The AMIC Host and Vehicle Services APIs

Edward Nelson
Ford Research Laboratory
September 2002
Requirements for entertainment systems, information systems are in scope.
- Audio system, Rear seat entertainment
- Navigation, Telematics
- Phone interface, mobile device interface

Vehicle control modules (brakes, powertrain, suspension, door/window control,…) are out of scope
- Specifications will not be used by OEMs to define vehicle control systems, such as cruise control
- They will not be used for the primary driver interface (instrument panel, etc)

AMIC will define a uniform interface to the control modules for access by information systems
- I.e., specifications will not be used to define vehicle control systems, but will provide access to those systems for entertainment and information devices in the vehicle.
Within this architectural framework, AMIC is defining specifications on three levels

- **Physical level**

- **Network protocol level**
  - Three network technologies are supported in Release 2
    - MOST
    - 1394
    - Bluetooth
  - AMIC has defined a Common message set on all three networks
    - Defines supported functionality
    - Specified in ASN.1

- **Software level**
  - Java APIs for platform and operating system independence
Role of host platform

- AMIC defines a *host* as any device on the multimedia network that executes non-embedded software.
- AMIC will not specify a single platform (processor, operating system) for the host.
- AMIC will specify an environment that allows applications to be written to run on a variety of vehicles containing different host platforms.
- Java was chosen as the interface language because of its support for platform independence.
Interoperability across multiple platforms is achieved by providing an OSGi framework on a Java runtime environment.

- **Core Software Services**
  - SW exec
  - Persist Store
  - VSI
  - ...
  - Serv Disc

- **Optional Services**
  - Navi
  - ...
  - Addr Book

- **Applications**

- **OSGi Framework**

- **Java Runtime Environment**

- **Operating System**

- **Common Message Set**

- **Low-level Network Drivers**
The interfaces being defined by AMIC can be grouped into three categories:

- Interfaces to services that are local to the host platform
- Interfaces to services provided across the in-vehicle network
- Interfaces to application domain specific services

Interfaces to network services must correspond to those defined at the network level by the common message set.

However, they are not specific to a particular network technology.

- Should be possible to write applications that use network based services and run on any of the AMIC supported networks.
Areas covered by host service APIs

- Software Execution
- Software Lifecycle Management
- Resource Management
- Service Discovery
- Internationalization
- Persistent Storage
- Security
- Off Vehicle Communication
APIs for Services Provided On Network

- Vehicle Services
- Human Machine Interface
- Audio/Video Services

The vehicle services APIs will be discussed in this presentation
APIs for the following services will be specified in this release of the AMIC specifications:

- Navigation
- Telephony
- Location Services
- Address Book

Additional application specific services will be addressed in Phase 3 of the AMIC specifications.
## Host Service APIs

<table>
<thead>
<tr>
<th>Category</th>
<th>APIs</th>
</tr>
</thead>
</table>
| Software Execution              | ● OSGi Framework  
                                | ● AMIC VI System management                                      |
| Software Lifecycle Management   | ● OSGi Framework  
                                | ● OSGi Package Administration Service  
                                | ● OSGi Service Tracker                                              |
| Resource Management             | ● OSGi Framework                                                   |
| Service Discovery               | ● OSGi Device Access Specification  
<pre><code>                            | ● AMIC Network Service Discovery                                  |
</code></pre>
<p>| Internationalization           | ● J2ME CDC Internationalization API                                |</p>
<table>
<thead>
<tr>
<th>Host Service APIs (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Persistent Storage</strong></td>
</tr>
<tr>
<td>- OSGi Framework</td>
</tr>
<tr>
<td><strong>Security</strong></td>
</tr>
<tr>
<td>- J2ME Permissions</td>
</tr>
<tr>
<td>- OSGi Permissions Admin Service</td>
</tr>
<tr>
<td>- OSGi User Administration Service</td>
</tr>
<tr>
<td><strong>User Profile</strong></td>
</tr>
<tr>
<td>- OSGi Preference Service</td>
</tr>
<tr>
<td><strong>Off Vehicle Communication</strong></td>
</tr>
<tr>
<td>- TBD</td>
</tr>
</tbody>
</table>
An interface between the multimedia/information networks and the vehicle controller networks

May be implemented as a separate component or integrated with another component, either on the multimedia side or on the vehicle side.
The vehicle interface provides a standard set of services across all vehicles
- Allows applications to use vehicle services without regard to details of vehicle implementation
- For example, vehicle speed could be provided by several types of sensor from wheel rotation counter to a GPS receiver, but application does not need to be concerned with sensor types.

It also provides isolation between the multimedia network and the vehicle controller networks.
- No direct access to devices on the controller network
- Allows the implementation of OEM specific policies with respect to the use of vehicle information and services.
Vehicle Services APIs

- Provide a Java interface to vehicle services
- Mirror the functionality available in the Common Message Set
- Must be implemented in every AMIC host.
- Provides vehicle specific information to services running on the host.
- Implementation of these APIs may be vehicle independent or vehicle specific.
The services defined by the AMIC vehicle interface APIs correspond to the functionality provided by the vehicle interface messages in the AMIC common message set.

These in turn correspond to the functionality defined in the MOST “function block vehicle” specification.

We are working with Bluetooth and 1394 groups to provide specifications for the same functionality on those networks as well.
Vehicle Services APIs

- **Organized into two Java packages:**
  - `org.amic.vehicle.description`
    - Contains static classes
    - Vehicle identification information (VIN, Make/Model …)
    - Description of vehicle features (engine, seats, doors …)
    - Must be implemented in every host – correspond to VSI core
  - `org.amic.vehicle.services`
    - Interfaces to vehicle information
      - Vehicle speed, Odometer reading, Airbag status, …
    - Interfaces to vehicle controls
      - Door lock control, mirror control, …
    - May be optionally implemented on any given vehicle
    - May also be implemented as a Java interface to CMS (no knowledge of specific vehicle used in implementation)
**Classes**

- **VIN**
  - Provides the vehicle identification number (VIN).
- **ModelInfo**
  - Provides the manufacturer, model and model year.
- **DateOfManufacture**
  - Provides actual date and location of manufacture.
- **Version**
  - Returns version of AMIC specs implemented by VSI
- **VehicleDescription**
  - Provides information on vehicle features, including engine type, doors, fuel capacity and type, mirrors, seats, anti-theft, ABS, etc.
public class ModelInfo
{
    private String Manufacturer;
    private String Model;
    private int Year;
    public String getManufacturer()
    {
        return Manufacturer;
    }
    public String getModel()
    {
        return Model;
    }
    public int getYear()
    {
        return Year;
    }
}
Comprises body and powertrain services of the VSI

Examples:
- Engine speed
- Current gear
- Oil pressure, temperature, level
- Engine coolant
- Cruise control state
- ABS system state
- Airbags
- Door locks
- Lights
- Seat position
- Windshield wipers
- Fuel level
- Rain sensor
- Mirrors
- Odometer reading
public interface EngineOil
{

    /* This method returns the engine oil level as a percent of full capacity. */
    public float getLevel();

    /* This method returns the engine oil pressure state. Possible values of the state are low and normal. The method returns true if the pressure is normal. */
    public boolean isNormalPressure();

    /* This method returns the engine oil temperature. */
    public float getTemp();
}
Current status

- Roughly 60 services are included in the vehicle interface specifications, including powertrain functions, vehicle status information, interior
- Does not cover audio functions; these are part of the audio/video application APIs
- System management is also a responsibility of the Vehicle interface module, but is not part of the vehicle services interface.
- Reference implementation has been constructed and validation is being carried out
- Internal review by AMIC member companies is in process
- Target date for publication of the specifications is Dec 2002