customizing a Windows Embedded CE 6.0 R2–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 the Platform Builder User's Guide, visit the following Web site: http://msdn.microsoft.com/en-us/library/aa913745.aspx.

Visual Studio 2005

Visual Studio 2005, the standard development suite for desktop Windows development, features an extensive set of tools, configuration samples, and guidelines that improve productivity from the initial design to the final testing and tuning. Visual Studio is a complete set of development tools for building Microsoft ASP.NET Web applications, XML Web services, desktop applications, and mobile applications. Microsoft Visual Basic®, Microsoft Visual C++®, Microsoft Visual C#®, and Microsoft Visual J#® all use the same IDE, which allows them to share tools and facilitates in the creation of mixed-language solutions.

Visual Studio 2005 is required for hosting Platform Builder 6.0. Using the Platform Builder mode, the developer can create and debug the actual operating system platform on which the HMI applications will run. Using Visual Studio in application mode enables development and debugging of applications that are running on top of the platform image that is already running on the device.

For more information about Visual Studio, visit the following Web site: http://msdn.microsoft.com/en-us/library/ms950416.aspx.

Development Hardware

The development and testing for Microsoft Auto 4.0 is typically done on prototype hardware rather than through software emulators. The development hardware should have available either a serial port and a USB device or an Ethernet port for connecting to a desktop computer. Microsoft Auto 4.0 does provide a hardware reference design—for details, see Appendix 2: Microsoft Auto Base Components and Hardware Reference Design.

Platform Development Kit

Microsoft Auto 4.0 ships as a PDK that installs on top of Windows Embedded CE 6.0 R2; it is used in conjunction with Visual Studio 2005 and Platform Builder 6.0 for development of the complete software stack for a specific device.

The Microsoft Auto 4.0 PDK includes:

- BSPs for MARPF1, MARPF2, Jacinto EVM, and SDK7785
- Binaries for three processor architectures: ARM, SH4, and x86
- Binaries for the Microsoft Auto middleware components
- Documentation
- The command-line tools necessary to create, modify, and extend a Microsoft Auto 4.0–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: Microsoft Auto 4.0 Samples.

Summary

Microsoft Automotive Infotainment Vision

Enrich the in-vehicle experience with an industry-leading platform for communication, entertainment, navigation, and connected services.

Microsoft Auto 4.0 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.

Microsoft Auto 4.0 also provides new ways to connect a vehicle to services such as local search, weather information, and gas information through Microsoft Auto Services. This makes the exploration of completely new in-vehicle scenarios possible.

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 Microsoft Auto.

Flexible

Microsoft Auto 4.0 provides a flexible, component-based middleware stack with the features that are needed to create an in-vehicle infotainment solution. These middleware components include Bluetooth phone support for hands-free calling, text messaging, and media streaming; USB-based media device integration; AM/FM and digital radio support; CD playback and ripping; connected services; and a speech HMI integration platform. Built on the flexible Windows Embedded CE 6.0 R2 operating system, Microsoft Auto 4.0 offers an open, layered architecture that makes it easy for developers to extend functionality with custom solutions.

This flexibility provides scalability from a cost-optimized basic system to sophisticated three-dimensional navigation systems. Supporting a wide range of processors and system components, a custom hardware solution can be designed to meet your precise needs. And, to speed time to market by letting software and hardware developers work in parallel, Microsoft Auto 4.0 includes a development reference platform and complete BSPs for ARM-based and SH-based processors. To provide an even greater range of scalability, this release now supports Intel architecture processors, including the newest options for the Intel Atom Z5xx series processors.

Microsoft Auto supports many vehicle networking options, including CAN, Ethernet, and MOST. Version 4.0 also adds support for IEEE 1394 (FireWire, i.LINK, and Lynx) device interfaces to enable high-speed audio and video streaming and data communication.

Reliable

Microsoft Auto, used today in a wide variety of OEM vehicles and aftermarket devices, was built and tested by a Microsoft team that has over 10 years of automotive industry experience. It

includes middleware applications, protocols, and services that have been tested together as a complete system. In addition, Microsoft has tested hundreds of mobile phones, media players, and USB devices to help ensure broad market compatibility. The platform is updatable; semiannual device compatibility updates help ensure that infotainment devices remain in sync with consumer needs, and service packs help OEMs address device issues in the field.

To further ensure that solutions remain current, the Microsoft Auto system architecture and tools facilitate easy system updates. These updates can deploy fixes, updates, and new applications and services over the lifetime of vehicle ownership.

Connected

Microsoft Auto 4.0 provides support for Internet-connected services with embedded platform components and for service components that can be used across a variety of embedded platforms (and are not limited to Microsoft Auto–powered systems). Microsoft Auto 4.0 provides expanded support for services, including local search, and richer location-based information, such as weather and gas prices. These service components provide innovative Internet-based features to meet the demands of consumers in an increasingly connected world.

Performance Optimized

Microsoft Auto 4.0 significantly reduces start-up time relative to previous versions. By using a staged driver load design and by starting software from flash memory, the radio receiver, the display, and the cameras become operational no later than 1 second after system power up.

Microsoft Auto 4.0 adds features that are designed to improve the overall car infotainment experience. From best-in-class, software-based acoustic echo cancelling and noise reduction and support for rich speech HMI development, to dedicated support teams for vehicle OEMs and tier 1 suppliers, Microsoft Auto 4.0 offers the best choice for next generation in-vehicle infotainment solutions.

For the latest information, visit Microsoft Auto on the Web:
http://www.microsoft.com/auto/default.mspx

Appendix 1: Microsoft Auto 4.0 Features

Microsoft Auto 4.0, the latest release, includes a host of exciting features, including the following.

Feature	Details
Bluetooth 2.0 + EDR support	• Includes a range of profiles and technologies used for integration of mobile phones and portable media players • A2DP support for the Subband codec (SBC) and MP3 codecs streamed over Bluetooth profiles • Support for the new, high-quality ringtones for Motorola phones • Phonebook download by using PBAP, SyncML, AT commands, and OBEX • Send and receive text messages via AT command • Data connection by using DUN • Twice yearly device compatibility updates
Hands-free phone integration of Bluetooth-enabled mobile phones	• HFP 1.5 and 1.0 support • Access to the contacts on the connected mobile phone • Send and receive SMS messages via AT command • Rich call control and multiple-call scenarios • Compatibility with a wide range of mobile phones • Rich contact synchronization, enabling the use of street addresses, birthdates, e-mail addresses, and contact pictures from the mobile phone
Supported Bluetooth technologies	• Generic Object Exchange Profile (GOEP) 1.1 • OPP 1.1 • SPP 1.1 • PBAP-PCE 1.0 • A2DP-SNK 1.2 • AVRCP-Controller 1.4 • HFP-HF 1.5 (backward compatible to HFP 1.0) • DUN-DT and GW 1.1 • MAP 1.0* * Preliminary implementation of the MAP profile pending finalization by Bluetooth SIG
Integration of portable media players	• Support for a wide range of portable media players (PMPs), including iPod, Zune, and Creative • Support to tag music for purchase on iTunes • Support for video playback on an iPod or iPhone • View album art for music brought into the car • Play media from an inexpensive USB storage device or other removable storage • Indexing and access to the metadata stored in the digital music brought into the car, and support for browsing a device while it is being indexed • Play DRM-protected music from most media players • Support for accelerated audio during fast-forward and rewind operations • Improved application control over decide indexing, allowing better implementations of media storage permanently in the car • Built-in support for CD drives and CD ripping
Support for media streaming in the car	• IEEE 1394 support • Enables automakers to develop navigation
Standard programming interface for controlling the radio in a car	• Support for AM, FM, and HD radio • Data from RDS, Traffic Management Center (TMC) • HD radio support for MPS, SPS, SIS, and PSD • Support for multiple tuners, phase diversity tuning, and scanning antenna diversity • Extensible architecture for additional radio types
In-vehicle device updatability	• DM technology in Microsoft Auto 4.0 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

Rich speech experiences	• Speech recognition and TTS engines from:
	  ○ Nuance
	  ○ SVOX
	• Supported with SAPI 5.3
	• Pluggable architecture for the developer's choice of speech engine

Development tools	• Microsoft Auto 4.0 is built on Windows Embedded CE 6.0 R2 • Integration with Platform Builder and Visual Studio provides a rich development environment
Included board support packages (BSPs)	• Renesas SDK-7785 system development kit • Freescale i.MX31 processor • Texas Instruments Jacinto processor and the EVM (TMS320DRA446) prototyping circuit board
Hardware options	• Support for Intel x86 processors (iA86) • Support for Renesas SH4-based processors (SH7785) • Freescale i.MX 31 and i.MX35 • Texas Instruments Jacinto
Security and reliability features	• Improved parameter validation for system calls • Per-process page and handle tables • Greatly improves process isolation • Improves code robustness • Secure stack • System calls run on special kernel side stacks • Safeguards system calls from stack tampering • Robust heaps • Heap control structures are separated from heap data • Safe remote heaps for operating system components • Operating system servers can open heaps in user process • Read/write for servers, read only for users • Performance optimization and safe from tampering • Device authentication • A service credential object to applications that can be used to authenticate the device to Web services • The Media Access Control (MAC) address of the Bluetooth module is used to derive a “userId” for the service credential object • Bluetooth connection with external devices is made over a secure link • Only the phonebook of the currently connected Bluetooth phone is accessible through the HMI
Boot times measured by Microsoft on the MARFP2 board	• First drivers: 350 msec • Radio: 550 msec • Min shell: 1.2 sec • Full sample apps: ~5 sec
Audio file types	• Playlists: WPL, ASX, M3U, Zune, iPod, and MTP • Media Files: WMA, MP3, MP4, AAC, and WAV • Codecs: WMA, MP3, AAC, and PCM WAV • Extension interface for additional file types
Media support	• Fourth generation and newer iPod, iPhone, and iPod Touch support, one-wire and two-wire • Video browsing and playback supported on iPod and iPhone devices • Zune 8 GB, Zune 30 GB, Zune 80 GB, Zune 120 GB, and devices certified for Windows Vista® (includes most popular players from Creative, SanDisk, Philips, Samsung, Archos, and others) • Album art supported across player types • Support for tagging to purchase music • Support for CDs, data CDs, and DVDs • Support for CD ripping to local storage, including plug-in interface for metadata database

Appendix 2: Microsoft Auto 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 Microsoft Auto 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, as well as by application-level components. The Microsoft Auto DPS implementation is a tool for BSP and application developers to store and access device-specific data.

- **Microsoft Auto IEEE 1394 drivers.** The Microsoft Auto 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.

Hardware Reference Design

Figure 22 shows a schematic for the F2 development platform.

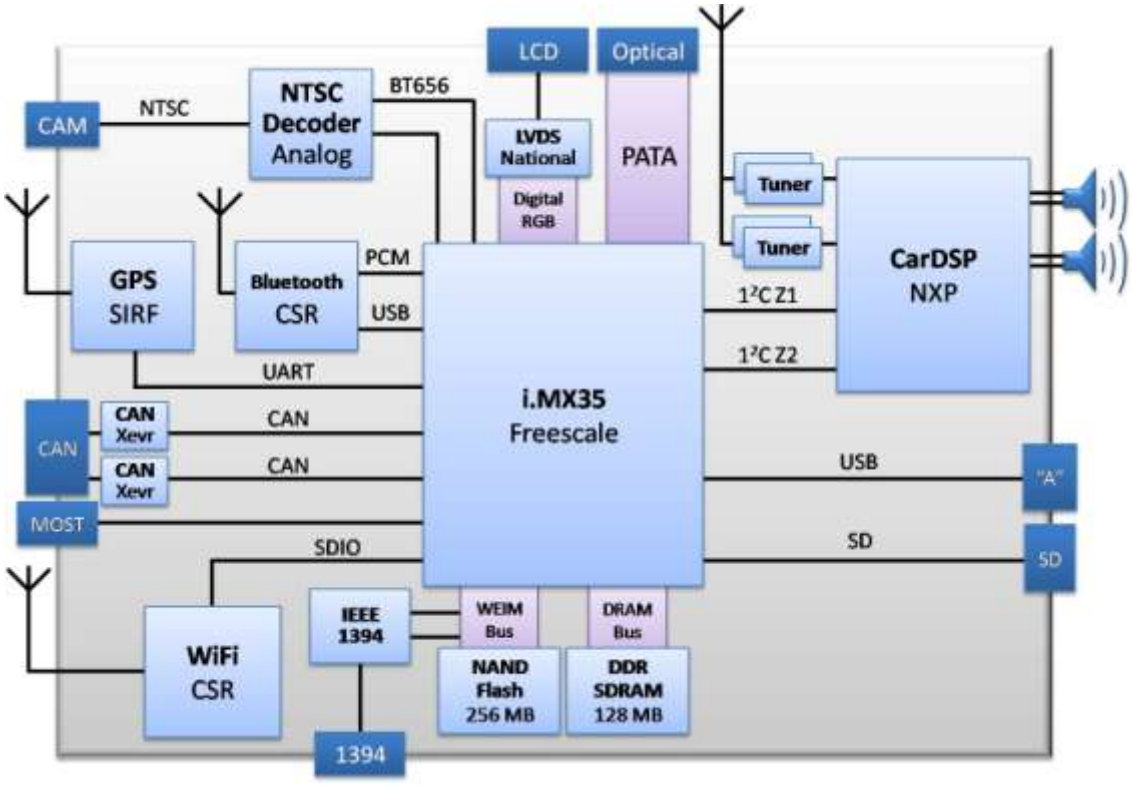


Figure 21. F2 development platform

Table 3 provides details of the Microsoft Auto 4.0 hardware reference design.

Table 3. Hardware reference design

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
	• 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
	• 5-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 2.0+ 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 3: Boot Times

In the Microsoft Auto 4.0 hardware reference design, the software is started from a flash memory instead of from a fixed disk. This greatly reduces the boot time. Figure 23 and Table 4 show the results that Microsoft obtained by using Microsoft Auto development hardware.

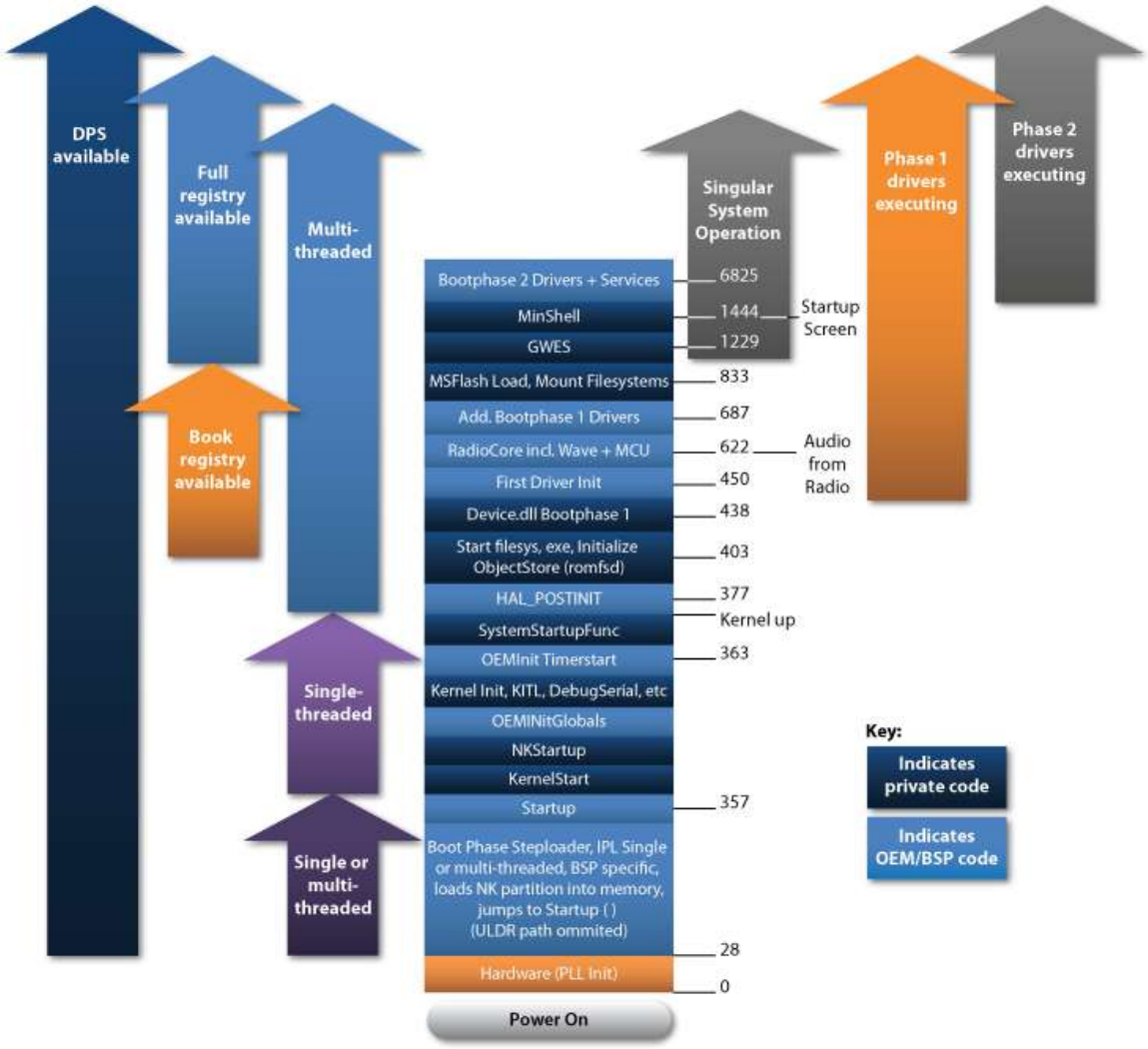


Figure 22. Boot times measured by Microsoft on Microsoft Auto reference hardware

Table 4. Boot times on Microsoft Auto reference hardware

Operation	Time (msec)
Startup, initial program loader (IPL), and NK load*	357
OEMInit timestart	363
HAL_POSTINIT	377
Start Filesys.exe, initialize ObjectStore	403
Radio	680
MS 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 4: Media Device Services

The following table details the media device services.

Table 5. 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 Small, random selection of files available to play while indexing
	Browsing	• Mass storage device (MSD) content is immediately accessible via browser interface • Audio content is immediately accessible via browser interface prior to index completion
	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 msec 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 2.0A and 2.0B authentication chips, 2.0B recommend
		  ○ Partner must acquire authentication chips from Apple

	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 msec per song)
A2DP	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 indexing	• No indexing is supported
CD	Playback	• Redbook Audio and 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 (e.g., 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 • WMA encoder is provided in platform • Encoder can be replaced by a new encoder or an encoder 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 5: Microsoft Auto 4.0 Samples

The Microsoft Auto 4.0 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 6 describes media samples.

Table 6. 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\Autocomp\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\Autocomp\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\Autocomp\Oak\Samples\Mtp\Indexing

Telephony and Data Communications Samples

Table 7 describes telephony and data commnications samples.

Table 7. 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 Microsoft Auto 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 Microsoft Auto 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\Autocomp\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 calling number, the battery level, the signal strength, the phone carrier, the service state, and all call list and phonebook entries.	<*installation_point*>\public\Autocomp\Oak\Samples\Phone\Hfp

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\Autocomp\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\Autocomp\Oak\Samples\Sms\SmsSend

Installer Samples

Table 8 describes installer samples.

Table 8. 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 9 describes an HMI samples.

Table 9. HMI sample

Sample	Description	Location
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 you run this sample application, ensure that the .wav files are also located in the release directory.	*<installation_point>\public\Autocomp\oak\samples\Speech*

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.

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. For more information about alert tokens

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 of it by computer programs.

Application grammars—Grammars that are specified by the application.

Application process—An application that runs on the Microsoft Auto–based device that includes speech as part of its user interaction.

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 Microsoft Auto–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. Then, the responses are converted to AT response codes and are 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 control all of the audio/visual equipment to which a user has access.

Bluetooth wireless technology—An industrial specification for wireless personal area networks, named after the 10th century king of Denmark, King Harold Bluetooth. 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 the only job of which is to load software to start the operating system.

CAB file format—The Microsoft Windows native compressed archive format. It 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 communicate with each other and without a host computer.

Codec—A device or a program that is capable of encoding and decoding a digital data stream or signal. Microsoft Auto 4.0 only provides production-licensed decoders for Windows Media Audio (WMA) and a development license for MP3.

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.

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. DHSP 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.

DirectX®—A collection of APIs for handling tasks that are related to multimedia, especially game programming and video, on Microsoft platforms.

DLL—Dynamic-link library. DLLs are implementations of the shared library concept in the Microsoft 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.

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.

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 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.

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 Microsoft 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.

ICS—Internet Connection Sharing. ICS is the feature of the Windows operating system for sharing a single Internet connection on one computer between other computers on the same local area network. It makes use of DHCP and Network Address Translation (NAT).

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.

`ipconfig`—Internet Protocol Configuration. In Microsoft Windows, `ipconfig` is a console application that displays all current TCP/IP network configuration values and refreshes DHCP and DNS settings.

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 Protocol. L2CAP is a lower connection-based Bluetooth communication protocol that manages the data transport through the logical links in a physical channel. L2CAP does not implement flow control. It relies on a device-to-device baseband link that is provided by Bluetooth hardware.

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 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).

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 that is based on synchronous data communication 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—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 NICs. NDIS is a Logical Link Control (LLC) forms the upper sublayer of the 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.

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

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 (smart phones), Web browsers, or portable media players.

PDK—Platform Development Kit. Refers to a collection of tools, documentation, and code that enables a platform developer (likely a supplier) to 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 can modify an image in order 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 self test the network interface card of the computer or 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.

PPP —Point-to-point protocol. PPP is a member of the TCP/IP protocol suite and 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 will be paused, and the system will switch to listening mode. When the system finishes listening, any process that was paused will resume.

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 Microsoft Auto 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. The process that 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 to enable the use of speech recognition (converts spoken words to machine-readable input) and text-to-speech (the artificial production of human speech) within Windows applications.

SBC—Subband codec. SBC is an audio encoder and decoder to connect Bluetooth high-quality audio devices like headphones or loudspeakers as well as the Internet.

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 (SD) 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.

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 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.

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, FAX, 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.

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 phrases "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 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.

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, designed to enable peripherals to be connected by using a single standardized interface socket and to improve plug-and-play capabilities by letting devices 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 e-mail messages but 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.

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 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.

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 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 Microsoft Windows NT®). Windows Embedded CE 6.0 is supported on Intel x86 (and compatible processors), MIPS, ARM, and Hitachi SuperH processors. Microsoft Auto 4.0 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 that was developed by Microsoft as part of the Windows Media framework. 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.

Related Links

The following Web pages provide additional information.

- For more information about Microsoft Auto 4.0, visit the following Web site: http://www.microsoft.com/auto/default.mspx

- For more information about Windows Embedded CE 6.0 R2, 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 a complete list of the components of Windows Embedded CE, visit the following Web site: http://www.microsoft.com/windowsembedded/en-us/products/windowsce/component-library.mspx

- For more information about the power management architecture of Windows Embedded CE 6.0, visit the following Web site: http://msdn.microsoft.com/en-us/library/aa929261.aspx

---