Abstract — The goal of this paper is to use the Platform as a Service (PaaS) methodology and to utilize the cloud as a storage mechanism for personalization data that is commonly stored in consumer electronic devices such as radios, televisions, and other devices that commonly store presets or other configuration data. In generating the application for this paper, several considerations for cloud based application development and deployment are identified, and should be considered for future cloud-based software developers and companies wishing to deploy such solutions.

Index Terms—Cloud Computing, PaaS, Personalization, Cloud Storage, Arduino, Automotive Electronic Module, Netbook

I. INTRODUCTION

The cloud-based storage of application data is becoming more commonplace as connectivity to the Internet is becoming ubiquitous. The ability to serve applications and their data from the cloud allows for scalable applications, with the ability to easily remedy any issues found in the field at one central location. A radio interface with six presets was selected as an application for this paper. PaaS is a category of cloud computing services that provide a computing platform and a solution stack as a service [1]. By utilizing the ‘cloud’ as a storage mechanism for the preset data, the cloud can be utilized to provide a person with the ability to retrieve these presets on any radio enabled with the cloud services for the application. The closest mention of any such project in any publication was for a cloud-based radio platform focused on collecting data from the web, simplifying it, and making it consumable while driving [2]. It however does not match the capabilities demonstrated in this paper.

In the case of a person that travels often, the task of finding and storing their favorite satellite radio stations when they get into their rental vehicle is surely a process that they would rather not have to do. Using a cloud based solution, the accessing and storage of the person’s favorite presets would be as simple as providing their account credentials, and acknowledging a request to update. Using the cloud for such personalization data could be used for a variety of configuration parameters that span industries including automotive, consumer electronics, heating ventilation and air conditioning, and more. In researching the use of the cloud for such applications, no application specific articles are known to have been published.

The hardware environment chosen for the Cloud Based Module Configuration project was a netbook computer using 64-bit version of Windows 7, with 2GB RAM, using an AMD C-50 processor running at 1GHz.

The paper uses Microsoft Expression Blend 4 for creating the user interface, Visual C# for the application behind the interface, and Microsoft Azure SQL for the database that stores the configuration data for the given component.

This paper will describe the issues, reasoning, results, and suggestions for future consideration for other application developers seeking to develop cloud-based solutions for module configuration of electronic modules.

II. CLOUD BASED HARDWARE ALTERNATIVES

The CLOUD BASED MODULE CONFIGURATION paper’s scope originally included three potential candidates to select from. The first was an automotive electronic module that can assume configuration parameters through traditional configuration methodologies used in industry. The second was the use of Arduino hardware to simulate configuration parameters (abstracting the parameters from the physical components) in order to prove the concept is valid and applicable to a wide array of applications. The third was netbook hardware to simulate a variation of hardware (automotive, traditional consume electronic, avionic, etc). Each of the candidates was reviewed for their ability to be developed within the scope for the project. In evaluating each of the candidates for the project, the architectures for each alternative was identified, providing for the scope or each alternative to be compared.

The findings for each alternative for the paper were:

A. Automotive Electronic Module(s)

The electronic module considered for configuration was a body control module that is used in automobiles to control vehicle features including exterior lighting, interior lighting.
vehicle alarm, locking, unlocking and other various functionality that vary depending on the features in a given vehicle [3]. In order to pursue configuration of this module, the interface requires that:

1) **Power (12V DC)** be provided and maintained for the duration of the configuration/communication.
2) **Controller Area Network (CAN) communication protocol** be used in order to communicate with the module.
3) **Use of proprietary security algorithm** for negotiating access to configuration related data.
4) **Design of a tool** that can translate data stored in the ‘Cloud,’ that can interface with CAN hardware that will work with the vehicle module network.

The benefits of using the body control module for the paper would be the demonstrated ability of having vehicle level configuration data that could store a setting for what types of headlights are used on the vehicle, how many doors a vehicle has, or how long a vehicle should wait before turning off its exterior lights.

As a proof of concept, this example would illustrate the capability of interfacing data stored in the cloud (whether it is Original Equipment Manufacturer configuration or customer preference configuration) to a vehicle.

This methodology, could allow for customers (backyard mechanics) to replace modules in their own vehicles, using the cloud to configure them to their original state provided by the auto manufacturer. Or, if customers had certain preferences that they are known to enjoy in any vehicle that they drive (e.g. radio presets, preferred climate control settings, seat position and steering wheel locations, temperature points for when they prefer their heated seats to turn on, etc.). These data points could be stored in the ‘Cloud’ and associated to their account.

Then, whether the customer buys a new car, or rents one; access to their preferences from the ‘Cloud’ would allow for these configurations to be ported to the new vehicle in a minimum of time when compared to a customer manually configuring the multitude of configurations that take place in a modern-day vehicle of today.

There were no articles or research publications indicating that using the cloud for configuration or programming of automotive electronic modules is an area that is currently being considered.

In order to pursue configuration of this module, the interface requires that:

1) **Power (recommended 7-12V)** be provided and maintained for the duration of the configuration/communication.
2) **Embedded software solution** for the Arduino would need to be developed in order to act on the request for configuration(s) to be stored and retrieved from the ‘Cloud’.
3) **Hardware interfaces** would be required in order to provide the Arduino with inputs to trigger requests to store, and requests to retrieve configuration(s).
4) **Hardware interfaces** would be required in order to provide the Arduino with outputs to provide the user with configuration related data.

The benefits of using the Arduino for the paper would be the demonstrated ability of having configuration data in the ‘Cloud’ that can be stored and retrieved. As a proof of concept, the ability work with configuration data using an embedded device would allow for the configuration parameters that would be used in the automotive electronic module abstracted, and be performed without the necessity for proprietary algorithms and security access. Furthermore, there are communication libraries available for specific Arduino hardware to allow for the hardware to be hooked to different cloud data services via an open automation platform called Lagarto [4].

**C. Netbook Hardware**

Using Netbook Hardware allows for the entire simulation of hardware to be performed in a virtualized environment. The embedded device can be simulated as interfaces were the user could see. Access to the ‘Cloud’ can be facilitated through the same mechanisms used by the hardware included within the netbook for network connectivity. Furthermore, the netbook has the necessary input mechanisms to allow for a user interface to be interacted with, were additional hardware I/O is not needed.

In order to pursue the configuration of the virtualized module, the interface requires that:

1) **A netbook computer (or any computer with network connectivity) is available for simulating the module configuration(s).**
2) **A software toolset that is capable of generating simulations that can work with the network interfaces to access the ‘Cloud’**

The benefits of using the Netbook Hardware for the paper would be the ability of having configuration data in the
‘Cloud’ that can be stored and retrieved demonstrated on a user interface that can be prototyped quickly. As a proof of concept, the ability to work with configuration data using a simulated user interface would allow for the capabilities of such a solution to be visually understood via demonstration. In addition, the development environment for the application and the user interface can both reside in the Netbook, therefore aiding in providing for a quicker prototype to be developed.

III. CLOUD BASED SOFTWARE ALTERNATIVES

The ability to store and retrieve configuration parameters for the module required the selection of a ‘Cloud’ storage solution. The alternatives that were looked into were a stand-alone server that supports SQL services (a local SQL server running on a netbook), Google Cloud [5] SQL, Amazon Web Services running SQL Server [6], and SQL Azure [7] (a Microsoft solution). Evaluating the alternatives, the solutions provided by Google, Amazon, and Microsoft are what was identified within this report as Platform solutions. The standard SQL server is referred to as a non-Platform based SQL Solution. At the onset of the paper, the scope of the module configuration information was clearly of a scope that could be easily managed within the a SQL server running on any server, therefore the solutions were evaluated for future scalability, to support for future development or potential of deployment of the prototype within industry.

A. Non-Platform based SQL Solution

Since each of the SQL solutions provide for application development to have a common interface to the configuration data, the ability to scale to a Platform-Based solution is something that is relatively simple to implement. The use of a SQL server running on a local host is feasible for prototype purposes and can be ported easily to a Platform-based solution.

B. Platform-Based SQL Solution

In considering the various alternatives of Platform-Based SQL solutions, the ability for the data to be stored within the respective company’s database was supported in each.

In the evaluation of alternatives for platform services, the ability to develop applications that work with the platform was available through Amazon, Google, and Microsoft. However, after evaluation of each of the alternatives, the Microsoft solution provided the benefits of a fully integrated development environment through the use of Windows Azure, Expression Blend [8], and Visual C# [9]. This allows for scalability that provides for substantially more powerful implications to the potential for this paper’s scope to evolve and allow industries to leverage the ‘Cloud’ for cost savings in future product development. These cost savings will be described in the summary of the paper.

IV. HARDWARE AND SOFTWARE SELECTION

Having reviewed the hardware alternatives available for the paper, the Netbook along with the Microsoft Expression Blend version 4 which includes Visual C# was selected for the development of the prototype. The hardware interface allows for the prototype to be run on the netbook device and use the mouse and keyboard inputs to simulate user interactions on a physical device. Using Expression Blend allows for quickly developing a prototype interface that can be developed using XAML [10], C# and/or Visual Basic [11] that is stable, scalable, accessible, reliable, and highly secure. By using this toolset for developing the prototype, support for connecting the interface to backend business services and databases through Windows Azure is tightly integrated in the development ecosystem.

V. THE APPLICATION

In order to demonstrate the functionality the ‘Cloud’ based module configuration per the intent of the paper, the application prototype models the functionality of an FM tuner radio with six presets.

As with most radios used in automobiles or in the home, the devices allow for the user to store a set of presets. The process for associating a radio station to a preset traditionally involves the user of the radio tuning to the appropriate radio frequency, assigning the radio frequency to a preset (typically by pressing and holding one of the several available preset buttons), and repeating this process for the remaining presets available on the radio that the individual is using. After presets are stored, any short press of a preset button will recover the user’s previously associated radio station for that preset.

Figure 1 provides the state transition diagram for the application developed for the paper. It consists of six states, with entry actions that are defined in Table 1. Finally, Table 2 provides each of the conditions identified in the state transition diagram and the events that are associated with them.
Figure 1. Cloud-enabled radio state transition diagram.

<table>
<thead>
<tr>
<th>State</th>
<th>Entry Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Store Preset X</td>
<td>Tuner value stored into selected preset location in Radio</td>
</tr>
<tr>
<td>2 – Recall Preset X</td>
<td>Selected radio preset updated in tuner</td>
</tr>
<tr>
<td>3 – Store to Cloud</td>
<td>All presets in Radio updated to SQL preset table</td>
</tr>
<tr>
<td>4 – Retrieve from Cloud</td>
<td>All presets from SQL preset table loaded into Radio presets</td>
</tr>
<tr>
<td>5 – Tune Up</td>
<td>Tuner Value is incremented, and rolls over if exceeding maximum FM tuner value</td>
</tr>
<tr>
<td>6 – Tune Down</td>
<td>Tuner Value is decremented, and rolls over if less than minimum FM tuner value</td>
</tr>
</tbody>
</table>

Table 1. States and entry actions for cloud-enabled radio state transition diagram.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Save to Cloud button pressed</td>
</tr>
<tr>
<td>b</td>
<td>Tuner value stored into preset button held for over 4 seconds</td>
</tr>
<tr>
<td>c</td>
<td>One of the preset buttons was pressed for less than 4 seconds</td>
</tr>
<tr>
<td>d</td>
<td>Save to Cloud button pressed</td>
</tr>
<tr>
<td>e</td>
<td>Tuner value stored into preset button held for over 4 seconds</td>
</tr>
</tbody>
</table>

Table 2. Conditions and associated events for cloud-enabled radio application.

For this paper, the radio prototype has a total of six presets that can be stored to the radio itself, and it has the option of storing this same information to the ‘Cloud.’ In addition, the radio has the ability to retrieve previously uploaded presets from the ‘Cloud’ into the presets in the prototype. Figure 2 shows the user interface for the simulated Radio.

Figure 2: User Interface for the simulated Radio

This application allows for the concept of Cloud Based
Module Configuration to be realized. However, this solution, coupled with user authentication working in an industry standardized solution (Where all radios access the same database that store User Presets) would allow for users to be able to have their presets follow them from car to car to home radio to portable radio devices. The need to assign presets repeatedly would be alleviated. In such a scenario, if a user has a rental car when their primary vehicle is in the shop for repairs, all they would have to do is retrieve their account’s radio presets from the cloud and assign them to the rental car’s radio within seconds. Scaling Cloud Based Module Configuration parameters further in the scope of vehicle configuration elements, settings custom tailored for a user can follow them from car to car for items such as:

- Preferred temperature settings for driver and passenger
- Memory seat locations
- Mirror locations
- Steering wheel positions
- Exterior temperature set-points for when to turn on heated or cooled seats

VI. ISSUES IN DEVELOPMENT

The prototype radio interface being built within Expression Blend 4 tool allowed for a very intuitive ability to provide the inputs and outputs required for a radio meeting the requirements outlined for the paper. However, the toolset is very expansive in its capabilities for developing prototypes. It introduces a new language called XAML for use in providing the interface between the prototype being designed, and the underlying code that gets executed in C#. Therefore approximately three weeks of time went towards finding and working with the appropriate training materials to create the interface that was delivered.

In addition, using C# for the development of actions for the elements of the Expression Blend interface required further research. For example, the expected behavior for storing a radio preset is to press and hold the button that you wish to store a radio frequency on to. Unfortunately, there is little to no documentation provided with the tool to address many of the items required for this application, including a long-press of a button. Various alternatives were pursued, including starting counting threads that would run until the button was unclicked. However a mechanism was eventually found through the XAML code used in Expression Blend that allowed for the ‘OnClick’ event to repeatedly be called at a predefined rate to allow for counting to an appropriate time before allowing a preset to be stored in the prototype.

Lastly, while Microsoft Azure SQL is very similar to many of the other SQL variants in the industry, the choice of development environment required the use of C# for the application to connect to the ‘radio_preset’ database in order to access the ‘preset_table’, and each of the ‘sqlpreset_x’ values representing the ‘Cloud’-based configuration values associated with each of the radio’s presets. In the efforts to update the table with new preset values, an issue was found where the double-typed values used in the prototype were incompatible with the float-typed table columns in SQL. At the time of application completion, this issue remained. In order to provide a working prototype, the ‘Cloud’ access itself was simulated within the C# code itself.

VII. RESULTS

The prototype Radio developed within the Expression Blend 4 tool was able to simulate the functionality of a ‘Cloud’ based Radio capable of storing presets to the Radio and to the ‘Cloud’ as well as retrieving and assigning the presets previously stored to the cloud. The feasibility of the using the ‘Cloud’ for a mechanism for storing and retrieving electronic module configurations was proven to be feasible in this application.

The user is capable of tuning to new stations; assign the tuner value to a preset on the radio prototype, as well as storing all the presets to the ‘Cloud’. In addition, the user is able to retrieve the ‘Cloud’ presets and have them assigned to the presets on the simulated radio.

Figure 3: Radio prototype storing and retrieving Presets from the ‘Cloud’ platform Windows Azure.

Figure 3 captures the view of the project as was developed in order to evaluate the feasibility and demonstrate the capabilities provided by having module configuration data stored on the ‘Cloud.’

This level of functionality is of significant value to the consumer because it uses a device such as a radio to store presets. It also has the ability to store data and host applications within the Windows Azure Platform. It utilizes the suite of tools (Expression Blend and C#) which allows for scalability that could transform the way that industry views and works with electronic modules.

First, expanding on what the results of the paper show that the many industries could utilize such a mechanism:
1) **Automotive:** Provide module configurations stored in the ‘Cloud’ from the vehicle when it was built in order to configure a replacement module (as can occur when replacing certain modules after an accident). With an intelligent vehicle electrical system that can recognize a replacement part, it could conceivably retrieve the replacement part’s configuration from the ‘Cloud’. It can also update the replacement to match its intended configuration with no intervention from a mechanic, or the use of specialized tools.

2) **Consumer Electronics:** As was described in this paper, the prototype that was created mirrors the functionality of a traditional radio. However, when the ‘Cloud’ based storage of presets was included, it allows for the portability of the preset information to any devices associated with that user. While some aspects of the application’s prototype would require adaptation in order to address usability concerns, the ability to have a user’s preferences follow the user from device to device is one that would minimize the time wasted on establishing preferences on any new device the customer purchases, borrows, or has to replace.

3) **Home Appliances:** The ability for access to the ‘Cloud’ from home appliances such as a refrigerator, air conditioning, or home alarm could allow for various solutions that could impact a user’s daily routines in a more favorable way. For instance, a ‘Cloud’ connected refrigerator, could have a grocery list that is accessed by all the individuals in a household that they could access while at a grocery store. A ‘Cloud’ connected heating and air conditioning system could allow for remotely controlling the temperature of the home.

While some industries have created proprietary solutions for similar functionality, the use of platform as service solutions such as Windows Azure, Visual Studio, and Expression allow for a level of scalability that should be considered, especially as network connectivity becomes more ubiquitous at speeds allowing for usability.

The beauty of applications developed for use in this platform is the ability for the application to be ported to a variety of interfaces.

Take for example the Radio prototype used for this paper. It runs as an executable on the netbook computer, however, it can be easily ported to run as an application on the Azure Platform.

In a scenario where an application can be served from the ‘Cloud,’ manufacturers of devices such as automotive radios, navigation devices, media players, and many others could minimize the costs for their hardware by relying on the expansive storage available on a server in the ‘Cloud.’ In essence, using the ‘Cloud’ to serve an application could provide many benefits to both industry and the consumer, including:

- Minimizing hardware and localized storage to support the execution of application and media.
- Potential to reduce the cost, weight, and packaging size of a device when compared against the non-Cloud based alternative.
- Reduction in power consumption at the module level by relying on server based application processing to take the burden of processor intensive tasks.
- Potential cost reduction in deploying bug-fixes. Since applications running in the cloud are served to their clients, costly recalls can be mitigated.
- Upgrades are assured. When an application on the ‘Cloud’ is upgraded, all of the clients for the application will receive it.

Viewing the Radio prototype developed for this paper as an application served from the ‘Cloud’, Figure 4 shows how the Display is the Client in this scenario, and the Radio Prototype application and Azure SQL Database reside in the Windows Azure Cloud Platform.

![Figure 4: Radio prototype and Azure SQL both running on the 'Cloud' platform Windows Azure. Client Display is accessing the running application through the internet.](image)

VIII. **TEST RESULTS FOR THE APPLICATION.**

Use case Results for prototype application with screenshots:

Start up screen, with tuner set to 88.7.

Presets tested:

- $1 = 88.7$
- $2 = 96.3$
- $3 = 95.5$
- $4 = 98.7$
- $5 = 104.3$
- $6 = 107.5$
Tuner Up pressed until Tuner = 91.5.

Preset 1 held to store 91.5.

Presets tested:
1 = 91.5, 2 = 96.3, 3 = 95.5, 4 = 98.7, 5 = 104.3, 6 = 107.5

Retrieve from Cloud pressed.

Presets tested:
All presets = 96.3

Tuner up pressed until tuner = 100.2

All presets 1-6 held to store 100.2.

Presets tested:
All presets = 100.2

Store to Cloud pressed.

Tuner up pressed until tuner = 100.8

All presets 1-6 held to store 100.8.

Recall from Cloud pressed.

Presets tested:
All presets = 100.2

IX. CONCLUSION

The CLOUD BASED MODULE CONFIGURATION project evaluated a prototype application that allowed for module related configuration data to be retrieved from and stored to a simulated radio. This simulation proved that the capability for retrieving and storing preset information is viable given the ability for a module to have access to internet connectivity.

In addition, the paper evaluated other future alternatives of ‘Cloud’ based applications. The alternatives included applications that utilized the ‘Cloud’ to store and retrieve data as was performed in the Radio prototype, as well as applications that were wholly served from the cloud to a user client.

REFERENCES


