SIEMENS

# Colledia Control™
# V4.5 General Documentation

| | | | |
|---|---|---|---|
| Project: | Colledia Control™ | Subject: | V4.5 General Documentation |
| Date Created: | 19/7/07 | Created By: | Mark Baldry |
| Date Amended: | 19/7/07 | Amended By: | Andrew Prince |
| Status: | Draft | | |

# Contents

# Introduction

## 1.1 What does this document tell you?

This document will provide an overview of Colledia Control™ version 4.5. For those that require further details on each module, including detailed specifications, separate documentation is provided.

It is assumed that the reader has an understanding of and previous experience with BNCS.

## 1.2 What is v4.5?

Colledia Control™ v4.5 is designed to:

- Provide a standard way of achieving various tasks including:-
    - Simple installation with a coherent set of files
    - Workstation updates
    - Panels configuration
    - Common tasks
- Encourage a  standard approach
- Provide configuration tools
- Allow drag and drop panel creation for simple panels
- Allow creation of re-usable panel components
- Allow the power of C++ to be used for automatics and complex panels
- Speed up delivery of projects
- Provide a consistent means of navigation between applications

This all leads to greater supportability

## 1.3 What v4.5 is not.

Colledia Control™ v4.5 is not:

- A closed system. Panel code is supplied in the same way that Applcore is, but like earlier versions the source code for the core system is not
- C++ is not closed, it is just a different, more powerful language
- Completely new. It still uses the same drivers and CSI as the network layer is identical
- Designed to restrict usage. It should help do things in a standard way, thus making the system easier to maintain.

# 2 Concepts

## 2.1 System

Colledia Control™ v4.5 is designed to allow a developer to easily switch between systems in their entirety without any manual moving, renaming or deleting of files. To achieve that all files for a system are within a single directory structure. The current system is selected using environment variables, these being

- CC_ROOT
- CC_SYSTEM.

The path also needs to have

- %CC_ROOT%\%CC_SYSTEM%\windows\bin
- %CC_ROOT%\%CC_SYSTEM%\windows\lib.

The installer will set these items automatically.

Once v4.5 has been installed there is a utility to enable a developer to rapidly switch between systems by changing these settings.

Note that there should be no Colledia Control™ files in the c:\windows, or equivalent, directory.

### 2.1.1 File System

The directory structure for a single system called “latest” is shown in **Figure 1 Directory Structure**; there may be other systems (not shown) in the CC_ROOT directory (“c:\Colledia” in this case.)

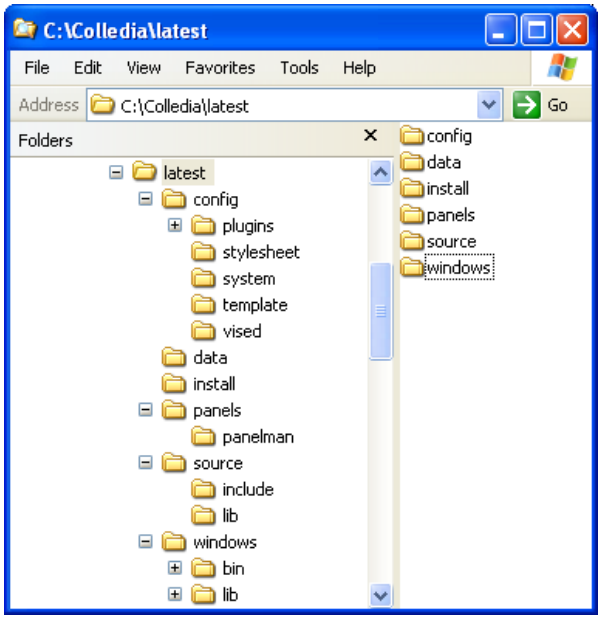


**Figure 1 Directory Structure**

All the binary files are located in the project ‘windows’ directory, with the ‘bin’ directory containing all of the executables while the ‘lib’ directory contains all of the DLLs. The project specific drivers should go in a subdirectory of the ‘bin’ directory called ‘drivers’.

## 2.2 Devices and Parameters

### 2.2.1 Simple Devices

In a broadcast system there are often many types of equipment - Routers, ARCs, Satellite receivers etc. These are called device-types, and for each type there can be many of that device-type installed, called instances.

V4.5 uses the idea of an instance to allow panels to ask the system to adjust an instance. The system will then lookup the device-type for that instance, and start the appropriate panel.

Each device-type can be described as a group of parameters, each with different attributes. For instance a processing amplifier may have a gain parameter, which has a range from -127 to +127, and an output parameter, which is an enumerated list with input, bypass and test as valid settings.

### 2.2.2 Complex Devices

Some devices will present themselves through other devices. For instance an ARC may present itself directly for most parameters, but the PSU status may be through a GPI. This is known as a complex device-type. The system will treat these in the same way that it treats simple instances.

### 2.2.3 Composite Devices

In many cases there are virtual devices formed of several real devices in a chain, these are called composite devices. The system will treat these in the same way that it treats simple instances.

### 2.2.4 Type Handlers

For each type of device a default panel can be assigned for making adjustments to that device, this is called a type-handler. Type Handlers also apply to both composite and complex devices. In the case of a composite device the panel can be designed to control all of the individual items in the composite device.

## 2.3 States, Styles and Skins

Rather than hard-coding how a button looks it can be configured. This is achieved using states, styles and skins. This leads to a consistent appearance, makes it easy to change and can speed up re-use.

### 2.3.1 Skins

How a control is drawn is determined by the skin, which defines the shape of the button, the edge shading etc. The skins are hard-coded, but can be selected for a system.

### 2.3.2 Styles

Controls are given a style, which will typically set static items such as fonts and possibly colours. For example a style called “ParamLabel” may set the font to size 14 with white text on a blue background. Setting all parameter labels to use this style will make them appear the same and easy to change globally.

### 2.3.3 States

Controls can have a state applied to convey information, for example a control may have a state applied called “AlarmOK” which is defined as having a green background. When the state of a parameter has changed the state of the button can be changed to “AlarmFail” which could have a red background. More than one setting can be applied in a single state. In the given example the text, font size and text colour could all be different. The important thing being that the changing appearance of the control is set using a state. It should be noted that a style can be applied as well as a state. The style should contain the settings that do not change, i.e. font name.

**Figure 2 Styles, States & Modern Skin** shows two controls set with style ‘label’, in this case with white text on a blue background. It also shows two buttons with the same style but different states, indicated by the colour change. **Figure 3 Styles, States & Traditional Skin** shows the same panel with a different skin applied, but with the same styles and states.

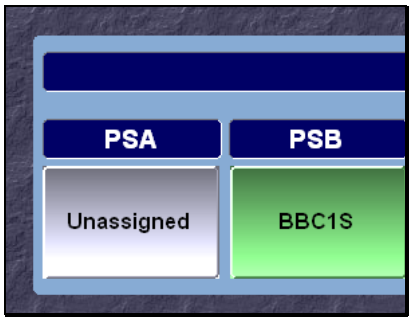


**Figure 2 Styles, States & Modern Skin**

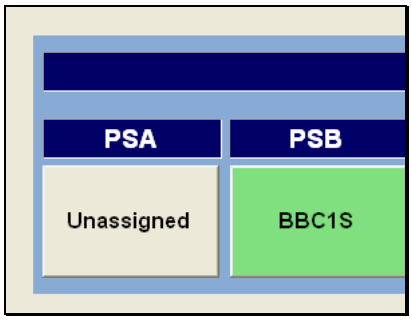


**Figure 3 Styles, States & Traditional Skin**

# 3 Configuration

To aid with configuration of the system there is a configuration tool, which consists of a server and client. The configuration server runs on the nominated server machine while the client can run on any or multiple Colledia Control™ workstations.

The aims of the configuration tool include:-

- Reduce the need for the developer to manually change the files
- Incorporate some cross-checking where appropriate
- Guide the developer through various configuration tasks.

The configuration client application will copy the files that the user wants to edit to a temporary directory on the local machine. The configuration tool will then allow the user to edit these files. While a client has the files open on their machine for editing no other client can change the same file.

The configuration client will load various plug-ins to perform the various configuration tasks. Each of these plug-ins works on the same temporary files, so there will be interactions between them. Therefore it is recommended that changes are all saved or rejected as a group otherwise partial changes may be applied.

It should be noted that not all of the configuration tasks can be completed without some knowledge of the files, and some files still require manual editing. If you do manually edit files remember to put them on the server, see **4.1.1 Workstation update** for the reasons why!

# 4 Start-up

At start up the main application workstation manager (BNCS_WS_MAN.EXE) is run. It manages the workstation performing several tasks detailed below. It replaces typical tasks that were previously done using batch files and 'launch'. It is expected that a shortcut to the workstation manager is placed in Windows Startup so that it is automatically run whenever a workstation is started. The overall procedure is shown in **Figure 4 Workstation manager run order**.

The workstation manager will log activity to the "%CC_ROOT%\%CC_SYSTEM%\logs" directory.

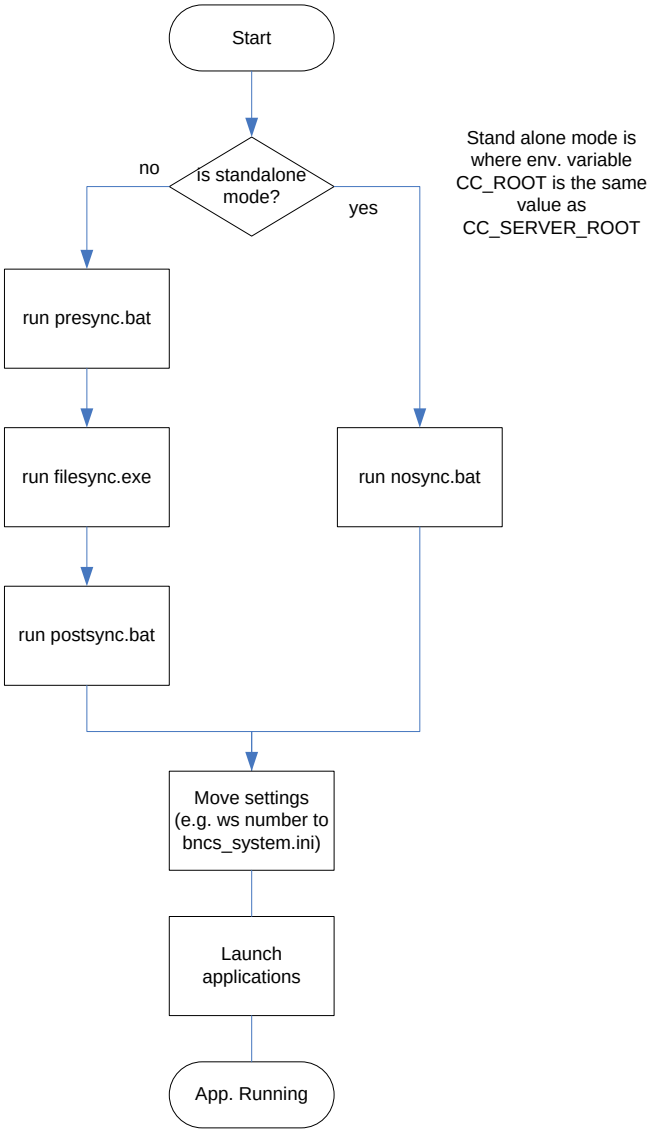


**Figure 4 Workstation manager run order**

### 4.1.1 Workstation update

The workstation manager will operate in two modes, stand-alone or client. The mode is set using the bncs_inst_env.exe application. Each mode will run relevant batch files.

All of the batch files are expected in the "%CC_ROOT%\%CC_SYSTEM%\windows\bin" directory. If the batch files do not exist, as they may not be needed for every system, workstation manager will carry on. The batch files must complete if they exist, which means that they should not wait for user input otherwise the workstation manager will be paused until that has happened.

#### 4.1.1.1 Stand-alone mode

In stand-alone mode no files are copied from anywhere but 'nosync.bat' is run. Stand-alone mode is expected to be used for developer machines and the main system server. Typical uses of 'nosync.bat' include copying files on the server that need to be in the v2 path, for instance driver configuration for older drivers that run on the server, or for implementing custom settings on development machines.

#### 4.1.1.2 Client mode

In client mode 'presync.bat' is run before performing a file sync, then 'postsync.bat' is run.

**Important**

The file sync ensures that the local machine is exactly the same as the server; this includes removing local files that are not on the server and deleting newer files. The file sync ensures that the master of any given file is on the server and allows a backup of the server to actually be a complete snapshot of the system. If you manually make changes on a local machine, and you want to keep them, then copy the files back to the server otherwise they will get over-written.

Some directories are excluded from the synchronisation process; these are 'source', 'temp', 'docs', 'data', 'install', 'backup' and 'logs'.

A reserve server can also be defined such that if the main server is not available the update can still take place - in this case you need to ensure that the reserve server is kept in sync with the main server.

Typical uses of postsync.bat include copying files on the local machine that need to be in the v2 path, for instance driver configuration for older drivers, or workstation specific files such as bncs_system.ini. We would recommend that older drivers, if required to be used in new projects, be updated to conform to the new configuration file locations.

### 4.1.2 Copy settings

This part of the workstation manager copies settings from the local machine into the newly copied files to ensure that any workstation specific settings are maintained, including the adapter number and the workstation number.

### 4.1.3 Launch

The workstation manager will launch the applications that are required on the workstation. The first application that is launched is CSI, followed by all the other applications which get queued until CSI has fully loaded. Setting the launch order and any options is done using the configuration tools.

### 4.1.4 Normal running

Unlike previous launch applications, the workstation manager will stay running and monitor the applications that it has started. If an application crashes workstation manager can be set to re-start it. If CSI crashes the workstation can be set to re-start. For applications to be able to restart they must not put up a dialog when they crash as this means that the application is still running. Each application will re-start a maximum of 10 times; after that the workstation manager will stop re-starting the application.

When the workstation manager is shut down it will close all of the applications that it started, including CSI.

# 5 Installation Procedure

## 5.1 PC Setup

Note that the following instructions are intended as a guide not a step-by-step set of instructions as these may vary depending on your PC and operating system.

- Bios settings
    - Disable any power saving settings
    - Start-up automatically after power loss
    - Do not report keyboard errors on boot
    - Disable hyper threading.
- Create user accounts to be used as required. It is suggested that there is a single user account for all machines to make file sharing easier, e.g. cc_runtime.
- Date/Time
    - Set the time zone as required
    - Disable auto-adjust for daylight savings on all machines, except the master. On the master time server you may need it set depending on how or if you are locking to an external reference.
- Security settings are specific to your project, but typically:-
    - Automatic updates off
    - Disable firewall
    - Disable security centre alerts.
- Network settings are specific to your project, but typically:-
    - Set the IP address
    - Enable NetBIOS over IP
    - Enable file and printer sharing on the server.
- Power options
    - Set to always on

    - Do not turn off hard disks, displays etc.
- User password options
    - Ensure that the user does not need to change the password at next logon
    - Set the user not to be able to change the password
    - Set the password never to expire
    - Automatic logon for normal user
        - either
            - Edit the following settings in the registry HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon
            - DefaultUserName Set as required
            - DefaultPassword Set as required.
            - AutoAdminLogon Set to 1. You may need to create this entry
        - or
            - Start>Run “control userpasswords2”
            - Clear “Users must enter a user name and password to use this computer”
            - Select Apply
            - Set user name and password as required.
    - Note that these settings can be set using TweakUI from Microsoft.
- Display properties (in Appearance|Effects…)
    - Ensure that there are no effects set for menus and tool-tips
    - Set the display resolution to match the system.
- Taskbar settings
    - Set not to keep the task bar on top of other windows.

You may have other settings that you wish to set such as explorer display preferences and Start Menu appearance.

## 5.2 Server

Run the Colledia Control™ installer, selecting the options as required.

Using the configuration server/client

- Create the workstation
- Set the workstation to launch:-
    - CSI, this ensures that the router database files on the server are kept up to date when there are database names changed on the system

    - Configuration server (config_server.exe)
    - Any other applications required, such as drivers etc.
- Create a shortcut to workstation manager in Windows->Startup so that it automatically starts
- Share the Colledia Control™ system directory to enable the update mechanism to work; you can set this to be read-only if required.

## 5.3 Workstation

To install a workstation:-

- Navigate to the share on the server.
- Locate & run the workstation installer. <c:\...\installer\CC_Workstation_Install.exe> This will install the minimum number of files to run the workstation manager, including update, and guide you through the rest of the setup.
- Set the options required; note that the server path needs to include the share on the server
- Select the correct network adapter
- Start the workstation manager. This will do an update from the server to install the rest of the system.

# 6 Panels

Panels are edited using the Visual Editor application, bncs_vis_ed.exe. Note that panels are expected to be in the 'panels' subdirectory for the system, and the panel directory must match the name of the main panel loaded, e.g. "%CC_ROOT%\%CC_SYSTEM%\panels\test1\test1.bncs_ui". The same applies to script panels that are DLLs.

Panels are loaded at runtime by the panel manager <%CC_ROOT%\%CC_SYSTEM%\windows\bin\panelman.exe>. For client machines it is normal that this is one of the applications that the workstation manager launches. The end user can not and does not need to know if the panel that they are currently using is a simple panel or a scripted panel.

Icons for the panels that the panel manager uses on the menu buttons are stored in the folder with the panel. The name does not have to match the application; the icon association is set in one of the configuration files.

## 6.1 Controls

There are a range of controls, known as smart controls, which require no programming to communicate with a device, just configuration. The list of buttons that are available for use in the visual editor is in "%CC_ROOT%\%CC_SYSTEM%\config\vised\buttons.xml" enabling buttons to be removed or added to a specific project as required.

Images for range controls are stored in the "%CC_ROOT%\%CC_SYSTEM%\panels\images" directory; which is also the suggested location for any other images that are used so that they can be accessed by all panels.

## 6.2 Targeting Controls

Smart controls can be targeted at a specific instance of a device; this can be done automatically by the system at run time, programmatically at run time, or at design time. To target a control at an instance that forms part of a composite instance the ‘group’ parameter of the control needs to be set to the group element in a composite instance. See separate document for more details.

## 6.3 Simple Panels

Simple panels have no code associated with them; any buttons or labels that perform a function are smart controls.

Only the visual editor is required to create simple panels.

Note the term simple can be misleading as script controls can be placed on to the panel and a lot can be done using only smart controls.

## 6.4 Script Panels

Script panels have code associated with them in the form of a DLL, which contains whatever functionality is required. To create a DLL either use the wizard to create a new panel, or if you already have a panel that you need to create a script for, create the script using the appropriate toolbar item in the visual editor.

The panel layout and any simple controls are configured using the visual editor. Any coding that is required is completed using Visual Studio. The Visual Studio project is created using a wizard, which will leave you only to add your code, all other settings being done for you.

When looking at the code created by the wizard it will be seen that there are several functions already created, these supply you with basic functionality. For full documentation on all of the available helper functions see the separate Web documentation.

## 6.5 Re-usable components

One of the controls that can be added to a panel is a script control. A script control itself then hosts another panel, which means that you can create a component that is re-used many times on one or many panels. The script control is designed to be used where the standard smart buttons do not provide the required functionality. Note that script controls are the same as script panels, the difference being that a panel is hosted by the panel manager while a script is hosted within a panel.

**Figure 5 Script Controls** shows an example panel where there are many instances of a component depicting a switch. In this case pressing the component pops up a control to toggle the switch. In this case a normal smart button could not provide the required functionality as the outgoing switch needed a momentary closure and the incoming status was on a different GPI.

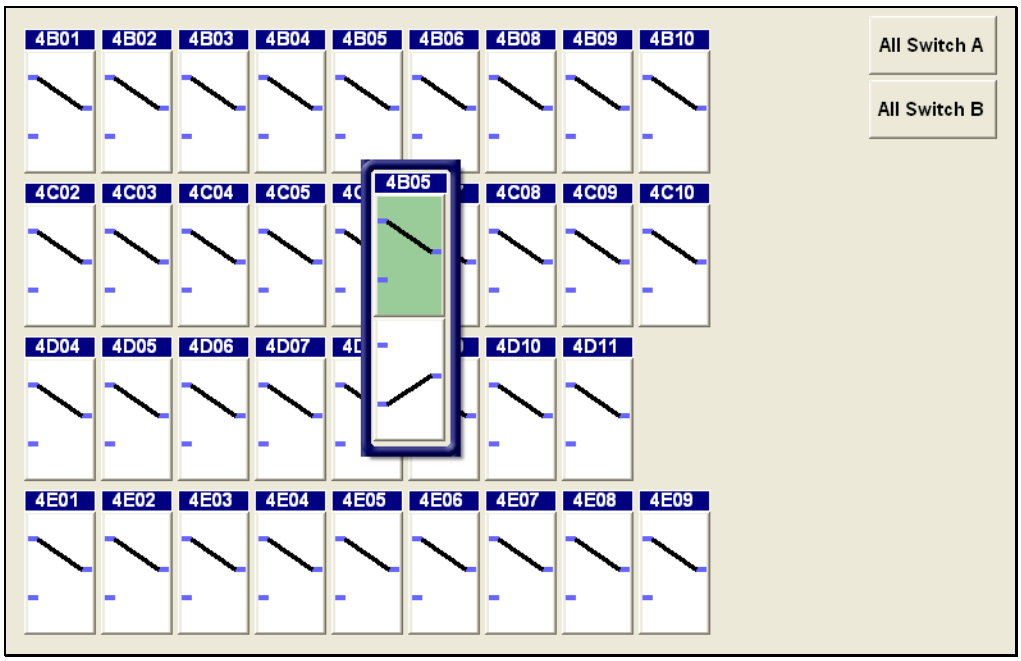


**Figure 5 Script Controls**

To set parameters on a script at design time the function “parentCallback( parentNotify *p )” in the script must return a list of items when the command is “return”. See **Table 1 Script parentCallback function** for an example where the parameter “Label” can be set at design time.

```
// all parent notifications come here i.e. when this script is just one
//  component of another dialog then our host might want to tell us things
bncs_string test::parentCallback( parentNotify *p )
{
	if (p->command() == "Label")
	{
		sLabel = p->value();
		//Add code to action this change
	}
	else if (p->command() == "return")
	{
		bncs_string sRet;
		sRet += "Label="+sLabel + "\n";
```

**Table 1 Script parentCallback function**

## 6.6 Pre-defined panels

There are certain pre-defined panels that the panel manager uses that can be altered for any system; these are detailed in **Table 2 Predefined Panels**. All of these panels exist in the "%CC_ROOT%\%CC_SYSTEM%\panels\panelman" directory.

| Name | Usage |
|---|---|
| pm_app_1024.bncs_ui & pm_app_1280.bncs_ui | Menu-bar. This can be altered, typically to add to the menu bar a script control to act as alarm indicator. |
| pm_blank.bncs_ui | Blank panel brought up when panel manager starts. |
| pm_confirm_exit.bncs_ui | Optional panel that is called when the user wants to exit. It gives a Yes/No option to confirm closing panel manager. |
| pm_popup_pin.bncs_ui | PIN entry panel to access the engineering menu. |
| pm_start.bncs_ui | Panel that is shown when the "Start" button is pressed. |
| pm_status_1024.bncs_ui & pm_status_1280.bncs_ui | Panel seen at the top-left hand corner of the screen. Typically this will show system information that needs to be seen at all times. |
| pm_title_1024.bncs_ui & pm_title_1024.bncs_ui | Panel seen at the top-right hand corner of the screen where the current panel title is shown. This is also where the clock is shown. |
| pm_wait.bncs_ui | Optional panel displayed while a panel is loading. This is normally only appropriate when there is a panel that has many controls and hence takes a while to load. |

**Table 2 Predefined Panels**

## 6.7 Calling panels

Panels can be activated programmatically in two different ways, described below. In both cases the command is written to the workstation slot on infodriver 998, which allows one workstation to invoke an application on another workstation. When writing script controls or panels there are helper functions for calling other panels on the local workstation, these are also detailed below.

### 6.7.1 Panel Execute (PX)

To call a specific panel either write "PX applicationName" directly to infodriver 998 or from a script panel use the helper function navigateExecute("applicationName"). The requested panel will then be loaded and brought to the front of the panel stack for the user to use.

### 6.7.2 Panel Adjust (PA)

Panel adjust allows the user to request that the panel to change settings on a specific device to be made active. The system then decides what the correct panel is and activates it, thus the calling panel does not need to know what type of panel is being called. The command syntax is similar to Panel Execute, either write "PA instanceName" directly to infodriver 998 or from a script panel use the helper function navigateAdjust("instanceName")

# 7 Applications

## 7.1 Runtime applications

These are the main applications that are used when a v4.5 Colledia Control™ system is running in a standard system. Note that these may well call other applications at start-up and this list does not include drivers and automatics for a specific system.

| Name | Description |
|---|---|
| AlarmControl.exe | Main alarm application |
| AutoHost.exe | Runs script panels as automatics. This saves having to write a full automatic for a simple action |
| Bncs_ws_man.exe | Manages the workstation from start-up to shut down, including updates. |
| Panelman.exe | Hosts all of the UI panels. |
| V4CSI.exe | Network interface for all applications on the workstation. |
| V4grd.exe | Generic router driver. |
| V4infDrv.exe | Infodriver. |

## 7.2 Tools

This section describes the tools that a developer will use to create, maintain, edit and test a Colledia Control™ v4.5 system.

| Name | Description |
|---|---|
| bCommand.exe | Utility to send and receive commands on the network. |
| Bncs_inst_env.exe | Tool to allow changing of various workstation settings. Including system, server and workstation number. |
| Bncs_vis_ed.exe | Application to create and edit panels. |
| Config_editor.exe | Configuration tool client to allow a system maintainer to change the configuration of a system. |
| Config_server.exe | Allows the configuration client to get the files to edit, ensuring only one person can do this at a time. |
| Dosnapshot.exe | Creates a snapshot, as a zip file, of all of the currently active system for backup. |
| Paramtester.exe | Tool to allow a tester to change values of any parameter of any instance of any device. |
| Sql_script.exe | Tool to create and run SQL scripts, primarily to maintain the historical database used by the alarm system. |
| Wiz.exe | Utility to create various new modules including script controls. |

## 7.3 Utilities

This section describes the utilities that are used in a Colledia Control™ v4.5 system, most of which the developer will not normally use but are called by other applications.

| Name | Description |
|---|---|
| Bncs_filesync.exe | Synchronises files with a server. Internal application, do not run! |
| Bncs_filesynclist.exe | Generates a list of files in a given path. Internal application, do not run! |
| CC_Sys_Check.exe | Application to update or remove files, used when upgrading a system. Called by the installer. |
| CC_WS_Manager_update.exe | Updates workstation manager. Internal application, do not run! |
| Db_loc_blast.exe | Takes the location information from the instance.xml file and writes it to a database for use with the historical log. |
| DesktopBmp.exe | Takes a bitmap, adds workstation information and sets it as the desktop bitmap. |
| setCCEnvSetting.exe | Tool for setting environment variables. Internal application, do not run! |
| Setlana.exe | Tool to set the lana number the Colledia Control™ system will use. |
| WritePrivateProfile.exe | Command line utility to amend ini files. For example used on a test system to set simulation flags to 1. |

# 8 Alarm system

The alarm system is a programmable logic engine mainly used for alarms although; it can be used for other logic functionality such as working out if a source is live having gone through various switches.

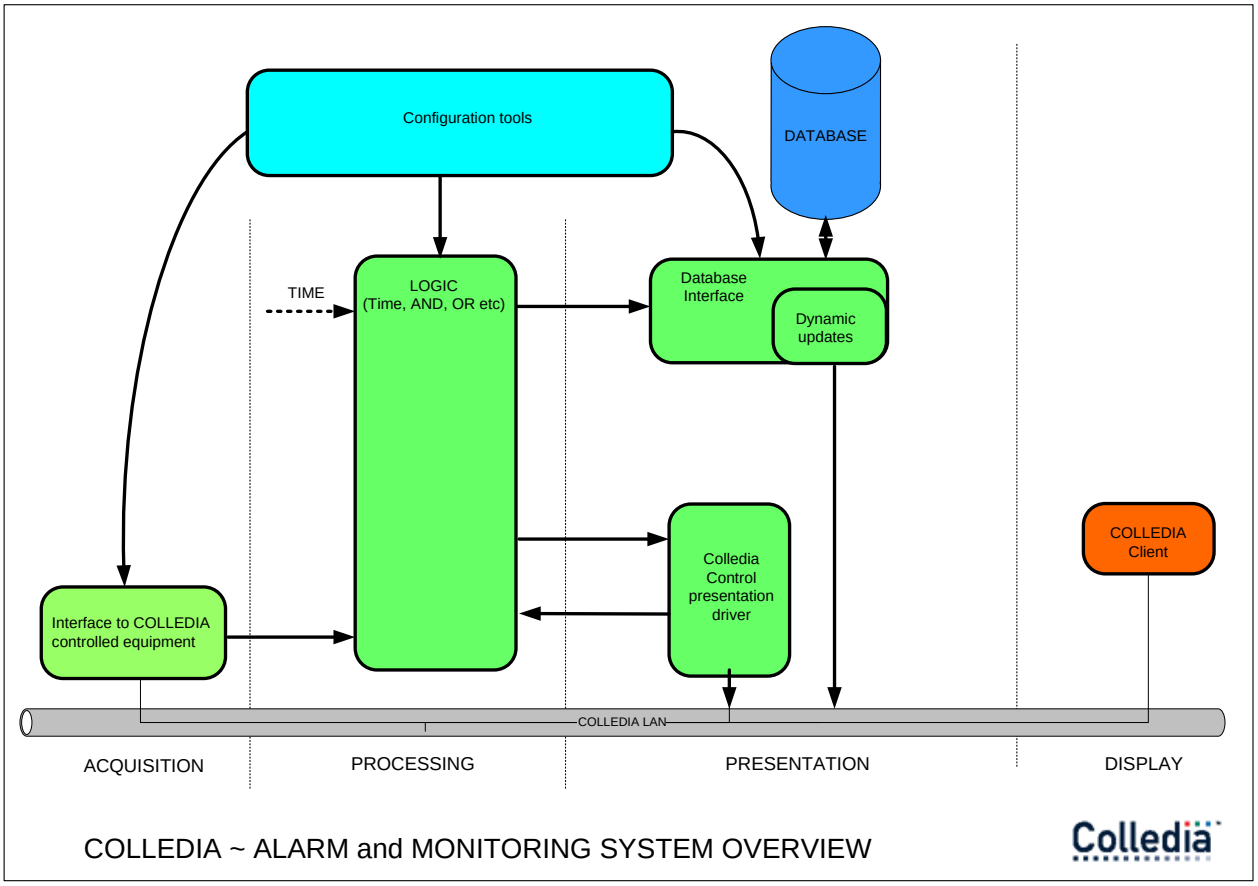


**Figure 6 Alarm overview**

The alarm system is designed to be extendable. It is has the structure shown above; a main logic block which is an executable and input and output modules which are separate DLLs. The configuration is done using the standard Colledia Control™ v4.5 configuration tools. The whole system has been designed so that with appropriate input and output modules there is no reliance on a full Colledia Control™ solution.

## 8.1 Inputs

Currently there is only a single input module; others could be produced that take data directly from hardware.

### 8.1.1 Colledia Control™ acquisition

This module gathers information from any driver on the Colledia Control™ network, performing any preconditioning to produce an OK or fail; this includes filtering out short term faults if required.

## 8.2 Processing

The processing takes the form of as many individual processes as required to perform the logic required. Each process can be an

- OR gate.
- AND gate.
- truth table.
- timed event, pre-programmed. Not from a varying schedule.
- hold-off.

Each process gate type can also be set to be latching so that fleeting alarms can be captured.

The state of each process can be

- On or off. These are normal states.
- Forced on or forced off. These states are set manually.
- Ignored. This is forced off until the alarm clears at which point it goes to off, this can be considered as ignore this instance of the alarm, but tell me next time it happens.
- Latched. The alarm occurred but has now cleared.
- Acknowledged. The user acknowledged the alarm, and it is still active.
- Unknown.

## 8.3 Outputs

Currently there are only two output modules, but others could be produced that report alarms to SNMP managers, SMS text systems or email groups of people.

### 8.3.1 Colledia Control™

This module outputs the results back to the Colledia Control™ network for use on any clients that need the data. The module also allows clients to interact with the alarm system by forcing and acknowledging alarms.

### 8.3.2 Database

This module writes the current state of all processes to a database table as well as logging to a historical log, both database tables are on a standard SQL database, either a Microsoft SQL Server database or MySql database.

## 8.4 Client controls

There is a special control, for use with the database, which provides a log display that can be configured to show various views of the data, both historical and live. As this control needs to display historical information as well as live information it will talk directly to the database server as well as using the standard messaging via CSI.

# 9 Appendix

## 9.1 Developer mode

Setting ‘developer mode’ on a workstation will:-

- Confirm update from the server before starting it, unless started in quiet mode.
- Prevent the workstation manager re-starting the workstation if CSI crashes.

## 9.2 Example batch files

### 9.2.1 Post sync batch file

```
echo off

rem ***** DO NOT CHANGE ANYTHING ABOVE THIS LINE!!! *****

echo Copying WORKSTATION specific ini files...

xcopy
"%CC_ROOT%\%CC_SYSTEM%\CONFIG\WS\%CC_WORKSTATION%\bncs_system.ini"
"%CC_ROOT%\%CC_SYSTEM%\CONFIG\system" /c/i/f/y


IF
EXIST %CC_ROOT%\%CC_SYSTEM%\CONFIG\WS\%CC_WORKSTATION%\wspostsync
.bat
CALL %CC_ROOT%\%CC_SYSTEM%\CONFIG\WS\%CC_WORKSTATION%\wspostsync.
bat


desktopbmp.exe
```

**Figure 7 PostSync batch file**

The example ‘postsync.bat’ in **Figure 7 PostSync batch file** copies the bncs_system.ini file for the local workstation to the correct location, then tries to run the ‘wspostsync.bat’ file if it exists and then sets the desktop bitmap. In the example shown ‘wspostsync.bat’ could be used to copy device configuration files for v2 drivers to use on selected workstations only.

Document Control

Document Review

| Name | Role | Date |
|---|---|---|
| | | |

Change History

| Version | Date | Description of Changes | Approval |
|---|---|---|---|
| 0.2 | 26th Feb 2007 | Removed some highlighting, paginated the document slightly better | |

Distribution

| Name | Organisation | Role |
|---|---|---|
| | | |

Referenced Documents

| No | Document | Available from |
|---|---|---|
| | | |