Abstract - The design and implementation of a browser-style interface to a home automation network is described. The interface supports browsing and navigation of network devices and context structures and the user can interact with individual devices on the network, and access and control context and object structures within these devices.

The interface can be used to access a local home automation network from a standard desktop PC, or from a TV/STB system with built-in interface hardware. Furthermore, as the interface is implemented using conventional Internet and home network technology it can provide access and control services to the home network from any computer with an Internet connection.

1. Introduction
The Consumer Electronic Bus (CEBus®) is a multi-media LAN standard developed for home automation applications [1]. In earlier work we discussed some of the issues involved in providing access to a CEBus network via a conventional Web browser and Java® virtual machine JVM [2]. The aim was to demonstrate that wide-area-network (WAN) access to a local control network was practical and broadened the scope and range of applications of home automation technology.

However we note that the market penetration of such technology has remained slow, apart from certain niche applications. The authors feel that this is largely due to a combination of (i) poor end-user perception of the benefits of home networking and (ii) a reluctance on the part of consumer electronics manufacturers to take risks in applying new technology in unproven markets.

Further, drawing on the recent example of the mass public acceptance and, in some instances, obsession with the Internet, we argue that the lack of a practical, generic and readily available user-interface to home automation networks is a key reason for poor user perceptions and lack of market penetration. In this paper we offer a preliminary approach to the issue of providing a generic user-interface suitable for the smart home of the 21st century.

2. Philosophy and Design Issues
In this section we give an overview of the issues involved in providing a useful, but non-trivial user interface to a home automation network and the individual devices which are members of the network.

2.1 End-User Requirements
It is important that any such interface is both user-friendly and can provide generic access to the broad range of consumer appliances found in a modern home. This requires a degree of flexibility and interoperability which is currently unavailable in any of the interfaces used in, for example, the desktop computing market.

In particular it should be noted that few home users of such a system will be more than marginally computer literate. Such users are unlikely to tolerate the situation which persists with computer peripherals, where every new piece of hardware has a manufacturer-supplied software device driver which must be installed by the end-user. Thus the interface software for any new consumer devices must not only be user-friendly, but it should also be self-installing and largely self-configuring. The vast majority of home users will tolerate nothing less.

2.2 The Requirements of Manufacturers
In addition we note some of the needs and requirements of the manufacturers of consumer electronic products. Such companies require a means to (i) differentiate their products and to (ii) provide user-friendly access to complex features.

It is also important that manufacturers have the flexibility to change and adapt the user interface, ideally without requiring that devices be manually upgraded or otherwise serviced. This is particularly important where a single consumer appliance may serve several different markets. If the usage and complexity of this appliance can be changed simply by loading a different user interface then the same physical appliance may be able to serve multiple markets, reducing overall production costs for the manufacturer. Finally, manufacturers may be able to earn additional revenues by selling user-interface (UI) upgrades to add new operational features and enhance the functionality of their appliances.

2.3 The Limitations of Consumer Appliances
It is also noted that many consumer appliances are not yet sufficiently intelligent to support full Internet connectivity or to present a high-end graphic user interface (GUI) to the user. However most appliances contain sufficient intelligence to support a product
modeling scheme such as the Common Application Language (CAL) as defined in [3]. In fact CAL was designed to meet the specific requirements of home automation networks.

Thus we do not feel that local home automation networks will, in the near future, be directly integrated with broader WAN networks such as the Internet. In the opinion of the authors, a much more practical and secure approach employs a local home gateway which intelligently brokers network traffic and integrates the home network with the outside world.

Further, we feel that the real benefit of such integration will not be an ability to remotely access the home network, but rather the ability of the home network to access external services and resources to enhance its own functionality and that of the appliances connected to it. In the context of this present work these external resources could be complete GUI elements for each consumer appliance in the home.

2.4 System Requirements
Consideration of the above issues led us to the consider in some detail the hardware subsystems and the software architecture necessary to support the requirements of both end-users and appliance manufacturers. Details of the gateway system implementation are given in a companion paper [4]. In this paper we focus on how this gateway system described in [4] can provide a generic mechanism for downloading user interfaces to home automation devices over a WAN.

In section 3 we describe how the gateway architecture supports intelligent UI elements which can interact with daemon programs to send and receive CAL messages to real network devices, or Rdevs. These UI elements, or UIDevs, may contain their own independent CAL generation and interpretation layers, or they may also take advantage of inbuilt primitives within the CALNetd daemon program. Furthermore this architecture supports the creation of virtual home automation networks and virtual devices. It is thus possible to create and test, in software, complete device structures with related user interfaces.

In section 4 we describe the practical provision of a GUI for a simple home automation network. An overview is given of the actual hardware and software used in the implementation. We also describe how the database of device information contained in system memory is made available to the user through a graphical browser interface, similar in concept to a Web browser. When devices are accessed through this "device browser" a specific UI element for that device is loaded by the system.

A description is also given of how several UIDevs may be combined in a group and operate through a single network socket. Such groups of GUI elements provide the basis for building more complex interface structures and we have coined the term HiPlet, or home-interactive programlet to describe their functionality.

Section 5 gives some simple, but practical examples of HiPlets and their functionality is described. We then outline how they can interact with each other, or with real network devices. A summary and some conclusions are then drawn in section 6.

Fig: 1 Overview of System Architecture and UI

3. System Architecture & Implementation
In this section we briefly describe the gateway architecture and how it supports intelligent UIDevs which interact with the core system daemon, CALNetd and with external real-devices (Rdevs) and internal virtual-devices (Vdevs).

3.1 System Hardware
In its present implementation the system software runs on a desktop PC platform, but as all of the key system software is coded in a platform independent language the core system elements are suitable for down-sizing and porting to the embedded hardware platforms used for Set-Top-Box or Web-TV appliances. The interface to the powerline CEBus network is through a proprietary module, but the hardware components are available as low-cost ICs for production purposes.
3.2 The Gateway Software Daemons
The software architecture of the Internet/CEBus gateway is described in detail in a companion paper [4].

Briefly a lightweight IOd daemon - represented as a hardware abstraction layer (HAL) in Fig 2 - continually monitors the traffic on a local powerline network. As devices initiate or receive messages, coded in Common Application Language (CAL) this system daemon provides low-level hardware acknowledgements and responses as required, then passes on the CAL messages to a big-sister daemon, CALNetd.

The CALNetd daemon records the state of network devices in a local registry of devices. Thus state information for the local CEBus network is available without the need to query individual devices. It also brokers and routes CAL messages between the various flavours of system devices: UIDevs, RDevs and VDevs.

3.3 The Device Browser Interface
The user interface has two distinct components. Firstly there is a graphical device browser for the local home network. This browser simply shows the list of network devices registered in the local database, in the same way that file management utilities show a list of files in a local filesystem. However when a device is accessed in the browser the user-interface for that devices does not originate from the local system software - instead it is loaded, as a HiPlet, from an HTTP-style universal resource locator (URL).

3.4 Software Interfaces to Individual Devices
This scheme allows the local Internet/CEBus gateway to load user interfaces for any consumer appliances with a CEBus powerline interface and an embedded URL.
Furthermore it satisfies the requirements of manufacturers to be able to customize and differentiate their products by creating unique user interfaces and product features.

4. Implementation of the User Interface
Prototype implementations of the user-interface elements describe above have been developed on a local TCP/IP network and interfaced with a local powerline CEBus network. The gateway is implemented on a standard desktop PC and the interface to the powerline network is via the RS-232 port and a proprietary interface module. The PC is connected to the TCP/IP network using standard thin-wire ethernet.

In its normal mode of operation the IOd daemon program (HAL) monitors the local home network traffic and provides hardware-level responses. The actual CAL packets are passed onto CALNetd which, if appropriate, routes the packets to UIDevs or VDevs and records and registers new devices. It can also register and interact with active virtual devices (VDevs) or user-interface devices (UIDevs). User access to the registered devices is via the browser interface shown in Fig 3.

4.1 The Network Browser
The browser described in section 3.3 above is quite a simple implementation of a device browser. It provides a very useful level of interface for most end-users who only need partial access to the complete functionality of most network devices. However, for the purpose of developing new UIDevs, HiPlets and VDevs it would be very useful to have a more flexible browser which provides detailed access to the complete CAL structure of network devices.

In the interests of completeness we show in Fig 4 such an alternative browser implementation which can provide very detailed access to the CAL structure of a network device. In addition this alternative browser implementation is also aware of many of the standard Device and Context structures defined in [3]. It is somewhat more sophisticated than we would expect to see in a home user's system, providing access to the entire context/object-instance-variable structure of a CAL device. The function of such a browser is to provide an engineering tool to assist in the development and debugging of new UIDevs and HiPlet interfaces to the home network.

4.2 User Interface Devices (UIDevs)
In our implementation these user-interface elements are specialised Java applets which are uploaded from a URL on a TCP/IP network and attach through a container object, to a network socket provided by the local CALNetd. They may also attach through a direct-interface-layer or DIL. The socket-based connection allows the UIDev to exist at a remote IP address or in a separate JVM, whereas the DIL requires that it reside in the JVM of the local gateway.

Note that a UIDev can access additional system functionality over that normally available through a standard TCP/IP network socket. Much of this functionality is designed to support the low-level requirements of the local home automation network and also to provide some CAL related services to simplify the design and testing of UIDevs. In the current system implementation many of these system functions are still under development. Some brief details of the basic CALNetd services which are currently available are given in section 4.4.

**Fig: 4 An Enhanced Device Browser can provide very detailed access to the CAL Object Structure of Appliances connected to a Home Automation Network.**
4.3 Home-Interactive Programlets (HiPlets)
To support the development of more sophisticated UI elements we have introduced the concept of grouping UIDevs into interactive container objects known as home-interactive programlets, or HiPlets. This allows several different network devices, or appliances to have their functionality combined and accessed through a single user-interface. At the same time, each device may have its own independent interface.

Each such HiPlet attaches to a specific CALNetd socket which is mapped onto a local CEBus MAC address. Thus a one-to-many mapping between a single user interface and several local network devices can be achieved.

4.4 CALNetd Interface Services
As was mentioned above, we are still developing and experimenting with the internal services and structures of CALNetd. The basic services which are currently available include:

- full/filterd packet dump
- tx packet/rxack packet
- list network devices
- list/seek device state info
- list/seek house state info
- list/seek context state info
- select TCP/UDP broadcast
- network statistics
- MAC binding services

5. Practical Device Interfaces
In this section the intention is to introduce some simple HiPlet implementations which are available live on our Website [5]. These are quite basic interfaces, but they demonstrate how more complex interfaces can be built by combining interface elements.

They first example we give is the standard CEBus light-switch. In this case it is a bit different as it is a light-switch with a GUI. A more complex example is then given which combines a keypad HiPlet with a display device HiPlet. For convenience we will show the software implementation of each device, but we note that each device can have a real-world equivalent which could just as easily substitute for the software object model representing it. Thus a real CEBus keypad could send data to the software display HiPlet in exactly the same way as the virtual keypad does.

5.1 The Light-Switch with a GUI
This is the classic CEBus example. In our simple example a binary sensor is bound directly to a binary switch.

The sensor interface could equally well be a real sensor device, such as a PIR sensor in a security system. In our virtual example activating the check box has the same effect as activating the PIR sensor. This generates a CAL report to the binary switch UI which turns on a virtual LED to indicate that the switch, which could be an alarm relay, is active.

5.2 A Sample Keypad Device
A very common UI device is the simple numeric keypad. These devices are found in many applications, from touch-tone phones to point-of-sale terminals. In Fig 6 we show the equivalent HiPlet with a single UI context. When a key is activated this generates a CAL message containing the value of the key depressed. The message will be passed on to any devices which are bound to the keypad HiPlet.

5(b) The UI element for a basic binary sensor.

Fig 6 The Keypad - a single UI-context HiPlet; more complex HiPlet interfaces may have multiple UI-contexts.
5.3 A Display HiPlet using List Memory
This simulates one of the most common display devices - a simple ASCII character display. The HiPlet may be bound to any CAL source of character data.

![Fig 7 A simple Display HiPlet - this is based on the List Memory CAL object.](image)

5.4 Interaction Between Devices using CAL
In Fig 8 below we show the basic browser described in section 3.3. This has been used to load HiPlet interfaces to a keypad and a display device, as described in sections 5.2 and 5.3 respectively. These devices are bound to each other, so that the CAL output from the keypad is directed to the display HiPlet. A live, Internet-based, demonstration is available from [5].

6. Summary and Conclusions
Prototype implementations of the user-interface elements described above have been developed on a local TCP/IP network and interfaced with a local powerline CEBus network. The gateway is implemented on a standard desktop PC and the interface to the powerline network is via the RS-232 port and a proprietary interface module. The PC is connected to the TCP/IP network using standard thin-wire ethernet.

The most important lessons to be learned are with regard to the auto-configuration of UI elements as described in this paper. Briefly this is a non-trivial exercise, and it is clear that the gateway between WAN and local control network must be quite an intelligent one, if auto-configuration is to be achieved in practice.

Finally, in order to achieve a truly generic mechanism for loading UI elements for home automation networks it is essential that open standards be developed and promoted in this field. As our work in this area proceeds it is hoped to make more detailed results available through the Web-site at [5].

![Fig: 8 The Device Browser described in 3.3 is shown with two single-context HiPlets, A Keypad and a Display.](image)
REFERENCES


Biographies

**Peter Corcoran** received the BAI (Electronic Engineering) and BA (Maths) degrees from Trinity College Dublin in 1984. He continued his studies at TCD and was awarded a Ph.D. in Electronic Engineering in 1987 for research work in the field of Dielectric Liquids. In 1986 he was appointed to a lectureship in University College Galway. He is involved in international Joint-Venture projects in both China and Eastern Europe. He is currently teaching on a full-time basis at University College Galway. His current research interests include home automation networks, Internet technologies, embedded computing applications, and telecommunications technologies. He is a member of the IEEE and a technical committee member of the IEEE Consumer Electronics Society.

**Joe Desbonnet** received his B.Sc. degree in Physics and Electronics from University College Galway in 1993. He has run a professional WWW site at http://www.wombat.ie since 1992. He is presently working as a Research Associate at University College Galway and is a director of World-Wide-Web Marketing, a Galway based Internet services company.