Advanced Control

# System Components

* Control Core (IIS/Windows Service)
  + Hardware Serialisation
  + Client Authentication
  + Module Management
  + Connection Management
  + Input Output Formatting
  + Status Reporting
  + Mainline [Optional User Defined Module]
* Client Interface (Managed DLL)
  + Device Listing
  + Function Registrar
  + Status Information
* Device Modules
* IO Modules
* Mainline Structure
* Hardware Inspector (Tool for defining systems)

## Core Control

This software enables client machines to interact with a system through interfaces that can be defined in both code and XML.

### Hardware Serialisation

System hardware is inspected and available physical devices, which can be used for interaction, are listed and associated data stored with them.

### Module Management

XML is inspected to find required device modules, these devices modules are loaded into memory in an uninitialized state. Mainline is also loaded if available and IO modules are loaded on demand.

### Connection Management

Device modules are assigned hardware communication devices based on their generic XML connection types, ordering (implicit and explicit) and information supplied by the module.

Connections are tested (module function), status of each module made available through publishing.

Clashes are not resolved (first in first serve / heterogeneous) however the error is logged and status flagged.

* XML defines
  + Device Module
  + Communication Medium
  + Medium Settings (BAUD, IP Address, Protocol ect)
  + Optional IO Formatting Requirements (If overwriting the module defaults)
  + Connection type priority / order

### Mainline

This is where user defined functions are registered. Modules define states and actions which a user can request to be changed or action made and the referenced module will attempt to move to that state or request an action.

Users can define macros in XML that change multiple states and trigger different actions.

Sometimes a simple macro is not enough. Changes in status might need to trigger actions based on state which requires more logic than can be defined in XML. Mainline allows users to define delegates for status events, complex macros and, in fact, interact with all parts of the system.

### Client Authentication

XML defines how various clients must authenticate to be able to interact with the core system.

Should support Windows, LDAP and Internal authentication methods (some kind of interface defined). This lends itself to windows based biometric and onetime key systems through vista and windows 7 authentication providers.

### Status Reporting

Status registrars must be setup for clients and mainline so only required status is sent through raising events.

This reduces medium traffic and allows for 2 way communication to all subscribed endpoints.

## Client Interface (Endpoint)

Client interfaces can be defined as anything from a control panel to specialised software running remotely. A managed DLL is provided to enable simple interaction with the remote system.

### Device Listing

List of generic device types available, including their non-generic device information.

### Status Information

Feedback required from devices must be registered to provide live status feedback.

Indicators that may be used on screen need to be linked to events provided by the status reporter. This follows the standard subscription method of .net however is encapsulated to hide details of registering the requirement on the far end.

### Function Registrar

The status registrar should also handle the reverse case where the client wants to request changes occur at the far end – not entirely sure how this will be implemented the .net way, still requires testing and research.

## Module Layout

Events

Control module sends direct messages to the device modules. (Class function)

Devices modules raise an event for each status item that changes with a string the core module can use to request the status if an end point requires it.

Device Module

IO Formatter (optional)

Core Control Module

Core Control will send an initialise command to the device module when connections are setup and ready.

The module may be interrogated for connection information (ie: RS232 or IO Formatting requirements)

## Endpoint Communication

Events

End Point

Core Control Module

End points make initial connection to Control Module and send event requirements.

Core module then connects to the End Point and sends event data as it happens. Initially all event data is sent to synchronise the End Point.

Requirements

## (.Net) End Point to Interface Communication

Different modules would have to be setup for every type of interface we would like to support.  
This could include: Ruby, Flash, Java etcetera.

Event Registration

The Interface code must register which functions it wants called on various events.

Event registration will be fairly overloaded to allow for generic function calling, button toggling, and type conversion etcetera.

The end point interface stores local copies of the status information.

Events

Interface Code

End Point Interface

# System Help Wizards

To help users understand what is available on various systems, provide tools to define device modules, connection order, macros and debugging.

## Key Ideas:

1. Systems are grouped in the database at the user discretion.
   1. This may be due to a build program
   2. Physicality of systems
   3. Security or system use
   4. Ect
2. Systems can be joined to multiple groups

## Setting up the Controller:

The controller must be joined to a systems group for setup in the database.

* The Setup will request database connection information
* An name that can be used to identify the system (An id key will also be created)
  + Names can change
  + Names or id can be used to communicate to systems (id preferred)
    - Interface wizard can lookup connection details based on room name
    - Interface can lookup available functions
* This will automate the serialisation of capabilities of the controller
  + COM Ports, Nic Cards and other devices of interest
* The setup can then be completed entirely on the controller or remotely.

A second wizard will then walk you through attaching devices to a system.

* New modules can be installed into the database as well as using currently available modules.
* It will also be possible to test device connections remotely.
* Devices are given friendly names (Local to each system or uses default)
  + A list of previously used friendly names will also be available

Yet another wizard can be used to build and test macros, as well as being used as a debugger

* Create new macro or import a macro from another system
  + Importing is sharing (many to many)
  + Changing an imported macro will ask if you would like to only change this one or all
* Name the Macro (Information on how to call the macro from a interface will be provided here)
* Drop down list of known commands provided (Queried from the Module(s) and already written macros)
  + Alternatively this can be defined manually
  + Detects and avoids macro loops (Where both macros call each other, recursive check including sub macros)
* Debugger connects to a live system and outputs all the status items being sent and received from the live program. (Or related to an individual device)

Mainline codes can also be loaded and unloaded in real time.

* Debugger could be a mainline module
* Mainline modules have access to all system events
* Can control module connectivity