



Key Feature Details of Windows® Embedded NavReady™ 2009

Published: October 2008

Summary: Windows Embedded NavReady 2009 provides OEMs with the technology to create portable navigation devices (PNDs) with smart, innovative features. This white paper provides technical details about key features present in NavReady 2009.

Copyright

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, Windows and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Contents

Introduction	1
Live Search for Devices	1
MSN Direct Service	4
Connection Manager	5
Bluetooth Profiles	7
ActiveSync Desktop Pass-through (DTPT)	12
Conclusion	13

Introduction

In June 2008, Microsoft released the first version of Windows Embedded NavReady. NavReady includes a number of innovative features for portable navigation devices (PNDs), allowing OEMs to create innovative solutions to meet growing customer needs. This white paper describes the key technologies at the core of the NavReady 2009 release, covering the details of their functionality as well as how a developer can integrate these features when building devices.

Portable navigation devices are becoming more feature rich as the market matures and continues to grow. Microsoft has recognized this demand by introducing NavReady 2009 based on Windows Embedded CE 5.0. OEMs are well poised to create enhanced offerings with Live Search, Bluetooth enabled handsfree voice calls and audio streaming, as well as desktop connectivity. NavReady's powerful architecture and features provide a service layer to applications via well documented APIs. In order to utilize NavReady, an OEM will need to include the desired components in the platform configuration and develop support for the features into the product's applications.

Live Search for Devices

The Live Search for Devices (LS4D) feature enables a PND to provide the user with up-to-date, location-aware search results. As the name implies, LS4D utilizes Microsoft's Live Search service to capture results from online sources. This functionality allows a PND to search for businesses, people, or points-of-interest (POI). OEMs can utilize the search results to provide relevant information to consumers, such as directions to destinations, ratings on businesses, or additional value-added features within the navigation application.

The Microsoft Live Search server uses a SOAP API for normal application requests. SOAP is a standard communication protocol used to exchange XML-based message packets within HTTP requests. The protocol is extremely flexible and robust, but that comes at the cost of size and complexity. Windows Embedded CE devices, like all portable devices, are more resource constrained than workstation computers, particularly for network capacity and processing power. For this reason, LS4D utilizes a custom protocol on top of HTTP to support the needs of PNDs that significantly reduces application overhead compared to SOAP. The LS4D protocol encapsulates the functionality needed by PNDs, to reduce the bandwidth requirements by approximately 80 percent compared to the Live Search native SOAP interface. The LS4D architecture is made up of three major layers: the portable navigation device, the LS4D Web server, and the Live Search server, as shown in the following figure.

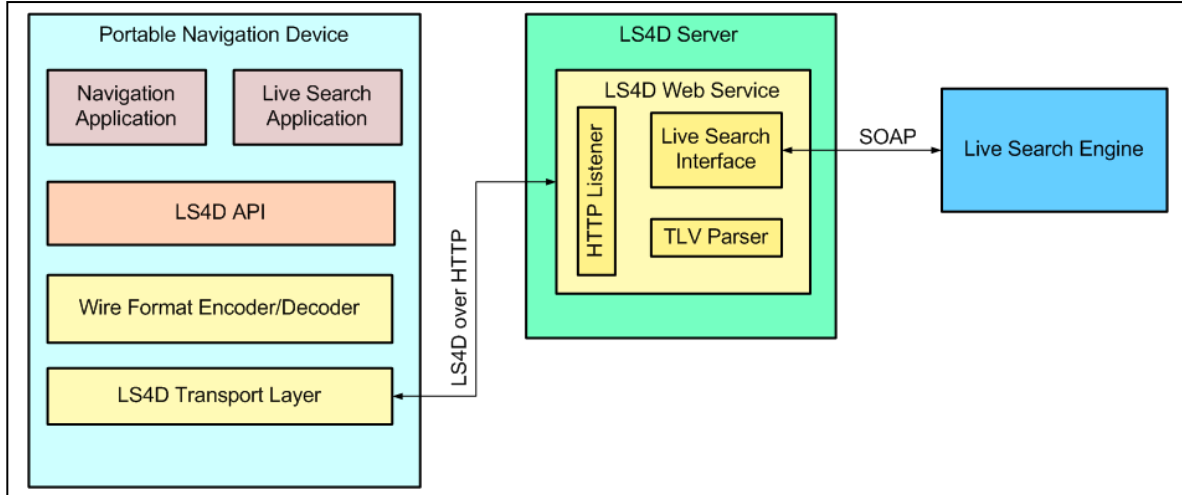


Figure 1 – LS4D System Architecture

The LS4D protocol includes security support, allowing the LS4D server to log and monitor devices that utilize LS4D. This is accomplished by using a device-specific identifier and a device-OEM identifier, respectively called the "Device ID" and the "Device Group ID". Each LS4D request includes this information. The LS4D Web server will then deny the request or let it pass through to the Live Search servers based on the specific device identifier. The LS4D server logs all search requests, making it simple to identify large numbers of devices from the same OEM, as well as analyze loading information for system planning.

LS4D allows users to find people, businesses, or POI using broad or specific criteria. Search types can be categorized as: standard keyword, advanced keywords and search operators (filtered from a specific source), or by category. These search types allow the PND vendor to create simple or complex search interfaces. NavReady does not ship with a user interface (UI) for LS4D. The OEM must create a UI that integrates with the device-specific UI and that meets the Microsoft Live Search branding requirements. For example, a custom Live Search UI might present the user with a category search to allow users to refine their searches, if desired, as shown in the following figure.

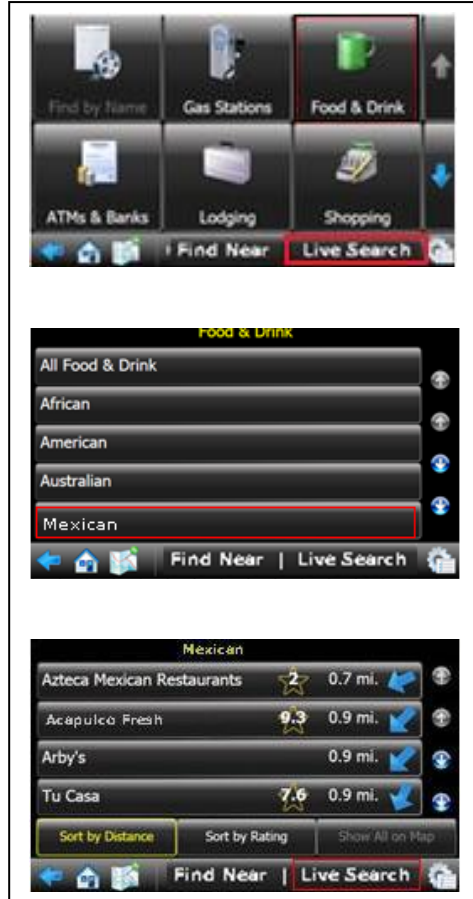


Figure 2 – Example UI Showing Category Search and Results

The LS4D API is a simple yet powerful toolset for the application writer. As mentioned above, the application writer has the ability to search using broad or specific items including name, zip code, city name, keywords, and points of interest, such as a landmark or national park. The results generated by Live Search can be further refined with the API by categorizing the search results based on distance, relevance or radius of search. Currently, the API does not provide support to sort the results based on user ratings, alphabetically, or other unique sorting requirements, as it is up to the vendor to provide this functionality. Additionally, vendors can and should support the ability to filter the results based on the source of the data: White Pages or Yellow Pages.

LS4D can be added to existing OS designs by including the “Live Search for Devices (LS4D)” component from the Windows CE 5.0 Platform Builder Catalog. Additionally, the device must have a unique Device ID and an OEM unique Device Group ID. The OEM must register with Microsoft to obtain a Device Group ID and store it in the run-time image of the device. Similarly, the Device ID must also be stored across reboots, requiring persistent registry in order to support LS4D. OEMs have a number of options for generating the Device ID for each individual PND, as follows:

- Use a hardware identifier by implementing IOCTL_HAL_GET_UUID. The GUID returned by this IOCTL must uniquely identify the device using a hardware identifier such as flash or processor serial number. Note that this ID only has to be unique for all devices of a particular Device Group ID.
- Use an OEM-specific function call to generate the identifier. This function must be called at first boot and the resulting ID gets stored in the persistent registry.
- Use a software-generated identifier created by using a LS4D function call. If no unique hardware device is available, IOCTL_HAL_GET_UUID should return a NULL GUID. When LS4D is first utilized it will query IOCTL_HAL_GET_UUID and if a NULL GUID is returned it will look in the registry for a stored identifier. If no identifier is present, it will generate a new identifier and store it.

See this Microsoft Web site for further details:

<http://msdn.microsoft.com/en-us/library/cc510790.aspx>

PND applications included on the platform can now use the LS4D APIs for location aware online services. Note that LS4D requires some form of Internet connectivity. Hence, additional components such as the Bluetooth Dial-up Networking (DUN) profile or Wi-Fi support will also need to be added to the platform.

MSN Direct Service

In addition to LS4D, Microsoft's MSN Direct Service is available as a NavReady 2009 component, allowing the PND to provide users with locale-specific, up-to-date data such as traffic reports, gas prices, weather reports and forecasts, movie times, local events, stock quotes, news headlines, emergency alerts, and received locations sent from the desktop. MSN Direct is currently available in over 125 major regions in the U.S. and Canada. MSN Direct data is distributed via a special sub-carrier band in the FM channel and is transmitted through FM radio towers, as shown in the following figure.

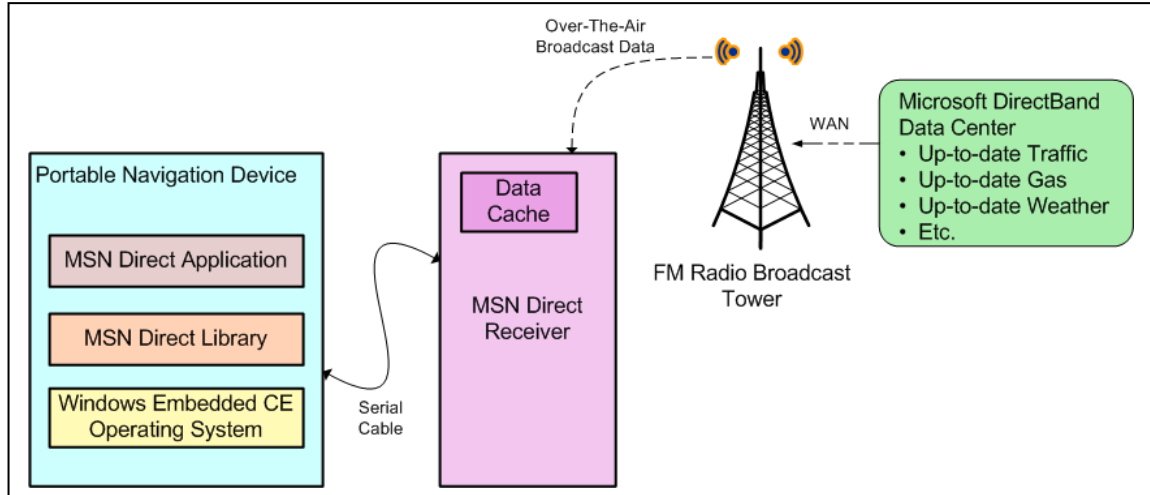


Figure 3 – MSN Direct Service High Level Architecture

Information available from the MSN Direct Service can greatly enhance consumer satisfaction on PNDs. Instead of giving the fastest or shortest route to a destination, the PND's navigation engine can provide up-to-the-minute traffic conditions and alter the driving route, if necessary, in real-time. The various types of information broadcast by the MSN Direct Service provide new opportunities to aid and enhance a user's driving and navigation experience.

See this Microsoft Web site for further details:

<http://msdn.microsoft.com/en-us/library/cc510527.aspx>

Connection Manager

The addition of location-based services, such as LS4D and other new-data connectivity features, presents a challenge for the platform developer. Managing the different connections via the PND application would be cumbersome and error prone without some central coordination. NavReady provides this functionality via the Connection Manager. Connection Manager supports Bluetooth Hands-Free Profile (HFP), Bluetooth Dial-up Networking (DUN), and ActiveSync Desktop Pass-through (DTPT).

The Connection Manager is made up of an API for other applications, the Connection Manager application, and the Connection Planner DLL that coordinate device connectivity for the PND applications, and the connection service-provider DLLs. The applications are not aware of the specific service provider in use, or its configuration; it is completely transparent. This architecture, illustrated in the following figure, allows for easier application development as well as optimal resource utilization.

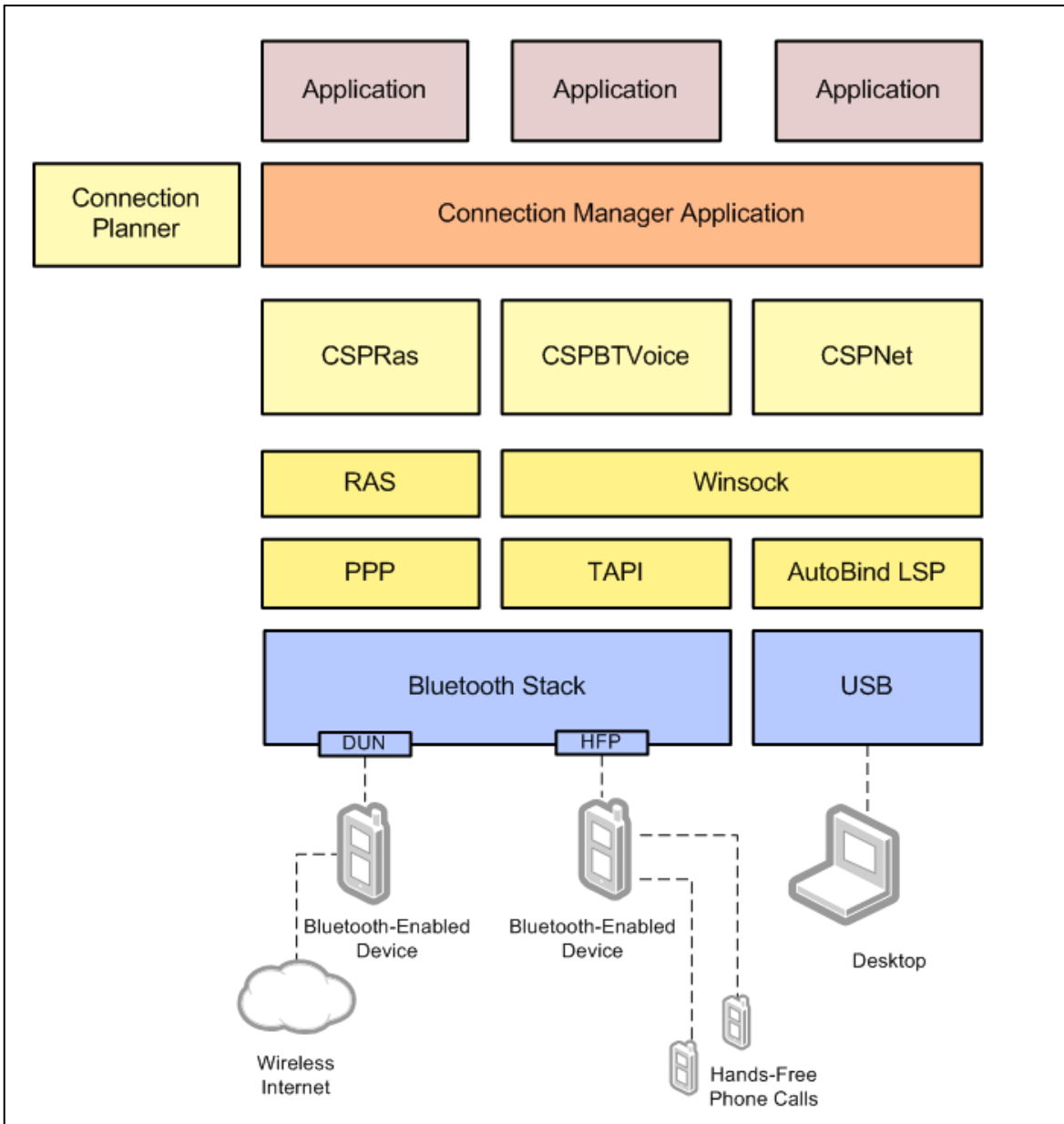


Figure 4 – Connection Manager Software Architecture

The Connection Manager application is the core of the Connection Manager architecture. It interacts with the PND applications, the Connection Planner DLL and the Connection Service Providers present on the system. Applications make a request to the Connection Manager for a specific network type. This can be Internet connectivity, corporate LAN, or WAP network, depending on the application. Connection Manager then processes the request utilizing the Connection Planner to determine the best service provider to use. At the time scheduled via the Connection Planner, Connection Manager will use the specific service provider to establish the connection and notify the application(s) of the

connection status. Further, Connection Manager takes a proactive approach to managing connections by disconnecting idle connections after a specified period of time. Connections can also be requested to be “always on”; maintaining the PND’s packet-data connection after the initial connection has been made. Connection Manager will maintain and recover connections marked always on, if necessary. Most connections default to “not always on”. The exception is the DTPT connection which defaults to always on and cannot be changed.

The Connection Planner uses information from the application request and service-provider characteristics to determine the ideal connection path to use. Application requests contain network type, connection interval, security, and priority information. This information is critical for Connection Planner as it is used to determine the correct network service provider to use, to determine any connection optimization that might be needed, and to determine whether any existing connection needs to be preempted due to priority. The service provider is selected based on network type, cost, latency, and bandwidth of each path. Optimizations of the connections are a key feature of the Connection Planner, reducing the total use of service-provider resources as well as allowing use of the same connection by multiple applications simultaneously. All applications for a given network type will be notified when the connection is established and the connection itself can be scheduled such that it meets the interval requirements of each application.

Connection service providers report critical information about the connection to Connection Manager, such as the latency and cost. Certain service providers also work with Connection Manager to make the connections, when requested. This is done via the NDIS User-mode I/O (NDISUIO) driver with the service provider binding to it when connected. This architecture allows the Connection Manager to make informed decisions about the best path to select and abstract the hardware details into the service-provider DLLs.

The Connection Manager can be added to a platform via the Catalog. Since it is dependent on the various service providers, these must be added from the Catalog as well. The Connection Manager application does not include a user interface (UI) for configuration by the user. Instead, configuration is done using an XML provisioning file located on the PND. However, it is possible to alter the configuration programmatically using a published API. This allows the OEM to create a customized UI for setting configuration information after devices are released from production.

Bluetooth Profiles

Bluetooth support in Windows Embedded CE was first introduced in Windows CE .NET v4.1. Since that time Microsoft has continued to enhance and expand Bluetooth support for Windows Embedded CE by including additional profiles in new versions and feature packs for Windows Embedded CE. A Bluetooth profile contains the support for a particular protocol over the Bluetooth wireless transport. NavReady further adds to the Windows Embedded CE 5.0 Bluetooth offerings with the introduction of five new profiles

targeting PNDs: Hands-Free Profile (HFP), Dial-up Networking (DUN), Phone Book Access Profile (PBAP), Advanced Audio Distribution Profile (A2DP), and Audio/Video Remote Control Profile (AVRCP). Each of these profiles provides a critical feature for PNDs to help OEMs differentiate their products in the market.

Before the NavReady Bluetooth profiles can be used, the PND must be paired with the other devices supporting the specific feature, the same as with other profiles. For example, in order to use the Bluetooth HFP, the PND must be paired with a mobile phone. Pairing is the standard way of establishing a communication link between two Bluetooth devices. Pairing in Windows Embedded CE is performed using the Bluetooth Pairing Service. The service maintains records of the paired devices for use by the Bluetooth stack and applications. The Bluetooth Pairing Service is middleware; it does not provide a user interface for its operation. The PND application must include an application that interfaces with the Bluetooth Pairing Service to discover and pair itself with other devices. A high level example of this layout is shown in the following figure.

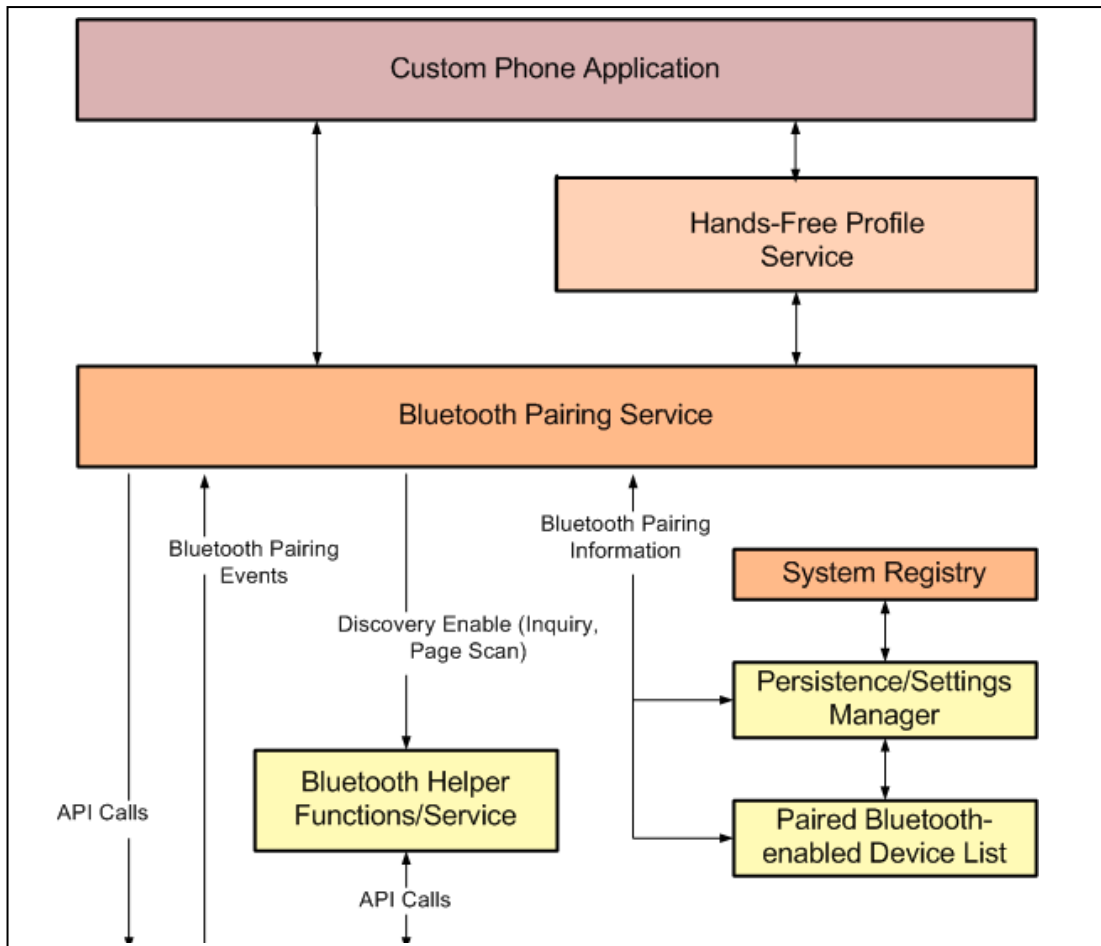


Figure 5 – Bluetooth Pairing Software Architecture

The HFP allows users to make and receive voice calls using a paired mobile phone supporting hands-free operation. This functionality is desirable for PNDs in an automotive setting where users might want to receive voice calls while driving, but due to safety concerns or local laws must use a hands-free device. The HFP can support making and receiving calls as well as reporting phone-signal level, reporting battery level, providing caller-ID information, switching between active and on-hold calls, and manipulating phone-book records. Some of these functions are provided by PBAP, which works closely with HFP. A PND vendor wanting to include HFP support must implement the necessary UI components to interact with and utilize HFP's capability, as shown in the following figure.

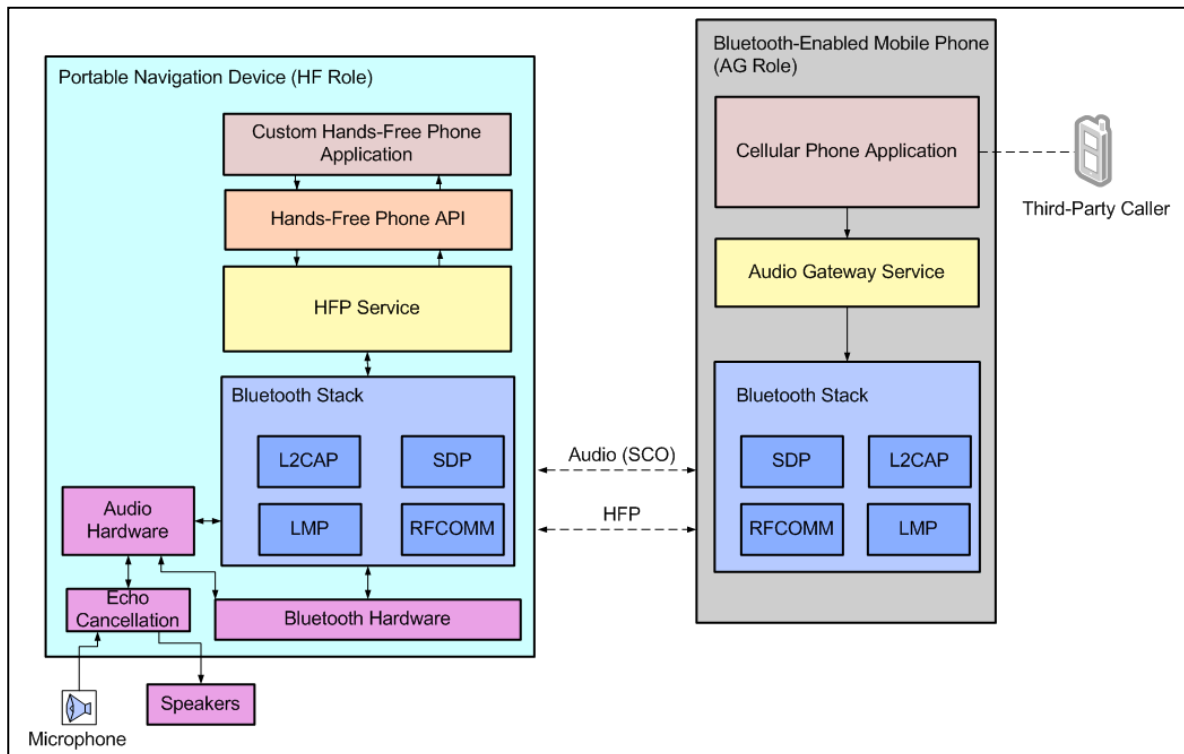


Figure 6 – Bluetooth HFP High Level Architecture

The UI interfaces with and drives HFP using a well documented API. HFP connections also communicate with NavReady Connection Manager, allowing Connection Manager to intelligently select and control Bluetooth DUN connections by using the same paired mobile device as HFP.

The Bluetooth DUN profile, like the HFP, provides a new mechanism for PND connectivity for data connections. The DUN profile can have two different roles depending on the device in which it is deployed. The gateway device is responsible for powering the wireless modem in order to achieve network connectivity. The Data Terminal (DT) role resides on the device wanting to use the network connection. PNDs play the role of the DT when using DUN, as shown in the following figure.

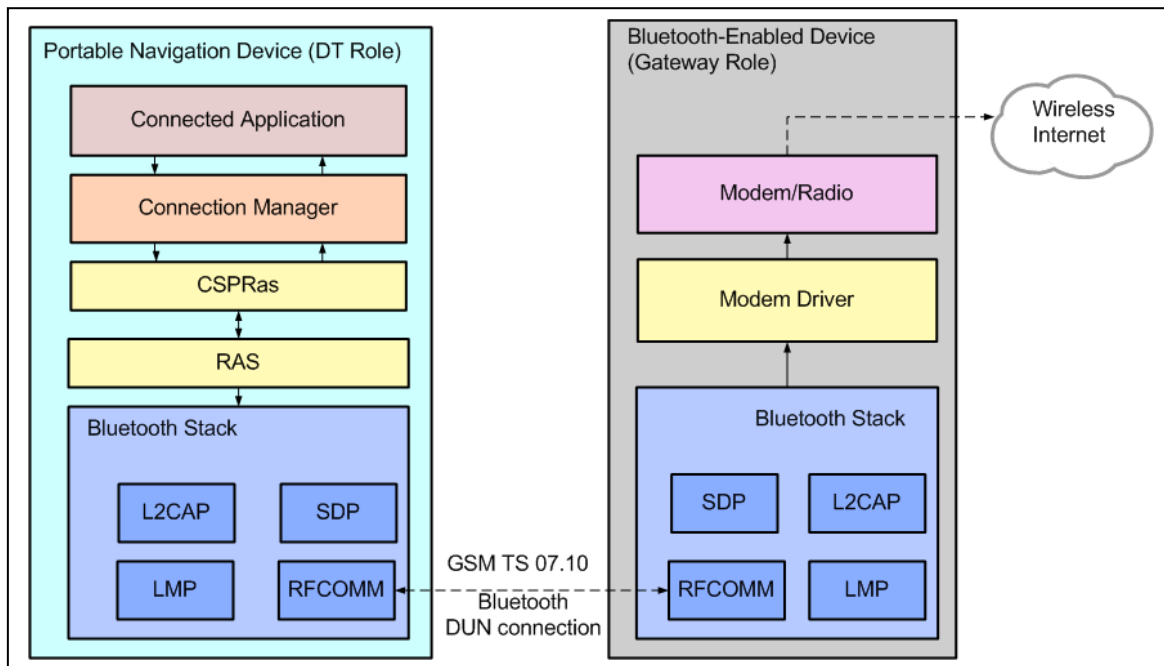


Figure 7 – Bluetooth DUN High Level Architecture

DUN works in conjunction with NavReady's Connection Manager to provide network access to the PND for performing Live Search or other Internet dependent operations. When Connection Manager is used to connect and disconnect to the network, the Bluetooth link to the paired wireless-modem device is automatically activated. The DUN component includes the Connection Manager Service provider for RAS, which drives the DUN Bluetooth profile. Alternatively, the Bluetooth DUN profile can be used directly by applications. However, this requires the PND application to activate the device as well as manage the connection, potentially affecting resource optimization.

Using PBAP, a PND application can leverage phone and address information stored in the user's mobile phone to place calls with the HFP. As previously mentioned, the PBAP is available to PND applications via the HFP API, as shown in the following figure.

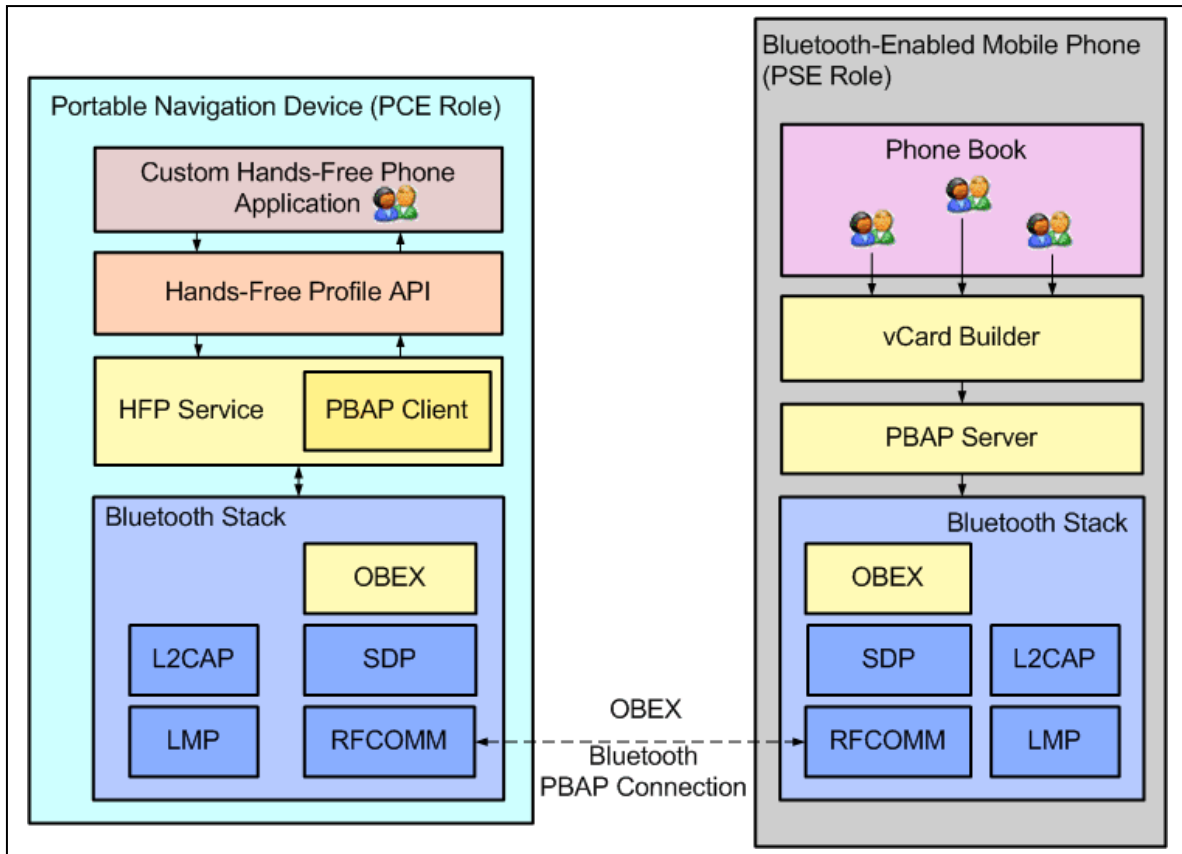


Figure 8 – Bluetooth PBAP High Level Architecture

PBAP utilizes the Object Exchange Protocol (OBEX) to retrieve vCards containing contact information and call history. PBAP supports vCard format version 2.1, which defines the size and format of the data. Each contact can have up to four phone numbers, each of a different type. Contact information, once synchronized to the PND by using PBAP, can be retained in persistent storage. Downloading, or synchronizing, contact information using PBAP can be done manually or automatically giving the OEM and user more refined control of the behavior. Manual download requires OEM UI support, whereas automatic download requires no special UI interaction. Download support of contacts using PBAP is performed on the entire phone book on the mobile device rather than for individual contacts. Using PBAP, PNDs can stay synchronized with a paired mobile device's contact list. Contact data itself cannot be changed, but contacts can be deleted from the PND.

Rounding out the NavReady Bluetooth offerings are two profiles that can work together to stream multimedia content from the PND to a paired device and control the PND's multimedia playback from a paired device. Using A2DP, the PND can stream stereo-audio content to a paired device, such as an in-car stereo. To control the playback of the audio content, AVRCP can be used with a paired device. The relationship between A2DP and AVRCP is shown, at a high level in the following figure.

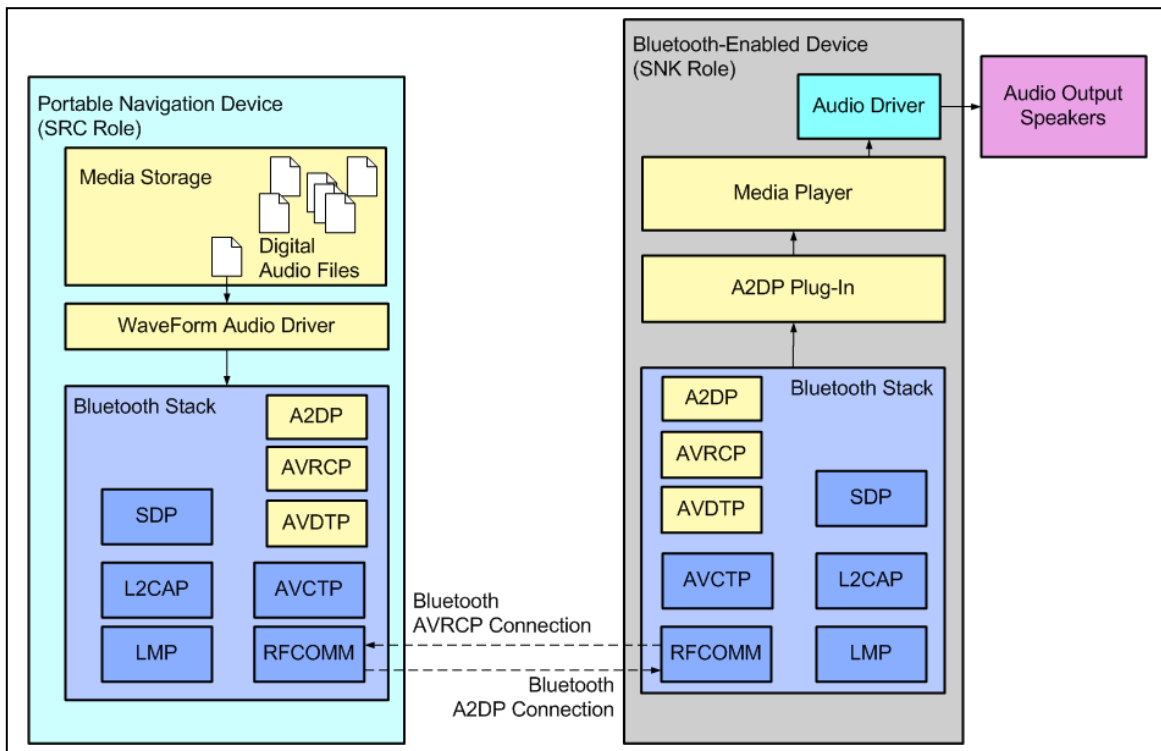


Figure 9 – A2DP and AVRCP High Level Architecture

The PND utilizes locally stored digital-audio files for the source of the streamed content. By using locally stored audio content, the PND does not have to have network connectivity to source the audio content. Additionally, this allows the PND to leverage synchronization with a desktop PC, via ActiveSync Desktop Pass-through (DTPT), in order to keep up-to-date playlists on the device. A2DP also supports audio compression via a built-in Audio Compression Manager (ACM). This allows the PND to support a variety of compressed audio formats supported by different devices, minimizing the bandwidth requirements of the Bluetooth link. AVRCP requires the PND application to support commands from the remote device. Additionally, A2DP and AVRCP include support for simultaneous use of the paired device with HFP. Incoming calls arriving through an HFP connection to the same device as A2DP will cause the A2DP stream to be suspended until the call is completed.

ActiveSync Desktop Pass-through (DTPT)

The multitude ways of using a PND are seemingly endless, as are the different functions users have come to expect. As we have seen so far, NavReady includes a number of new features for providing users with mobile, informative, connected information and entertainment. However, Microsoft has not stopped at mobile connectivity with NavReady. In addition, users can now use ActiveSync with their desktop PCs to enable desktop pass-through network and file connectivity.

ActiveSync technology allows Windows Embedded CE devices to connect to a host PC in order to share information and exchange data. DTPT utilizes ActiveSync and the transport mechanism for TCP/IPv4 traffic between the PND and the desktop PC. The PND utilizes a DTPT client, which handles forwarding of network requests, name service-provide requests, and network traffic. The desktop PC runs the DTPT server, which acts as a network proxy, connecting the DTPT client to external networks, as well as handling NSP requests sent by the DTPT client.

DTPT also provides the PND and desktop with full inter-connectivity. This can be used beyond network traffic for exchanging files, such as digital audio. The data-channel setup between the DTPT client and server is secure by design (connections configured by the developer as unsecure are not supported). While this provides a secure link between the PND and the desktop, application developers must be aware that network traffic is forwarded at the desktop to a network device connected to an external network without the network data being encrypted.

The NavReady Connection Manager can be used to setup and control DTPT connections without interaction from the user. To the application, the connection will look like any other TCP/IPv4 network link. This flexibility allows the application developer to leverage DTPT when available, as well as other connection technologies such Bluetooth DUN.

DTPT is a powerful NavReady tool, but it does have some restrictions. Due to security concerns, the DTPT connection on the desktop or the PND should never be bridged with other connections or used for Internet Connection Sharing (ICS). The TCP/IP address of the DTPT connection is managed by DTPT and should not be changed by applications or the user. To maintain reasonable resource utilization, up to 1000 concurrent requests can be outstanding to the desktop. Even with these minor restrictions, DTPT provides a high-speed connection mechanism to the PND, allowing it to take advantage of potentially higher speeds and reducing the cost of network access.

Conclusion

Windows Embedded NavReady 2009 includes a number of new features customers expect from their PNDs. The NavReady components are also well architected for easier adoption by OEMs. Advanced features, such as Live Search and Bluetooth hands-free voice calling can be tailored to the OEM's custom framework and UI theme using well defined APIs. There is no doubt that PND vendors will find new and creative ways to use NavReady's features to supply their customers with easy to use, robust platforms to guide them wherever their journey begins and ends.

For more information:

Windows Embedded Web site:

<http://www.microsoft.com/windows/embedded/default.aspx>

Windows Embedded NavReady 2009 Web site:

<http://msdn.microsoft.com/en-us/library/cc510895.aspx>