

# Kiwi CatTools

**Manage configurations on routers, switches, firewalls and other devices**

---

*by SolarWinds Inc.*

*Kiwi CatTools has the following features:*

*Standard edition runs on Windows 98/ME/2000/XP/2003/Vista/Windows 7 & 8/  
Windows Server 2008 & 2012.*

*Service edition runs on Windows 2000/XP/2003/Vista/Windows 7 & 8/ Windows  
Server 2008 & 2012*

*Schedule based operation with "Run Now" option*

*Encrypted database*

*Optional password protection for program and device database access*

*Telnet, SSH1, SSH1.5, SSH2 connectivity*

*"Connect via" intermediary devices. Telnet and SSH supported on Cisco, HP,  
Enterasys, CatalystCatOS, CatalystIOS, Linux devices.*

*Bulk import/export of devices via tab delimited text files*

*Built-in report viewing window*

*Multi-threaded operation for faster backups and data gathering*

*Fully scriptable. Write your own custom activities and reports using VBscript*

# Table of Contents

Foreword .....	0
<b>Part I CatTools .....</b>	<b>1</b>
1 What is CatTools .....	1
2 Installation .....	4
3 Getting started .....	4
4 Using CatTools on a VM .....	6
5 Feedback - Comments or Bugs .....	6
6 Software License Agreement .....	6
<b>Part II Application or Service? .....</b>	<b>6</b>
1 Installing CatTools as a Service .....	6
Automatic Service startup .....	7
How to add Service Dependencies .....	7
What is a Service? .....	9
What is the CatTools Manager? .....	10
Accessing UNC and Mapped drives .....	10
2 Installing CatTools as an Application .....	10
<b>Part III Devices .....</b>	<b>10</b>
1 Adding devices to CatTools .....	11
2 Creating a custom device .....	11
How to create a custom device .....	12
The custom device type file (.ini) .....	13
The custom device script file (.txt) .....	20
Custom device script cl. variables and functions .....	24
Testing your custom device .....	34
3 Device specific information .....	35
Device Variations .....	35
Variations GUI.....	40
Dell devices .....	43
ASA dap.xml file .....	43
HP2500 series switches .....	44
3Com superstack switches .....	44
Connecting via a Cisco Terminal Server .....	46
Connecting via a session .....	47
SSH2 for Cisco and Netscreen .....	49
Cisco VPN .....	49
Cisco WAAS .....	49
Backing up device via SFTP/SCP .....	50
4 Unsupported devices or device activity .....	50
<b>Part IV Activities .....</b>	<b>51</b>
1 Overview .....	51

<b>2 The Add/Edit scheduled activity details form .....</b>	<b>51</b>
<b>Activity tab .....</b>	<b>52</b>
<b>Time tab .....</b>	<b>53</b>
<b>Devices tab .....</b>	<b>53</b>
<b>Email tab .....</b>	<b>54</b>
<b>Options tab .....</b>	<b>55</b>
<b>3 Activities list .....</b>	<b>56</b>
<b>DB.Backup.CatTools .....</b>	<b>56</b>
Setting up the activity.....	56
<b>DB.UpdateDevice.Password field .....</b>	<b>56</b>
Setting up the activity.....	56
<b>DB.UpdateDevice.Text field .....</b>	<b>57</b>
Setting up the activity.....	57
<b>Device.Backup.Running Config .....</b>	<b>58</b>
Setting up the activity.....	58
Why you should backup your configs.....	61
<b>Device.Backup.TFTP .....</b>	<b>61</b>
Setting up the activity.....	62
<b>Device.CLI.Modify Config .....</b>	<b>64</b>
Setting up the activity.....	65
When to use the Device.CLI.Modify Config activity.....	66
How to: Enter different commands onto each device.....	67
<b>Device.CLI.Send commands .....</b>	<b>68</b>
Setting up the activity.....	68
Running External Scripts.....	70
Example Generic Script.....	72
How to: Enable/Disable ports on a Catalyst CatOS switch.....	74
How to: Backup Cisco IOS image to TFTP server.....	75
How to: Download Cisco IOS running config to TFTP server.....	75
How to: Download Cisco IOS running config to SCP server.....	76
How to: Upload Cisco IOS config from TFTP to device.....	77
How to: Upgrade Cisco IOS via TFTP.....	77
How to: Enter different commands onto each device.....	78
How to: Save output from multiple devices into a single file.....	79
<b>Device.ConnectivityTest.Login .....</b>	<b>79</b>
Setting up the activity.....	80
<b>Device.ConnectivityTest.Ping .....</b>	<b>80</b>
Setting up the activity.....	81
<b>Device.InterDevice.Ping .....</b>	<b>81</b>
Setting up the activity.....	82
When to use the Device.InterDevice.Ping.....	83
<b>Device.TFTP.Upload Config .....</b>	<b>83</b>
Setting up the activity.....	83
Devices supported.....	84
<b>Device.Update.Banner .....</b>	<b>84</b>
Setting up the activity.....	84
Devices supported.....	85
<b>Device.Update.Password .....</b>	<b>85</b>
Setting up the activity.....	85
Devices supported.....	86
<b>Report.ARP table .....</b>	<b>86</b>
Setting up the activity.....	86
<b>Report.CDP Neighbors table .....</b>	<b>87</b>

Setting up the activity.....	87
Devices supported.....	88
<b>Report.Compare.Running Startup .....</b>	<b>88</b>
Setting up the activity.....	88
Devices supported.....	89
<b>Report.Compare.Two files .....</b>	<b>89</b>
Setting up the activity.....	90
<b>Report.Error info table .....</b>	<b>91</b>
Setting up the activity.....	91
Devices supported.....	92
<b>Report.MAC address table .....</b>	<b>92</b>
Setting up the activity.....	92
<b>Report.Port info table .....</b>	<b>93</b>
Setting up the activity.....	93
<b>Report.SNMP.System summary .....</b>	<b>93</b>
Setting up the activity.....	94
Devices supported.....	94
<b>Report.Version table .....</b>	<b>94</b>
Setting up the activity.....	95
<b>Report.X-Ref.Port MAC ARP .....</b>	<b>95</b>
Setting up the activity.....	95
<b>System.File.Delete .....</b>	<b>96</b>
Setting up the activity.....	96
Troubleshooting.....	97
<b>4 Internal functions .....</b>	<b>98</b>
Ignore Text .....	98
Filename Variables .....	99
Meta Data .....	100
Meta Commands.....	100
%ctDB Command.....	101
Device Table .....	101
%ctUM .....	102
Meta Variables .....	103
<b>5 Creating a custom activity .....</b>	<b>104</b>
How to create a custom activity .....	105
The custom activity type file (.ini) .....	107
The custom activity main script file (.txt) .....	111
The custom activity client script file (.txt) .....	113
The custom activity device script file (.custom) .....	116
Custom activity scripts cl. and ct. variables and functions .....	118
Client script - cl. variables and functions.....	119
Main script - ct. variables and functions.....	123
Testing your custom activity .....	128
<b>6 Unsupported activities for a device .....</b>	<b>130</b>

## Part V Menus

130

<b>1 File .....</b>	<b>131</b>
Database .....	131
Export .....	131
Export devices to tab delimited file.....	131
Export activities to tab delimited file.....	131
Import .....	131

Import devices from tab delimited file.....	131
Import schedules from tab delimited file.....	135
Change encryption passw ord.....	136
Backup current database.....	136
Restore database from backup.....	136
Squeeze current database.....	137
Open other database.....	137
Create new database.....	137
Delete .....	137
Delete log file.....	137
Enable capture mode .....	137
Debug .....	138
Create diagnostics for Technical Support.....	138
Exit .....	140
<b>2 View .....</b>	<b>141</b>
View Reports folder .....	141
View Captured Data folder .....	141
View Configs folder .....	141
View Email log .....	141
View Activity log .....	141
View Info log .....	142
<b>3 Options .....</b>	<b>142</b>
Setup .....	142
Email .....	142
General Options.....	142
Primary Mail Server Properties.....	143
Secondary Mail Server Properties.....	144
Logging .....	145
Misc .....	146
TFTP Server.....	147
General Options.....	147
Security Options.....	148
Scripting Options.....	149
DNS Resolver.....	151
Device Wizard .....	151
Activities Wizard .....	151
<b>4 Interface .....</b>	<b>151</b>
Panels .....	151
Themes .....	152
<b>5 Help .....</b>	<b>153</b>
Contents .....	153
Online FAQ .....	153
Purchase CatTools .....	153
Enter registration details .....	153
About .....	153

## Part VI Panes 154

<b>1 Overview .....</b>	<b>154</b>
<b>2 Devices .....</b>	<b>158</b>
Add .....	158
Device info.....	160
Passw ords.....	161

Prompts .....	161
Contact info.....	162
Extra info.....	162
Remove .....	163
Edit .....	163
Copy .....	163
Filter .....	163
Show All .....	165
Select All Devices .....	165
<b>3 Activities .....</b>	<b>166</b>
Add .....	166
Edit .....	166
Copy .....	167
Run now .....	167
Start timer .....	167
Remove .....	167
<b>4 Activity log .....</b>	<b>167</b>
Clear .....	167
View report .....	167
<b>5 Compare .....</b>	<b>167</b>
Save .....	167
Clear .....	167
Select .....	168
<b>6 Info log .....</b>	<b>168</b>
View Filter Drop-Down .....	168
Clear .....	168
<b>7 Report .....</b>	<b>168</b>
Open .....	168
Save .....	168
Clear .....	169
Delete .....	169
Refresh .....	169
<b>8 TFTP .....</b>	<b>169</b>
Start .....	169
Stop .....	169
Clear .....	169
<b>9 Display .....</b>	<b>169</b>
Clear .....	169
<b>10 Mail .....</b>	<b>169</b>

## **Part VII Notification 170**

1 Email .....	170
---------------	-----

## **Part VIII Custom scripting 170**

1 Variables .....	171
2 Functions .....	176
cl.Log .....	179
3 Script editors .....	179

<b>Part IX Troubleshooting</b>	<b>180</b>
1 Device specific .....	180
2 "Error: 70 Permission denied" message in Info Log .....	180
3 Reporting problems .....	181
4 Remote Desktop Systems .....	181
5 Remote Authentication .....	182
6 Anti Virus Tools .....	182
7 XP Firewall .....	182
8 The service is running but nothing is scheduled .....	182
<b>Part X API</b>	<b>183</b>
1 Environments .....	183
2 Classes .....	184
Database .....	184
Sample Code.....	185
Devices .....	185
Sample Code.....	186
Device .....	186
Sample Code.....	187
DeviceTypes .....	189
Sample Code.....	189
DeviceType .....	190
Sample Code.....	190
Groups .....	190
Sample Code.....	190
Group .....	191
Sample Code.....	191
3 Limitations .....	191
<b>Part XI Installation</b>	<b>192</b>
1 Automating the installation of CatTools .....	192
2 Problems after installing or upgrading .....	192
<b>Index</b>	<b>194</b>

# 1 CatTools



**CatTools** - Configuration Automation Tracking Tools

Program copyright 1996 - 2013 SolarWinds. All rights reserved.

**Website:**

<http://www.kiwisyslog.com>

**Support:**

<http://www.kiwisyslog.com/support>

**Get started:**

[getting started](#)

**FAQs and Online Help:**

<http://www.kiwisyslog.com/support>

**Evaluate the licensed version:**

<http://www.kiwisyslog.com/downloads/registration.aspx?productType=ct&AppID=881&CampaignID=70150000000Es8Y>

**Purchase a license:**

<http://www.kiwisyslog.com/purchase>

**Full release history:**

<http://www.kiwisyslog.com/products/kiwi-cattools/whats-new.aspx>

## 1.1 What is CatTools

CatTools is an application that provides **automated device configuration management** on routers, switches and firewalls.

Support is provided for Cisco / 3Com / Dell / Enterasys / Extreme / Foundry / HP / Junpier / Nortel devices and many more.

Some of the many tasks CatTools perform to make your life easier are:

- Perform configuration backups and have any differences instantly e-mailed to you.
- Issue commands via Telnet or SSH to many devices at once.
- Change the configuration at scheduled times.
- Change all your network device passwords in one go.

This configuration management tool is also fully scriptable, has a built-in TFTP server, supports SSH, Telnet and more.

CatTools can be run on the following operating systems:



- Windows 2000
- Windows XP
- Windows 2003 / R2
- Windows Vista
- Windows 7
- Windows 8
- Windows 2008 / R2
- Windows 2012

CatTools is available in two separate versions, an Application and a Service version. Both versions are available within the single installation package.

The **Application** version runs interactively and only operates while a user is logged on to the system.

The **Service** version runs as an automatic NT service. This version does not require a user to be logged on to operate.

#### **CatTools has the following features:**

- Run CatTools as a standard application on Windows 2000/XP/2003/Vista/Windows 7/Windows 8/Windows 2008 Server
- Run CatTools as a service on Windows 2000/XP/2003/Vista/Windows 7/Windows 8/Windows 2008 Server
- Schedule based operation with "Run Now" option
- Encrypted sensitive database fields
- Optional password protection for program access
- Telnet, SSH1, SSH1.5, SSH2 connectivity
- "Connect via" intermediary devices.
- Bulk import/export of devices and activities via tab delimited text files
- Built-in report viewing window
- Multi-threaded operation for faster backups and data gathering
- Built-in multi threaded TFTP Server
- Device template script. Write your own custom device types using VBScript

#### **Activities Supported: \***

- DB.UpdateDevice.Password Field (updates password fields in the CatTools database Device table)
- DB.UpdateDevice.Text Field (updates any of the clear text fields in the CatTools database Device table)
- Device.Backup.Running Config (makes a backup of the running config and compares it to the last one stored on disk. Includes a diff report)
- Device.CLI.Modify Config (enter commands into the running config)
- Device.CLI.Send commands (enter commands in privileged mode, and capture the output to file)
- Device.ConnectivityTest.Login (login to each device and enter enable mode)
- Device.ConnectivityTest.Ping (ping each device and return the round trip statistics)
- Device.InterDevice.Ping (pings a series of addresses from each device, or pings all other devices in list from each device)
- Device.TFTP.Upload Config (upload a text config file to a device)
- Device.Update.Banner (allows you to apply a banner to your device)
- Device.Update.Password (enables you to change passwords for certain devices)

- Report.ARP table (builds a report of ARP table entries and tracks changes)
- Report.CDP Neighbors table (builds a report of neighboring devices)
- Report.Compare.Running Startup (compares the running and the startup configs of your devices and reports on the differences found)
- Report.Compare.Two files (runs a compare against two files which you define and reports on the differences it finds)
- Report.Error info table (builds a report of error counters for many Cisco devices)
- Report.MAC address table (builds a list of all MAC addresses on the network)
- Report.Port info table (builds a report of the interface configuration and state of the devices)
- Report.SNMP.System summary (builds a summary report of device information gathered via SNMP)
- Report.Version table (builds a report of device serial numbers, hardware and software versions)
- Report.X-Ref.Port MAC ARP (builds a cross reference of the Port/MAC/ARP reports)

**Devices supported: \***

CatTools currently supports a number of different devices manufactured by:

- Cisco (routers, IOS & CatOS switches, firewalls, etc.)
- 3Com
- Dell
- Enterasys
- Extreme
- Foundry
- HP
- Juniper
- Nortel
- ... and more! For a current list of devices types, please see our [device matrix](#).

See the [chapter](#) on Device specific information for more details regarding some device types.

\* Additional device types, reports and activities are being added all the time. Be sure to check the [website](#) to ensure you have the latest version.

CatTools will work for the most common and up to date CLI syntax for the devices listed above. If you have problems getting your device to work properly within CatTools, it is possible your device is not yet supported (or maybe just not for the activity you are trying to run). It may also be that your device is using a different CLI syntax to that expected by CatTools. In either case, please inquire via [thwack](#), the SolarWinds online community site for further information.

**Activity and Device limitations: (Edition differences)****The Freeware edition of CatTools supports:**

- 1 device in the database
- Up to 20 scheduled activities
- Up to 2 simultaneous TFTP sessions
- 1 client thread

**The Enterprise edition of CatTools supports:**

- Unlimited devices in the database - subject to file size limitations
- Unlimited scheduled activities - subject to file size limitations
- Up to 100 simultaneous TFTP sessions
- Up to 30 client threads

**How to purchase the Enterprise edition:**

Remember that the Freeware edition is free to use for as long as you like without having to pay anything.

If you would like to work with more devices, the Enterprise edition can be purchased via the [on-line ordering page](#).

## 1.2 Installation

When installing CatTools you will be prompted to choose between installing as an [Application](#) or as a [Service](#).

For further information on each of these options please see the [Application or Service](#) section.

## 1.3 Getting started

**Overview:**

You can get started with CatTools in just four simple steps:

1. Define your configuration options such as email using the Setup Menu. (Using the Options/Setup menu)
2. Enter the details for at least one network device. (Using the Devices tab)
3. Create an Activity and associate you device(s) to it. (Using the Activities tab)
4. Run the Activity. This can be accomplished using either the "Run now" button or the scheduler.

More details on each of these steps follows:

**Setting the CatTools configuration options:**

- Click on the Options | Setup menu.
- Set the [e-mail options](#)
  - Enter the e-mail address you want notified of errors and reports into the respective fields.
  - Enter your e-mail address into the From address field.
- Set the details of your main SMTP mail server.
- If your server requires authentication, enter the details (most servers do not require this).
- Send a test e-mail by pressing the Test button.
- Defaults are fine for the rest of the settings while getting started.

**Entering devices into the database:**

- Click the "Devices" pane.
- Click the "Add" button.
- Fill in each of the required fields. When a field has focus, the description of each field is displayed in the status bar (at the bottom of the "Device Information" window).
- Fields marked with an asterisk (\*) are required fields and must be filled in.

**Creating a scheduled Activity:**

- Click the "Activities" pane.
- Click the "Add" button.
- Select the type of activity to perform from the drop down list (e.g. [Device.Backup.Running Config](#)).
- Give the activity a meaningful unique name and description.
- Choose the persistence of the activity (e.g. Permanent or Run once then deactivate).
- The report file and e-mail addresses can usually be left as default.
- Click the "Time" tab and set the time you want the activity to run. Leaving the time as "Never" means that you must initiate it manually using the "Run now" button to activate the activity.
- If required, you can save your activity to the favorites database. This saves the Time part of the schedule, so you can recall it for use on other activities.
- Click the "Devices" tab and check all the devices you want to connect to for this activity.
- Click the "Options" tab and set any specific options that relate to this activity. In most cases, the options can be left as default.

**Running a scheduled Activity:**

- Click the "Activities" pane.
- Check (tick) all the Activities you would like to schedule.
- Press the "Start timer" button.
- CatTools will enter into "Active Timer Mode" and run the activities at the scheduled intervals and/or times.

**Running an Activity immediately:**

- Click the "Activities" pane.
- Select the activity you would like to run immediately.
- Press the "Run now" button (alternatively, right-click the activity and click on "Run now" in the drop down list).
- CatTools will run the selected activity and then return to idle mode.

**Monitoring the client Activity:**

- Click the "Info log" pane.
- Select the message level you want to see from the drop down list (Info messages only).
- View the messages as they arrive from the clients and main program.
- Last message received is shown at the top of the list.
- You can resort the info log entries by clicking on one of the column headers within the Info log pane.

**Viewing the results:**

- Click the "Activity log" pane.
- View the results of the activities as they finish.

**Viewing a report:**

- Select the "Report" pane, click the Open button and select the text report you wish to view in the report grid.

**Viewing captured data and config files:**

- Click the "View" menu and choose the folder to view.
- CatTools will open a Windows explorer session to the selected folder.

- Navigate the folder structure and view the desired files.

## 1.4 Using CatTools on a VM

### Overview:

CatTools will run on VM systems but you may need to adjust the setup of the VM in certain circumstances.

### Setting the number of processors

You can configure VM's to use multiple 'virtual' CPU's, (based on the number of physical CPU's you have on your server). However, because CatTools spawns multiple clients this can actually slow down the operation of CatTools while the VM waits for physical processors to become simultaneously free. For best results you should therefore ensure that there is only 1 virtual CPU configured for your VM.

## 1.5 Feedback - Comments or Bugs

We welcome your comments and suggestions on how we can improve the program.

Please contact us via [thwack](#), the SolarWinds online community site.

## 1.6 Software License Agreement

Please see the link below for details;

<http://www.kiwisyslog.com/kiwi-cattools-end-user-license-agreement/>

## 2 Application or Service?

CatTools can be installed as either an Application or a Service.

[The Service version](#) installs CatTools as a Windows service, allowing the program to run without the need for a user to log in to Windows. This option also installs the CatTools Manager which is used to control the service.

[The Application version](#) installs CatTools as a typical Windows application requiring a user to log in to Windows before running the program.

### 2.1 Installing CatTools as a Service

The Service version installs CatTools as a Windows service, allowing the program to run without the need for a user to log in to Windows. This option also installs the [CatTools Manager](#) which is used to manage your devices, activities and activity schedules.

You can interact with the CatTools Service by using the [CatTools Manager](#). This can be thought of as the user interface (GUI) for the Service.

It is important that the user account that CatTools is installed to "run as" has sufficient privileges to perform the tasks necessary for CatTools to operate. In most cases the Local System account is sufficient, however if you have difficulties getting CatTools to run as a Service, this user account is the first place to start looking.

We would recommend the Service installation to Enterprise customers as well as anyone who needs to run unattended Activities, or who need to make use of around the clock scheduling. This option would be the preferred method for anyone who is dedicating a machine specifically for CatTools to run on.

### 2.1.1 Automatic Service startup

When the CatTools Service is installed into the system/server, it is set to startup automatically. This means that when the system/server is restarted or rebooted, the CatTools Service will automatically start. This occurs even if no user is logged into the console.

#### Service Dependencies (particularly Windows 2000 Server and Windows 2003 Server)

Under most operating systems, the CatTools Service will start without problems. On some Windows 2000/2003 Server systems, the CatTools Service may have to wait for some other system Services to start before it can start.

A Service Dependency may be an issue if you see one of the following after a reboot:

- an error of "One or more system services failed to start" on the console, or
- the CatTools Service fails to start even though the 'startup type' for the service is set to 'Automatic'.

To test if this is a problem on your system, check the status of the CatTools Service using the Services manager.

Click the **Start** (button)  
Click **Run**  
Type **services.msc** in the open dialog box  
Click **OK**

Find the '**CatTools**' service and check the *status* column. It should be set to 'started'. If not then it could be a service dependency issue in that the CatTools service failed to start as it was dependent on another service being started first.

For more information on service dependencies see [How to add Service Dependencies](#).

### 2.1.2 How to add Service Dependencies

There are a couple of different ways to add service dependencies:

- The first method using **sc.exe** creates a dependency specifically for an existing service. If the service is deleted (i.e. CatTools is uninstalled) then the dependency is also lost.
- The second method modifies the Windows registry (via **RegEdit**) to create a new value, listing the dependencies. This value will then be applied every time the CatTools service is created on the system (i.e. if you upgrade or reinstall CatTools to your system), therefore this is the recommended 'permanent' fix.

#### How to add service dependencies using sc.exe

This is the easiest method to add service dependencies to the existing CatTools service. However if the CatTools service is uninstalled, then the dependencies are also lost.

To add a service dependency using `sc.exe`, you can either run a command from the **Start** (button) > **Run** dialog, or you can run the **cmd.exe** from the Run dialog and enter the command in the `cmd.exe` window.

For Example:

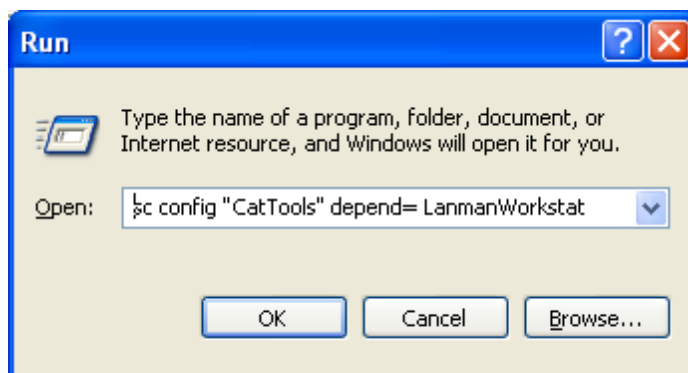
To add dependencies to the CatTools service for the LanmanWorkstation, TCPIP and WMI services you can use either method below to execute the command:

```
sc config "Cat Tools" depend= LanmanWrkst at i on/ TCPI P/ WMI
```

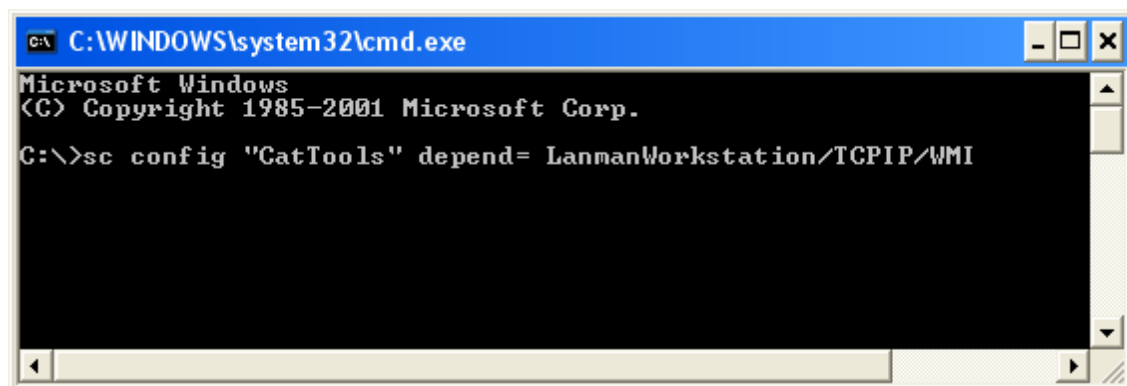
The important things to note are:

- the CatTools service must be stopped before configuring dependencies.
- the CatTools service must be contained within double quotes " " because the service name has a space within it.
- if adding multiple dependencies, separate them using a '/' (forward slash)
- the space ' ' between the 'depend=' text and the first dependency service name. If you do not include this space, although it will appear to work OK, the dependency will not be applied.

- using the 'Run' method. (Note: the full command to be executed cannot be displayed in the open box, although you can see it by clicking in the box and scrolling to the right.)



- using the `cmd.exe` method.



This example will ensure that the Workstation, WMI (Windows Management Interface) and

TCP/IP stack services are running before trying to start CatTools Service. Which service dependencies you need to add for your system depend entirely on your specific machine and what services are set to run on it at start-up.

If you need to remove dependencies, the simplest way is to uninstall CatTools, (a reboot after uninstall is preferred) and reinstall so that the modified CatTools Service is removed and replaced with the default one from within the CatTools installer.

You can use the command:

```
sc config "Cat Tools" depend= none
```

however this sometimes leave the CatTools service in an unusable state as the dependencies are not properly deleted.

If this is the case on your system, then the only solution is to uninstall CatTools, reboot the system, then reinstall CatTools again.

### How to add service dependencies using RegEdit

This method is the preferred solution as it then ensures that the CatTools service dependencies are always applied every time the service is created, therefore you don't need to run through the sc.exe method whenever you reinstall or upgrade CatTools.

To ensure that the required services have started before CatTools, you will need to modify the registry setting as follows:

**Section:** HKEY\_LOCAL\_MACHINE\SOFTWARE\Kiwi Enterprises\CatTools3

**Value (STRING):** NTServiceDependencies

**Default value:** Blank

**Type:** Text string of service names. Delimited by semi-colons.

E.g. ServiceName1;ServiceName2;ServiceName3

#### Step by step example:

- Uninstall CatTools
- Run **RegEdit**
- Locate the section **HKEY\_LOCAL\_MACHINE\SOFTWARE\Kiwi Enterprises\CatTools3**
- Create the new string value of **NTServiceDependencies**
- Modify the value data to include the list of services that need to start first , e.g. "LanmanWorkstation;TCP/IP;WMI" (without the quotes)
- Install CatTools again

The example above will ensure that the Workstation, WMI (Windows Management Interface) and TCP/IP stack services are running before trying to start the CatTools Service.

### 2.1.3 What is a Service?

A Service is a program that interacts with the machine according to different rules from a standard application. A standard application is able to run according to the limitations of the



user who is logged in at the time. A service by default runs under a special account known as the local system account, which normally has many restrictions in terms of file permissions and the like.

A service runs when the machine is powered on, you do not need to be logged in for the service to run.

#### 2.1.4 What is the CatTools Manager?

The Manager is the CatTools interface that is used to talk to (or manage) the CatTools Service.

If you like, the Application version has been split into two parts:

1. The Service is the part that runs in the background processing the data and runs the Activities for you.
2. The Manager is the part you see on your screen and interact with to manage your devices and activities.

The Manager interface takes your request and passes it on to the Service. The Service processes it and returns a reply which the Manager then interprets and displays to you.

#### 2.1.5 Accessing UNC and Mapped drives

By default the Service version is logged in using the "LocalSystem" account.

This account is different to a user account you may be accessing the machine running CatTools with. It does not know about any mapped drives that are defined and are accessible by your user account.

We recommend that where possible, any Activities run by the CatTools Service that need to access a network location to read or previously stored data do so by using full UNC path names, otherwise you may find your activities report incorrect results; for example: if a backup activity always returns a status of 'Configuration is New' every time you run the activity, it is likely that the CatTools Service cannot access the network location defined in the activity setup for the current configuration file.

## 2.2 Installing CatTools as an Application

The Application version installs CatTools as a typical Windows application requiring a user to log in to Windows before running the program.

We would recommend the Application installation to anyone who needs to run one off or attended Activities. This option would also be best for infrequent or part-time users who will be running CatTools on their own personal computers.

## 3 Devices

CatTools supports a wide range of different manufacturer and model devices.

For an up to date list of the devices currently supported and the activities supported for each device type, please see the [device matrix](#) on our [web site](#). For more information on adding a device using a pre-defined device type, please see the [Devices | Add](#) section in the [Panels](#) chapter.

- [Creating a custom device](#) - how to create your own custom device script and add it to the CatTools device type list
- [Device specific information](#) - important information with regards to specific devices
- [Unsupported devices or device activity](#) - information and links

## 3.1 Adding devices to CatTools

Information on how to add your devices to CatTools can be found within the following help file page:

- [Add](#) within the [Devices](#) sub-chapter of the [Panels](#) chapter.
- You can also use the [Device Wizard](#) on the [Options](#) menu to add devices.

## 3.2 Creating a custom device

CatTools provides a facility to create your own custom device type and script files should your device not be supported by one of the pre-defined device types.

### Pre-requisites

A reasonable understanding or experience of Visual Basic Scripting is assumed in order to successfully add custom scripts to CatTools. However, the help file documentation and comments within the example code template files found in the /Templates sub folder of the CatTools root directory, should provide a reasonable level of assistance for a technically competent novice to follow.

### Overview

To add support for a device in CatTools, two files are required:

- 1) The device type file ([.ini](#) file), which defines the following:
  - device type name,
  - device ID,
  - the user interface field values and defaults which are displayed in the device form when adding or editing a device.
- 2) The device script file ([.txt](#) file), which contains device type specific code to allow CatTools to login to the device, enter and exit different modes, perform different activities (e.g. configuration backups, send CLI commands, modify configuration) and parse command output data for reports.

The device script file also contains function calls and references to variables within the internal CatTools program code. These are prefixed with 'cl.' A list of these cl. functions and variables have also been made available within this chapter to help assist in the development of your custom device script.

- [How to create a custom device](#) - a simple step-by-step guide on how to create a custom device
- [The custom device type file \(.ini\)](#) - detailed information on the .ini file and how to create one for a custom device type
- [The custom device script file \(.txt\)](#) - detailed information on the .txt file and how to create the custom device script

- [cl. variables and functions](#) - information on the CatTools internal variables and functions exposed to the custom device script file
- [Testing your custom device](#) - help and tips on testing your custom device

### 3.2.1 How to create a custom device

#### Quick reference guide to creating a custom device in CatTools

This is a simple step-by-step guide on creating a new custom device in CatTools.

It contains brief instructions on how to add a custom device type to the CatTools GUI, and how to create a custom device script file.

The custom device templates files provided by CatTools and referenced in this guide, are based on a Cisco Router device.

It is likely that further modification will be required to the device script (.txt) file once you have created it, in order to get the script to work successfully with your specific device.

Further information regarding the [custom device type file \(.ini\)](#), [custom device script file \(.txt\)](#) and [code examples](#) of how to use the CatTools internal functions in your device script, can be found in the other sub pages of this chapter.

#### STEPS TO CREATE A CUSTOM DEVICE:

##### 1) Create a custom device type file (.ini)

Take a COPY of the Custom.Device.Template.ini in the \Templates sub folder of the CatTools root directory and save to the \Devices sub folder, giving it a new file name using the syntax: *Custom.Manufacturer.Type* (e.g. Custom.Cisco.FirewallPIX).

##### 2) Edit the .ini file

Using a text file editor (such as Notepad), open the .ini file created in [step 1](#). You must change the following items in the .ini file from the template default values:

```
[device]
name=Custom.Manufacturer.Type
id=4000
```

Change 'name' item to a UNIQUE device name. Example: Custom.Cisco.Router Note: do not use spaces, use a 'period' mark (.) instead.

Change 'id' item to a UNIQUE number (i.e. one not used in any other device .ini file). Number must be within the range of 4000 to 4999.

```
[item_Name]
default=Unique device name
```

Change the 'default' item to a UNIQUE device name.

##### 3) Save & restart

Once you have made your changes to the .ini file, save it back to the \Devices sub folder and close.

Restart CatTools to populate the 'Device type' drop-down field in the device setup screen with your new custom device type.

##### 4) Create a custom device script file (.txt)

Take a COPY of the Custom.Device.Template.txt in the \Templates sub folder of the CatTools

root directory and save to the \Scripts sub folder.  
Save it with a file name using the value entered for the [device] section 'name' item in [step 2](#)) above. Ensure you retain the .txt file type suffix.

Example:

(device .ini file settings)

```
[device]
name=Custom.Cisco.FirewallPIX
```

Script file name must therefore be: *Custom.Cisco.FirewallPIX.txt*

### 5) Edit the .txt file

Using a text file editor\*, open the .txt file created in [step 4](#)). Follow the instructions in the SCRIPT NOTES section of the .txt file to begin customizing the script for your device.  
(\*note: although the .txt files can be opened and edited using 'NotePad', a syntax highlighting [script editor](#) may make reading and editing the .txt file much easier)

### 6) Save & test

Once you have made your initial changes to the .txt file, save it back to the \Scripts sub folder and close.

To test your custom device, create an activity that is supported by your custom device script and run it manually using the 'Run Now' button. ([Device.ConnectivityTest.Login](#) is a good starting point activity as every device must be able to log in successfully in order to perform any of the other more complex activities).

Check the [Info Log pane](#) for errors and edit your .txt file as necessary.

See [Testing your custom device](#) for more information and tips.

Creating custom device checklist:

- Create device type file (.ini) \_\_\_\_\_
- Create device script file (.txt) \_\_\_\_\_

## 3.2.2 The custom device type file (.ini)

### Device types in CatTools

Devices types are defined within CatTools using text files (.ini files) and are stored in the \Devices sub folder within the root CatTools directory. Each .ini file represents a separate item in the 'Device type' drop-down list field in the [Device information](#) setup form.

When CatTools starts, it searches for all the .ini files in the \Devices sub folder and reads their contents. All the device types found are then available for selection in the Device type drop-down list when defining devices.

To add a new custom device type, you need to create a new .ini file defining the device and its characteristics.

### The .ini file sections overview

The contents of an .ini file are divided up within [...] sections. The main sections inside an .ini file are as follows:

**[info]** section

This is required and identifies the .ini file as a CatTools file.

```
[info]
cookie=CatTools
version=3
author=SolarWinds
```

**[device]** section

This is required and defines the name used within the CatTools user interface (i.e. the Device type drop-down list field) and within scripting. It also defines the unique key (id) of the device type in the CatTools database.

```
[device]
name=Custom.Cisco.Router
id=4000
```

Although the file name of the .ini file may be similar (or the same) to the [device] section *name* item, it is the *name* item that defines the device name within the user interface and not the .ini file name.

CatTools reserves id's from 0 to 3999 for predefined device types. You can use the number range from 4000 to 4999 for the custom device types you create, however you must ensure the number is unique.

**[item]** sections

Each item section defines the input fields used within the CatTools user interface for the Device setup screens.

Example:

```
[item_group]
name=Group
default=Default
required=1
info="The logical group that this device belongs to."
```

Each input field in the CatTools user interface has a separate [item\_xxx] type section.

The *name* item sets the text (or label) to be displayed next to the input or selection field.

The *default* item sets the default value to be used when adding a new device to the database.

The *required* item tells the user interface if the field must contain valid data or not (for example: must be an item from within a predefined list or whether the user can enter their own values). 0=not required, 1=required.

The *info* item is the description associated with this field. This text is displayed in the status bar of the user interface when a field has focus.

The above [item\_group] section is an example of a standard text box input field within the user interface.

When the input field is to be a list box, you need to define the list box contents using a *list* item within the item section (see below).

```
[item_model]
name=Model
default=Other
required=0
info="The device model number."
list=501,515,Other
```

The *list* item contains a comma delimited string of items to populate the list box with. The *default* field specifies which item from the list to show as the default when adding a new device.

Check box input fields can only accept values of 0 or 1. Any value other than a 1 is considered a 0. A check box example is below.

The check box is defaulted to 1 (checked).

```
[item_require_vty_login]
name=Initial login requires password
default=1
required=0
info="This device requires an initial password for access"
```

The size (length of the data that can be entered) of each field is determined by the design of the CatTools database. Any data entered via the CatTools user interface that is too long for the field will be truncated accordingly.

### The custom device type .ini template

CatTools provides a starting point template file to help assist in the creation of a new custom device type.

The template file is called *Custom.Device.Template.ini* and can be found in the \Templates sub folder.

This template file is included as part of the predefined device types in CatTools. As for all of the CatTools predefined device types, it may (as and when required) be subject to updates and modifications with each new release of the CatTools product.

For this reason, you should never modify the template file itself in order to create a new custom device type.

If you need to create a new custom device type, take a copy the template file and save it to the \Devices sub folder giving it a new file name.

### Creating your custom device type .ini file

If adding a new custom device type to CatTools, the first step will therefore be to take a copy of the template file *Custom.Device.Template.ini* in the \Templates sub folder and save it to the \Devices sub folder giving it a new file name.

This will give you a new starting point device type file to work with. When naming the file, try to use the naming convention:

*Custom.Manufacturer.Type*

*Custom* being the text "Custom" to distinguish custom device types from CatTools predefined types.

They will also be grouped together in the 'Device type' drop-down list field

*Manufacturer* the manufacturer or supplier of the device (Cisco, Nortel, 3Com, Juniper, Foundry, Enterasys, etc.)

*Type* the type of device (router, switch, firewall, etc.)\*

\* It is not recommended you specify the exact Model number for the device as the *Type*, as you may be able to reuse your script for different models of the same *Manufacturer* and *Type*. For example: PIX model 501 and Cisco PIX model 515 can use a script '*Custom.Cisco.Firewall*' or '*Custom.Cisco.PIX*' or even '*Custom.Cisco.FirewallPIX*'. The different Models of similar device types can be specified within the device .ini file in the [item\_Model] section, or alternatively you can add the Model to the Model drop-down list field at runtime.

*Note:* The Model field values in the CatTools predefined device types have no impact on the way the associated device script file ([.txt](#)) behaves.

Although it is recommended you follow this naming standard, the idea of the [custom device script file](#) is to give the user as much flexibility as possible while remaining within the limits of the application. Therefore if you need to have model specific behaviour within your custom device script, then you will need to code the device script file accordingly.

### Editing the .ini file

The next step is to open the custom device .ini file and make the required changes to the values within each section.

Below is an example of an .ini file with a number of items values highlighted.

The most important item values which *MUST* be changed are **highlighted in Red**. These items **must be unique** otherwise your device type may not appear in the CatTools 'Device type' drop-down list field.

Items values which *may* be changed if desired are **highlighted in Green**.

Additional comments are **highlighted in Blue** (*Note:* do not include any comments in your .ini file)

It is recommended that anything else that is not highlighted be left at its original setting. You can however, customise some of the items within a section, such as the *name* and *info* items, to make them more relevant to your device.

The text value within the *name* item is the field Label that is displayed within the form. The text value within the *info* item is the text that is displayed in the Status bar when you select the field in the form (i.e. gets focus).

Once you have made your amendments to the .ini file, remember to save it back to the \Devices sub folder.

You will then need to restart CatTools in order for your new custom device type to be read into the 'Device type' drop-down list field.

```
[info]
cookie=CatTools
version=3
author=SolarWinds enter your name or leave as the default

[device]
name=Custom.Manufacturer.Type change to a unique name. example: Custom.Cisco.Router
Note: do not use spaces. Use a 'period' mark (.)
id=4000 select a number within the range of 4000 to 4999. The number used here must be unique.

# Device info

[item_Group]
name=Group
default=Default enter a default Group for the device or leave as the default
required=1
info="The logical group that this device belongs to."

[item_Name]
name=Name
default=Unique device name enter a unique device name
required=1
info="A unique name for this device. e.g. sales-router or head-office-3500."
```

```
[item_HostAddress]
name=Host Address
default=127.0.0.1
required=1
info="IP address or host name of the device."
```

```
[item_Filename]
name=File Name
default=
required=1
info="The base file name to use for this device (unique)."
```

```
[item_Model]
name=Model
default=Custom enter a default Model to use from the list below or leave as the default
required=0
info="The device model number."
list=Custom add additional Model numbers separated by commas, for example:
Custom,501,515
```

```
[item_ConnectVia]
name=Connect via
default=Direct connect
required=1
info="The name of another device to connect to first."
```

```
[item_Telnet]
name=Method
default=Telnet enter a default Method to use from the list below (if your device only uses SSH,
you may want to change the default accordingly)
required=1
list=Telnet,SSH1,SSH2,SSH2-nopty,SSH1-DES,SSH1-3DES,SSH1-Blowfish,Cisco SSH amend the
list accordingly
info="Connection method to use."
```

```
[item_TelnetPort]
name=Port
default=23 enter a default port from the list below. Port field value automatically changes when
Method field is changed in the user interface.
required=1
list=23,22
info="Port number to use."
```

#### # Passwords

```
[item_VTYPass]
name=VTY Password
default=
required=0
info="VTY password."
```

```
[item_EnablePass]
name=Enable Password
default=
required=0
info="Enable or privilege password."
```

```
[item_PrivilegeLevel]
name=Privilege Level
default=
```



required=0  
info="Sets the enable mode privilege level. (Not required in most cases)"

[item\_ConsolePass]  
name=Console Password  
default=  
required=0  
info="The console (com port connection) password."

[item\_AAAUsername]  
name=Username  
default=  
required=0  
info="AAA/TACACS/RADIUS/Local username."

[item\_AAAPassword]  
name=Password  
default=  
required=0  
info="AAA/TACACS/RADIUS/Local password."

[item\_SSHPassword]  
name=SSH Password  
default=  
required=0  
info="SSH password."

[item\_SSHUsername]  
name=SSH Username  
default=  
required=0  
info="SSH username."

[item\_SNMPRead]  
name=SNMP Read  
default=  
required=0  
info="SNMP Read community name."

[item\_SNMPWrite]  
name=SNMP Write  
default=  
required=0  
info="SNMP Write community name."

[item\_RequireVTYLogin]  
name=Initial login requires password  
default=1  
required=0  
info="This device requires an initial password for access"

[item\_LoginUsesAAA]  
name=Initial login requires username/password  
default=0  
required=0  
info="The initial access requires a username and password"

[item\_EnableUsesAAA]  
name=Enable mode requires username/password  
default=0  
required=0

info="Enable mode access requires a username and password"

# Prompts

[item\_VTYPrompt]  
name=VTY Prompt  
info="Expected VTY prompt from the device. (Only required if non standard prompt is used)"

[item\_EnablePrompt]  
name=Enable Prompt  
info="Expected enable mode prompt from the device. (Only required if non standard prompt is used)"

[item\_ConsolePrompt]  
name=Console Prompt  
info="Expected console prompt from the device. (Only required if non standard prompt is used)"

[item\_AAAPassPrompt]  
name=Username prompt  
info="Expected Username prompt from the device or AAA server. (Only required if non standard prompt is used)"

[item\_AAAPassPrompt]  
name>Password prompt  
info="Expected AAA Password prompt from the device or AAA server. (Only required if non standard prompt is used)"

# Contact info

[item\_Address1]  
name=Address1  
default=  
required=0  
info="Location of the device"

[item\_Address2]  
name=Address2  
default=  
required=0  
info="Location of the device"

[item\_Address3]  
name=Address3  
default=  
required=0  
info="Location of the device"

[item\_ContactName]  
name=Contact Name  
default=  
required=0  
info="The name of the person responsible for this device."

[item\_ContactPhone]  
name=Contact Phone  
default=  
required=0  
info="How to contact the person responsible for this device"

[item\_ContactEmail]

```
name=Contact E-mail
default=
required=0
info="How to contact the person responsible for this device"

[item_ContactOther]
name=Contact Other
default=
required=0
info="Any additional contact info"

[item_AlertEmail]
name=Alert e-mail
default=
required=0
info="Who to notify by e-mail of any alarms or alerts for this device"

[item_SerialNumber]
name=Serial number
default=
required=0
info="The serial number of this device"

[item_AssetTag]
name=Asset Tag
default=
required=0
info="Asset tag information"

[item_Identification]
name=Identification
default=
required=0
info="Identification info for this device"

[item_SerialOther]
name=Other info
default=
required=0
info="Any other serial number information"

[item_ActivitySpecific1]
name=Activity Specific1
default=
required=0
info="Information specific to a particular activity"

[item_ActivitySpecific2]
name=Activity Specific2
default=
required=0
info="Information specific to a particular activity"
```

### 3.2.3 The custom device script file (.txt)

#### Device scripts in CatTools

Devices scripts are defined within CatTools using text files (**.txt** files) and are stored in the \Scripts sub folder within the root CatTools directory.

When an activity is run, CatTools reads the *name* item value in the **[device]** section from the device type [.ini](#) file to determine which device script .txt file it needs to use. Therefore each .txt file must be given the same file name as the *name* item in the corresponding [.ini](#) file. (Its also recommended that in order to save confusion, the .ini file and .txt files have the same file name apart from the file extension).

For all the predefined device types in CatTools, the associated device script .txt files have been encrypted. They are encrypted for two reasons. The first is to protect our intellectual property. The other is to prevent unauthorised modification of these files which may cause the scripts to fail at runtime.

### The custom device script .txt template

CatTools provides a starting point template file (based on a Cisco Router) to help assist in the creation of a new custom device script. The template file is called *Custom.Device.Template.txt* and can be found in the \Templates sub folder within the root CatTools directory.

This template file is included as part of the predefined device scripts in CatTools, but is unencrypted. As for all of the CatTools predefined device scripts, it may (as and when required) be subject to updates and modifications with each new release of the CatTools product.

For this reason, you should never modify the template file itself in order to create a new custom device script.

If you need to create a new custom device script, take a copy the template file and save it to the \Scripts sub folder giving it a new file name.

### Creating your custom device script .txt file

If adding a new custom device script to CatTools, the first step will therefore be to take a copy of the template file *Custom.Device.Template.txt* in the \Templates sub folder and save it to the \Scripts sub folder, giving it a file name the same as the value entered for the *name* item within the device type .ini file **[device]** section, and end with a .txt file type suffix.

This will give you a new starting point device script file to work with.

*For example:*

To create a new custom device script for a Cisco ASA Firewall device, you may have created an .ini file *Custom.Cisco.FirewallASA.ini* which contains the *name* item in the [device] section of *Custom.Cisco.FirewallASA*.

Therefore, the custom script file will need to be named *Custom.Cisco.FirewallASA.txt*.

### Editing the .txt file

The next step is to open the .txt file and make any necessary changes in order to get the script to work with your device.

The .txt file is well commented (documented) to provide instructions and assistance in making your modifications, however there are a few important sections at the top of the script file which require further mention.

#### General:

You may see the following line at the very top of the script:

```
Attribute VB_Name = "Dev_CustomDeviceTemplate"
```

This is the Visual Basic module name for the device script file when viewing within the Visual

Basic (or equivalent) development environment.

This line won't appear if editing the file in the VB development environment, but may do in others.

"Dev\_CustomDeviceTemplate" is the name given to the custom template file module.

If you have a number of custom device files, you may want to consider changing this name to reflect your device script file name. By doing so, you can then open multiple custom device script files (say within a project) in your VB development environment.

If you are using a text based editor to modify your scripts (such as Notepad.exe), then changing the module name is optional, although to avoid confusion you may prefer to change the name anyway.

### Option Explicit

The *Option Explicit* statement enforces a level of compliance for the scripting code. You must therefore explicitly declare all variables using the **Dim** statement, or if redimensioning, using **ReDim**. If you attempt to use an undeclared variable name, an error occurs at compile time. *Note:* as device script files use the Visual Basic Scripting language (VB Script), there is no declaration of the variable 'types'.

### Const declarations:

Below the Option Explicit statement is where all the script constants are declared.

```
Private Const SCRIPT_NAME = "Device Template"
```

The SCRIPT\_NAME constant is a reference given to the script file. This reference will appear within the Info Log pane and Infolog.txt file whenever the script is called.

```
Private Const DEVICE_
```

The DEVICE\_ constants are constants which define device configuration or responses from the device, for example: DEVICE\_USERNAMEPROMPT = "Username:" implies that every time we encounter a prompt from the device to enter a username, we are expecting to see a "Username:" prompt.

```
Private Const COMMAND_
```

The COMMAND\_ constants are constants which define commands that CatTools sends to the device in order to perform specific tasks or actions when running an activity. For example: COMMAND\_ENTERENABLEMODE = "enable" tells CatTools to send the 'enable' command to instruct the device to enter 'enable mode'.

### Notes:

```
--- SCRIPT NOTES ---
```

The SCRIPT NOTES section provides useful information and lists tasks that are required to be carried out during the initial script creation phase. You can edit this section as and when you feel it is appropriate. An example may be deleting the initial setup tasks block of text from the SCRIPT NOTES section after you have carried out the initial script setup tasks, as it no longer applies.

Any complex routines or script specific behaviour which may be of assistance to any person maintaining the device scripts, should be mentioned in the SCRIPT NOTES section. They should be either fully documented in the SCRIPT NOTES section, or a reference made to the relevant function where they are fully documented.

### --- DEVICE NOTES ---

The DEVICE NOTES section is an area where you can enter any device specific behaviour or quirks which may be of assistance to any person maintaining the device scripts. An example may be: *"The device you are creating the custom script for only supports SSH2"*.

#### Required functions:

There are a number of functions that must exist within your device script in order for CatTools access the device and run activities. These are:

Logi n	Authentication to the device
SendPost Logi nCommands	Commands to be sent after successful login to device
Ent er Enabl eMode	To enter privileged mode
SendPost Ent er Enabl eModeC ommands	Commands to set the environment after successfully entering enable mode
SendSessi onTer mi nat i onCo mmands	To send necessary commands to exit or logout of the device

The SendPost ... functions do not necessarily have to contain any commands, however the function must exist and return TRUE.

Likewise, if your device does not have the concept of an Enable mode, the device script will still require the Ent er Enabl eMode function, however the function only need include a single line of code sending the return value of TRUE.

#### Unsupported activities:

The device template file contains a number of activities functions which have a call to the internal CatTools client function [cl.CatToolsNoSupport](#).

You should call `cl . Cat Tool sNoSupport` so that a message will be written to the Info Log to inform you that the activity is not supported.

Most activities only require the one line of code call to the `cl.CatToolsNoSupport` function.

```

Funct i on Act i v i t y _ Updat ePasswor d ( )
    Cal l  cl . Cat Tool sNoSupport
End  Funct i on

```

Some also require a second line which supplies a return value to the CatTools main activity script.

```

Funct i on Act i v i t y _ TFTPUplo ad ( i Act i onType , sTFTPSever , sTFTPFi le ,
sResponseFi le , bDoSaveToNVRAM )
    Cal l  cl . Cat Tool sNoSupport
    Act i v i t y _ TFTPUplo ad = "Not  supported"
End  Funct i on

```

If you decide to add support in your custom device script for any of these activities, you need to ensure that you replace the call to the `cl . Cat Tool sNoSupport` function with your new lines of code and also update the return value appropriately if it exists.

#### **Saving the script file**

Before CatTools can access the script file you have built, you must save it to the \Scripts folder. Normally you do not have to stop and restart CatTools in order for script file code changes to

be picked up, however there may be occasions during testing that you find you may have to do this if the changes are not being recognised.

### Testing your custom device scripts

Once you have the scripts set up you can test the new device with CatTools by setting up activities using the new device.

See [Testing your custom device](#) for more information and tips.

### Send us your working scripts

Once you have successfully created support for your custom device in CatTools, if you would like us to consider adding it as a predefined device type to ship with the product, then please contact us using the [Technical Support](#) form on our [web site](#).

There are no guarantees as to when or if the device will be added to the predefined device types in CatTools, as there are a number of factors to consider: e.g. the number of requests for the device, complexity of the script, technical resources available, etc; however all scripts that are sent in will be cataloged for future reference.

## 3.2.4 Custom device script cl. variables and functions

The [custom device template script](#) contains many references to the internal CatTools client variables and calls to client functions.

These variables and functions can be identified by their 'cl.' prefix.

Although the functions and variables are not documented within the template script itself (mainly to cut down the size of the script file), in order for you to understand what each one is used for and in the case of the functions understand the parameters they require and what their return values are, each variable and function exposed in the custom device template file is detailed below.

### VARIABLES

*Device configuration variables (read from the device setup screen tabs)*

#### Device Info tab

cl.CurDevName	Unique name assigned by user to refer to device ( <i>Name</i> field)
cl.CurDevTelnet	The device connection method, i.e. Telnet or SSH ( <i>Method</i> field)
cl.CurDevGroup	The logical group that this device belongs to ( <i>Group</i> field)
cl.CurDevHostAddress	IP address or host name of the device ( <i>Host Address</i> field)
cl.CurDevModel	The device model /series number. ( <i>Model</i> field)
cl.CurDevType	The CatTools device type. ( <i>Device Type</i> field)
cl.CurDevFilename	The base file name to use for this device (unique) ( <i>File Name</i> field)
cl.CurDevConnectVia	The name of another device to connect to first ( <i>Connect via</i> field)
cl.CurDevTelnetPort	Port number to use ( <i>Port</i> field)

#### Passwords tab

cl.CurDevVTYPass	Device VTY password ( <i>VTY password</i> field)
cl.CurDevEnablePass	Device Enable password ( <i>Enable password</i> field)
cl.CurDevPrivilegeLevel	Device Enable mode privilege level ( <i>Privilege level</i> field)
cl.CurDevAAAUsername	Device Username ( <i>Username</i> field)

cl.CurDevAAAPassword	Device Password ( <i>Password</i> field)
cl.CurDevRequireVTYLogin	Device requires Password only to log in ( <i>Initial login requires password</i> tick box field)
cl.CurDevLoginUsesAAA	Device requires Username and Password to log in ( <i>Initial login requires username/password</i> tick box field)
cl.CurDevEnableUsesAAA	Device requires Username and Password to enter Enable mode ( <i>Enable mode requires username/password</i> tick box field)
cl.CurDevConsolePass	Device console (com port connection) password ( <i>Console password</i> field)
cl.CurDevSSHUsername	SSH username ( <i>SSH Username</i> field)
cl.CurDevSSHPassword	SSH Password ( <i>SSH Password</i> field)
cl.CurDevSNMPRead	SNMP Read community name ( <i>SNMP Read</i> field)
cl.CurDevSNMPWrite	SNMP Write community name ( <i>SNMP Write</i> field)

**Prompts tab**

cl.CurDevVTYPrompt	Device VTY custom prompt assigned by user to device ( <i>VTY Prompt</i> field)
cl.CurDevEnablePrompt	Device Enable mode custom prompt assigned by user to device ( <i>Enable Prompt</i> field)
cl.CurDevConsolePrompt	Device Console port custom prompt assigned by user to device ( <i>Console Prompt</i> field)
cl.CurDevAAAUserPrompt	Device AAA Username custom prompt assigned by user to device ( <i>Username Prompt</i> field)
cl.CurDevAAAPassPrompt	Device AAA Password custom prompt assigned by user to device ( <i>Password Prompt</i> field)

**Contact Info tab**

cl.CurDevAddress1	Location of the device ( <i>Address1</i> field)
cl.CurDevAddress2	Location of the device ( <i>Address2</i> field)
cl.CurDevAddress3	Location of the device ( <i>Address3</i> field)
cl.CurDevContactName	The name of the person responsible for this device ( <i>Contact Name</i> field)
cl.CurDevContactPhone	How to contact the person responsible for this device ( <i>Contact Phone</i> field)
cl.CurDevContactEmail	How to contact the person responsible for this device ( <i>Contact Email</i> field)
cl.CurDevContactOther	Any additional contact info ( <i>Contact Other</i> field)
cl.CurDevAlertEmail	Who to notify by e-mail of any alarms or alerts for this device ( <i>Alert e-mail</i> field)

**Extra Info tab**

cl.CurDevSerialNumber	The serial number of this device ( <i>Serial Number</i> field)
cl.CurDevAssetTag	Asset tag information ( <i>Asset Tag</i> field)
cl.CurDevIdentification	Identification info for this device ( <i>Identification</i> field)
cl.CurDevSerialOther	Any other serial number information ( <i>Other info</i> field)
cl.CurDevActivitySpecific1	Information specific to a particular activity ( <i>Activity Specific 1</i> field)
cl.CurDevActivitySpecific2	Information specific to a particular activity ( <i>Activity Specific 2</i> field)

*Other variables*

cl.RxBuffer	String of response data sent from the device
cl.ScheduleNumber	The current schedule number
cl.DeviceHostnameID	Device host name as recovered from the device after successful login
cl.DeviceVTYPrompt	The host name and ending with DEVICE STANDARDPROMPT
cl.DeviceEnablePrompt	The host name and ending with DEVICE PRIVILEGEDPROMPT
cl.DeviceConfigPrompt	The host name and ending with DEVICE CONFIGPROMPT

**FUNCTIONS - Summary list**



### General & file

cl.Initialise	Initialises (or defaults) cl. variables
cl.Delay	Force CatTools to pause for a given amount of time
cl. GetUniqueDeviceFileName	Generates unique filename in the \ClientTemp folder based on the device
cl.DBMetaCmd	Checks if a command is a database meta command, and then processes it
cl.UtilityMetaCmd	Checks if a command is a utility meta command, and then processes it
cl.Log	Sends a line of text to the Infolog.txt file and Info Log pane
cl.LogToFile	Writes data to a file

### Activity

cl.CatToolsNoSupport	Called for activities not supported by the device script
cl.DBCheckScheduleOption	Determine whether a particular option has been selected within a given activity

### Device

cl.FlushRxBuffer	Clears cl.RxBuffer string. Used to clear previous response data before receiving next data
cl.DetermineHostname	Establishes and sets the device hostname and prompts cl. variables
cl.SendData	Sends text to device
cl.SendAndWaitForEcho	Sends text to device and waits for echo
cl.SendAndWaitForPrompt	Sends text to device, waits for an echo and then device prompt
cl.WaitForData	Waits for a specific data string to returned from the device
cl.WaitForMultData	Waits for any one of a range of specific data strings to be returned from the device

### Text manipulation

cl.ReplaceText	Replace a substring within a given string of data, with a new substring
cl. TextRemoveBlankHeaderLines	Remove blank header lines from the beginning of a string (e.g. a device output buffer)
cl. TextRemoveLinesContainingText	Remove lines which contain a specific substring of text
cl.TextInText	Return start position of a substring within a string
cl.TextRemoveTextUpTo	Trim a text from a string up to or including a specified substring

### FUNCTIONS - Detail and examples

The functions listed in the above table are further detailed below with their input parameters, etc. and (in some cases) examples of their usage are provided.

#### **cl.Initialise()**

This function initialises (or defaults) [cl.variables](#) as follows:

```

cl.DeviceHostnameID = ""
cl.DeviceVTYPrompt = ""
cl.DeviceEnablePrompt = ""
cl.DeviceConfigPrompt = ""

```

```

cl.WaitForEcho = True
cl.WaitForTime = 0

```

### **cl.Delay(IDuration)**

This function forces CatTools to pause for a given amount of time.

It has one input parameter:

<i>IDuration</i>	Amount of time to wait for (in seconds)
------------------	---

### **cl.GetUniqueDeviceFileName(sFolderName, sPrefixName) As String**

This function generates a unique filename in the \ClientTemp folder based on the device. The function has a return value of String being the path and a unique filename.

It has two input parameters passed by value, and two optional parameters:

<i>sFolderName</i>	String value to append to beginning of filename. Can accept an empty string ""
<i>sPrefixName</i>	String value to append after <i>sFolderName</i> within the filename

Example: write the output to a temporary file in the \ClientTemp folder

```

' create the unique filename using only the prefix 'CLI'
sClientResultFile = cl.GetUniqueDeviceFileName( "", "CLI" )

' write the data in the RxBuffer to a file in \ClientTemp folder using the filename just
created.
cl.LogToFile sClientResultFile, cl.RxBuffer

```

### **cl.DBMetaCmd(sCmd) As Long**

This function checks a command to see if it is a database meta command. If it is, then CatTools tries to process it. A [database meta command](#) enables you to directly update a field in a table in the CatTools database as part of an Activity. They are not run on a device. The syntax for a database meta command is [%ctDB:tablename:fieldname:value](#)

The function has one input parameter:

<i>sCmd</i>	String value of the command being evaluated
-------------	---

Function return values:

0	Command is not a database meta command
1	Command evaluates as a database meta command and is processed successfully (a debug message also sent to the Info Log)
-1	Command evaluates as a database meta command, but fails to be processed (an error message also sent to the Info Log)

**Example:** check whether command to send is a database meta command. If it isn't (i.e. it is a device command) then send it to device.

```

I Ret Val = cl . DBMetaCmd( sCmd)
If I Ret Val = 0 Then 'not a database meta command so send it
    cl . FlushRxBuffer
    ' send command to device and wait for echo
    bRet Val = cl . SendAndWaitForEcho( sCmd)
    If bRet Val then
        ' command has been echoed, so lets execute it
        cl . SendData vbCr
    End if
End If

```

### **cl.UtilityMetaCmd(sCmd) As Long**

This function checks a command to see if it is a utility meta command. If it is, then CatTools tries to process it. A [utility meta command](#) enables you to set various options for use in the Device.CLI.Send commands and Device.CLI.Modify Config activities. They are not run on a device. Utility meta commands change the default behaviour of some of the internal CatTools functions.

The function has one input parameter:

*sCmd*                      String value of the command being evaluated

Function return values:

0	command if not a utility meta command
1	command evaluates as a utility meta command and is processed successfully (a debug message also sent to the Info Log)
-1	command evaluates as a utility meta command, but fails to be processed (an error message also sent to the Info Log)

**Example:** check whether command to send is a utility meta command. If it isn't then send it to device.

```

I Ret Val = cl . UtilityMetaCmd( sCmd)
If I Ret Val = 0 Then 'not a utility meta command so send it
    cl . FlushRxBuffer
    ' send command to device and wait for echo
    bRet Val = cl . SendAndWaitForEcho( sCmd)
    If bRet Val then
        ' command has been echoed, so lets execute it
        cl . SendData vbCr
    End if
End If

```

### **cl.Log(iPriority, sMessage)**

This function sends a line of text to the Infolog.txt file and [Info Log pane](#).

It has two input parameters:

<i>iPriority</i>	Integer range 1 to 4 representing the logging 'level' for the line being sent. 1=Error, 2=Warning, 3=Information, 4=Debug
<i>sMessage</i>	Message text of the line being sent

**Example:** log a level 4 (debug) message sending the text "Login Device Template: Custom

*Device 1*" (assume Const SCRIPT\_NAME from Device Template script and that you have named your device 'Custom Device 1' in the Name field of the Device Info tab of the device setup screen.

```
cl.Log 4, "Log in " & SCRIPT_NAME & ": " & cl.CurDevName
```

### **cl.LogToFile(sFilename, sData, bAppend)**

This function writes data to a file.

It has three input parameters:

<i>sFilename</i>	String of the name and path of the file to write to
<i>sData</i>	String of data to write
<i>bAppend</i>	Boolean value (default or if unspecified is False). If True, then the file will be appended to otherwise it is overwritten

Example: save current device response to temp.txt file on C:\ drive

```
cl.LogToFile "c:\temp.txt", cl.RxBuffer, 1
```

### **cl.CatToolsNoSupport()**

This function is called for those activities not currently supported by the device script.

### **cl.DBCheckScheduleOption(IScheduleNumber, IOptionNumber) As Long**

This function is used to determine whether particular options have been selected within a given activity.

It has two input parameters:

<i>IScheduleNumber</i>	The current schedule number as a Long
<i>IOptionNumber</i>	The option item within the activity we are checking as a Long

Example: check an activity (in this instance a [Device.CLI.Send commands](#) activity) to see if the output of the command(s) should be emailed

```

I Ret Val = cl.DBCheckScheduleOption(cl.ScheduleNumber, 8)
If I Ret Val = 1 then
    'code to email commands goes here...
End if

```

### **cl.FlushRxBuffer()**

This function clears the cl.RxBuffer string. Normally used to clear out any previous device response data before receiving the next chunk of response data.

### **cl.DetermineHostname(Optional ByVal vStandardPrompt As Variant, Optional ByVal vPrivilegedPrompt As Variant, Optional ByVal vConfigPrompt As Variant) As Boolean**

The purpose of this function is to establish and set the [cl.variables](#) listed below and return a value of **True** if successful.

cl.DeviceHostnameID The host name of the device (e.g. DevHost123), used as 'seed' for

following prompts:

cl.DeviceVTYPrompt	The host name and ending with DEVICE_STANDARDPROMPT	example: DevHost123>
cl.DeviceEnablePrompt	The host name and ending with DEVICE_PRIVILEGEDPROMPT	example: DevHost123#
cl.DeviceConfigPrompt	The host name and ending with DEVICE_CONFIGPROMPT	example: DevHost123(

It has three optional input parameters passed by value:

<i>vStandardPrompt</i>	Device standard mode prompt (or VTY prompt). If none specified then ">" is used as the default
<i>vPrivilegedPrompt</i>	Device privileged mode prompt (or Enable prompt). If none specified then "#" is used as the default
<i>vConfigPrompt</i>	Device configuration mode prompt (or VTY prompt). If none specified then "(" is used as the default

### **cl.SendData(sDataToSend)**

This function sends the specified text to the device.

It has one input parameter:

*sDataToSend*      string of the data to be sent to the device

Example: send a carriage return to the device

```
cl . SendData vbCr
```

### **cl.SendAndWaitForEcho(sDataToSend) As Boolean**

This function sends a text string to the device and waits for an echo. By waiting for an echo of the text string it ensure that the string that we are expecting to send is actually the string that is sent (just in case we send a string and the device doesn't echo back with it in its entirety, or it gets substituted/corrupted.)

The function has a Boolean return value; **True** if string is echoed back as expected, **False** if not.

It has one input parameter:

*sDataToSend*      String of the data being sent to the device and expected to be echoed back

This function is sometimes followed by a 'cl.SendData vbCr' which then executes the echoed string on the device.

Example: send command to show the device running configuration and wait for an echo. If echoed, then execute the command

```
bRetVal = cl . SendAndWaitForEcho("show running-config")
If bRetVal Then
    cl . SendData vbCr
End If
```

### **cl.SendAndWaitForPrompt(sDataToSend) As Boolean**

This function sends a text string to the device and waits for an echo. If successfully echoed, the string is executed and the function waits for a valid device prompt to be returned (standard or

privileged prompts).

This function is normally called when executing a known valid command on a device with no output, for example: 'term len 0' to turn off output paging.

The function has a Boolean return value; **True** if string is echoed back and you then receive one of the expected prompts after the command has been executed, **False** if not.

It has one input parameter:

*sDataToSend*      String of the data being sent, echoed and then executed on device

#### **cl.WaitForData(sData, ITimeout) As Boolean**

This function waits for a given amount of time for the specific string data to be returned from the device.

The function has a Boolean return value; **True** if string is found within the timeout period specified, **False** if not.

It has two input parameters:

*sData*              The string of data we are waiting to receive from the device  
*ITimeout*          The amount of time to wait for (in seconds)

Example: send the command to exit config mode and then check if we have been returned to the 'Enable' mode prompt within 30 seconds timeout

```
cl.SendData Chr(26) ' i.e. CTRL-Z
If cl.WaitForData('Host Name1#', 30) = False Then
    cl.Log 4, "Failed to exit Configure Terminal mode"
End if
```

#### **cl.WaitForMultData(rgMult, Optional iChoices = 0, Optional ITimeout = 0) As Long**

This function waits for a given amount of time for any one of string data items defined in the range, to be returned from the device within the specified amount of time.

The function has a return value of Long representing which item of string data it has found first. It returns 0 if none of the items are found within the given timeout value.

It has three input parameters:

*rgMult*            The range of possible string data items we are expecting back from the device  
*iChoices*        (*Optional*). The count of the number of items we are looking for - excludes *rgMult(0)* item. Optional as this is now calculated by the function itself if not specified  
*ITimeout*        (*Optional*). The amount of time to wait for (in seconds). Optional parameter, if not specified then default CatTools internal timeout value is applied

Example: wait for one of the device prompts to be returned within 30 seconds

```
rgMult(1) = "Host Name1>"
rgMult(2) = "Host Name1#"
rgMult(3) = "Host Name1( Config )"
iChoices = 3
ITimeout = 30

iRetVal = cl.WaitForMultData(rgMult, iChoices, ITimeout)
If iRetVal = 0 Then
    cl.Log 4, "Failed to receive device prompt"
End if
```

You could also just send:

```
cl.WaitForMultData(rgMult)
```

which uses the default timeout within CatTools, or:

```
cl.WaitForMultData(rgMult, , lTimeOut)
```

which uses your specified timeout value. The function works out the *iChoices* value.

If you are using the constant `COMMAND_TIMEOUT` to specify your timeout, then you can increase the timeout using:

```
Const COMMAND_TIMEOUT = 30 '(i.e. in seconds)

iTimeOutMultiplier = 2 '(increase by a factor of 2)
cl.WaitForMultData(rgMult, , COMMAND_TIMEOUT * iTimeOutMultiplier)
```

### **cl.ReplaceText(sData, sFind, sReplace) As String**

This function replaces a substring within a given string of data, with a new substring. It replaces all substring occurrences with the new substring. The function has a return value of String being the new string of data with replacements. If no replacements are made, then the return string is the same as the original string. The function returns a *null* string if the length of the input parameter *sData* is 0.

It has three input parameters:

<i>sData</i>	String to be searched
<i>sFind</i>	The substring we want to replace
<i>sReplace</i>	The new substring we want to replace with

Example 1: replace all occurrences of the substring "old text" with "new data" within the string object *sData*

```
sData = cl.ReplaceText(sData, "old text", "new text")
```

Example 2: remove *nulls* from within the string object *sData*

```
sData = cl.ReplaceText(sData, Chr(0), "")
```

### **cl.TextRemoveBlankHeaderLines(ByVal sData) As String**

This function is used to remove blank header lines from the beginning of a string (e.g. a device output buffer)

It is used primarily for massaging the configuration data output from a device to create the backup file.

It works through from the top of the output buffer string, removing lines containing just a carriage return <CR> or a line-feed <LF> (or both).

The function has a return value of String being the 'trimmed' buffer string.

It has one input parameter passed by value:

<i>sData</i>	String we are manipulating
--------------	----------------------------

Example: trim all the blank lines from the top of the buffer string object *sConfigData*

```
sConfigData = cl.TextRemoveBlankHeaderLines(sConfigData)
```

### **cl.TextRemoveLinesContainingText(ByVal sData, ByVal sFind) As String**

This function is used to remove lines from within the device output buffer which contain a specific substring of text.

It is used primarily for massaging the output data from a device to create a text file or report. The function has a return value of String being the new buffer with the lines containing the substring removed.

It has two input parameters passed by value:

<i>sData</i>	String to be searched
<i>sFind</i>	Substring we are searching for

**Example:** remove all config lines of data from the buffer string object *sConfigData*, containing the device paging prompt text i.e. --More--

```
Const DEVICE_MORETEXT = "--More--"

sConfigData = cl.TextRemoveLinesContainingText(sConfigData,
DEVICE_MORETEXT)
```

### **cl.TextInText(ByVal sData, ByVal sFind) As Integer**

This function returns the start position of a substring within a string.

If nothing is found, or the length of either input strings are 0, then the function returns 0.

It has two input parameter passed by value:

<i>sData</i>	String to be searched
<i>sFind</i>	Substring we are searching for

**Example:** find the start position of the device header text. If value is 0, we haven't found it so send a line to the Info Log

```
Const DEVICE_CONFIGHEADERTEXT = "Generating configuration:"

iRetVal = cl.TextInText(sConfigData, DEVICE_CONFIGHEADERTEXT)
If iRetVal = 0 then
    cl.Log 4, "Failed to find device configuration header text"
End if
```

### **cl.TextRemoveTextUpTo(ByVal sData, ByVal sFind, Optional bAndIncluding As Boolean = False, Optional bForwards As Boolean = True) As String**

This function is used to trim a text from a string up to or including a specified substring. The trim can work in either direction, i.e. from the start of a file to the end, or from the end to the start.

The function has a return value of String, being the new string with the relevant text removed, or if the substring *sFind* cannot be found then the original string *sData* is returned.

It has two input parameters passed by value, and two optional parameters:

<i>sData</i>	String to be searched
<i>sFind</i>	Substring we are searching for
<i>bAndIncluding</i>	(Optional). Boolean value; if set to <b>True</b> will trim up to 'and including' the



*bForwards* substring text *sFind*. If **False** (default) then the substring *sFind* will be not be trimmed (*Optional*). Boolean value; if set to **True** (default) will trim from the start (or top) of the string we are searching. If **False**, then the function trims from the end (or bottom) of the string backwards

Example: check output for header text and if found remove everything up to and including it

```
Const DEVI CE_CONFI GHEADERTEXT = "Generating configuration:"

If cl.TextInText(sConfigData, DEVI CE_CONFI GHEADERTEXT) > 0 Then
    sConfigData = cl.TextRemoveTextUpTo(sConfigData,
        DEVI CE_CONFI GHEADERTEXT, True, True)
End If
```

### 3.2.5 Testing your custom device

Testing your custom device scripts is relatively straight forward.

Once you have the device type [.ini](#) file and the device script [.txt](#) file set up, you can test the new device with CatTools by setting up activities using the new device type.

The most basic activity to start testing with is the [Device.ConnectivityTest.Login](#) activity to ensure you can access the device.

Nearly all of the activities that run on a device require logging in to the device, therefore if the [Device.ConnectivityTest.Login](#) activity fails, it is likely the other activities will also fail.

There are several testing aids available in CatTools.

#### 1) Info Log messages

Within the Info Log pane you see messages appear while an activity is running. The level of messages that appear can be filtered by the drop-down list near the bottom of the Info Log window. By selecting level "**4) Show Debug events and above**" you get to see all the [cl.log](#) messages being displayed during the running of an activity.

You should be aware that all the Info Log messages are logged to a file called InfoLog.txt in the root CatTools folder. This file can get very large very quickly, so can be purged by selecting the [File | Delete | Delete InfoLog.txt file](#) menu item from the CatTools File menu.

The client script function [cl.Log](#) 4,"message" is found throughout device scripts to display level 4 messages which help assist in troubleshooting device specific issues.

#### 2) Debug log

Under the File menu of CatTools there is an entry [Enable Capture Mode](#). This turns on the logging of the conversation between CatTools and the end device and creates a disk file in the \Debug sub folder of the communications. This can be extremely useful when the script does not appear to be communicating correctly with the device after entering a command. The scripting mechanism waits for known prompts when issuing a command to the device, but eventually times out if an expected prompt does not appear. The debug log capture mode file shows what the device actually sent to CatTools, and you can adjust your code accordingly.

## 3.3 Device specific information

This sub-chapter provides further information on various different devices that helps explain their behaviour or helps in setting the device up in CatTools.

### 3.3.1 Device Variations

#### **What are variations?**

There are a number of device specific script variables that CatTools uses during the execution of device scripts. If the characteristics of your device do not match these device specific variables then the script will not work correctly. The variations system provides a way to override these device specific variables on a device by device basis.

#### **Which devices support variations?**

The Generic.Device script is a basic script designed to be a starting point for the use of variations. However, some of the other scripts also support the use of variations. In the device information panel a variations tab will be available if the device supports variations. You can access the [variations GUI](#) via the **Variations** tab (shown below).

**Device Information**

Device info | Passwords | Prompts | Contact info | Extra info | Variations

Vendor: [All Vendors]   
 Filter by vendor or show 'All Vendors'.

Device Type: Cisco.Firewall.ASA

Group:\* Default   
 Logically group your devices, if desired.

Name:\* ASA Firewall 2   
 A unique name for this device.

Host Address:\* 127.0.0.1

File Name:\* ASA\_Firewall\_2   
 Base file name for this device.(unique)

Model: ASA5505   
 Select model from list or type your own.

Connect via:\* Direct connect   
 Name of another device to connect to first, if necessary.

Method:\* Telnet

Port:\* 23

Ping device | Telnet/SSH | OK | Cancel

### **Overview of how to use variations**

To use variations on a specific device a variations file for that device must be placed in the \Variations subfolder of the CatTools install directory. The variations file must be named the same as the 'Device File Name' in the CatTools Device Information|Device info pane. The file must have a .txt extension.

So for example if your device was 'Generic Device 1' with a File Name of 'Generic\_Device\_1' your variations file would be called 'Generic\_Device\_1.txt'

You can manually create this file using notepad or you can use the [variations GUI](#).

An individual variations file will take precedence over a group variations file (see below).

### **Applying variations to a group of devices**

It is possible to apply a single variations file to all devices of the same script type that belong

to the same group. To do this you should name the variations file the same as the device script type . eg. Generic.Device.txt. It should be placed in a folder with the same name as the group the devices belong to. This folder should be in the Variations\Groups folder. So for example to apply a variations file to all devices of Juniper.Router device type that belong to a group called HeadOffice your directory structure would be; ...CatTools\Variations\Groups\HeadOffice\Juniper.Router.txt

You can manually create this file using notepad or you can use the [variations GUI](#).

### **Variations File Contents**

The variations file should include only the items you need to override.

Outlined below are the items that can be overridden. The value within the "" is the value that you should change to your device specific value.

**Login Prompts:-** These are the prompts that the device will display when it asks for username or password.

```
DEVICE_USERNAMEPROMPT = "Username:"  
DEVICE_PASSWORDPROMPT = "Password:"
```

**Mode Prompts:-** These are the characters which appear at the end of the hostname in the various modes the device has. So for example, if at your VTY prompt 'Cisco2950>' was displayed the prompt would be ">".

```
DEVICE_STANDARDPROMPT = ">"  
DEVICE_PRIVILEGEDPROMPT = "#"  
DEVICE_CONFIGPROMPT = "("
```

Note that in this case, for the config prompt we have selected only the first symbol that appears to the right of the hostname in config mode. So this would match the 'extended' config modes also. eg. It would match Cisco2950(config) and Cisco2950(Config-if) because 'Cisco2950(' appears in both of these prompts.

**More Text (Paging prompts):-** This is the text that is displayed at the end of a page when device paging is turned on. Note: you must include all of the 'more' paging prompt text. The Device.Backup.Running Config activity strips the more text from the saved config. If you have not set the full 'more' paging prompt text, then the saved config may contain additional undesirable/invalid lines of config.

```
DEVICE_MORETEXT = "--More--"
```

**Invalid / Incomplete command :-** These are the messages output by the device when an invalid or incomplete command are entered. Note: you do not have to include all of the 'error' text but enough so that it can be uniquely recognised and distinguished from other device output.

```
DEVICE_INVALIDCOMMAND = "% Invalid input detected at"  
DEVICE_INCOMPLETECOMMAND = "% Incomplete command"
```

**Yes/No Text:-** This is the text that appears when the device asks you a question which requires a yes or no answer. For example: "Reload Config (y/n)". Note that you should not include all of the text but just enough so that it can be uniquely recognised and distinguished from other device output.

```
DEVICE_YESNOTEXT = "(y/n)"
```

**More Text Keystroke:-** This is the key that is pressed to continue from a paging prompt.  
 MORETEXT\_KEYSTROKE=" "

**Device Commands:-** These are commands that are issued to the device during execution of certain parts of the script.

The commands to enter and leave enable / privileged exec (enable) mode.

```
COMMAND_ENTERENABLEMODE = "enable"
COMMAND_EXITENABLEMODE = "disable"
```

The commands to enter / leave config mode.

```
COMMAND_ENTERCONFIG = "configure terminal"
COMMAND_EXITCONFIG = "CTRL-Z"
```

The commands to disable / enable paging.

```
COMMAND_DISABLEPAGING = "terminal len 0"
COMMAND_ENABLEPAGING = "terminal len 24"
```

The commands to show the running config and the start up config. Note it is best not to use abbreviated commands such as "sh run" as some of the scripts look for parts of the command, such as 'show' and use this to determine whether to increase the timeout for that command.

```
COMMAND_RUNNINGCONFIG = "show running-config"
COMMAND_STARTUPCONFIG = "show startup-config"
```

The commands to save changes, disconnect from the device and ping another device.

```
COMMAND_SAVENVRAM = "write mem"
COMMAND_DISCONNECT = "exit"
COMMAND_PING = "ping"
```

**Backup Ignore Text:-** The Device.Backup.Running Config activity defines certain items that will be ignored as changes when the current and previous configs are compared. This variation allows you to add additional device specific items to the ignore text. Note that unlike the variations above you can have multiple 'Backup Ignore Text' items in your variations file. You cannot '|' delimit multiple items in a single line. Each ignored item should be on a new line. See [Ignore Text](#) for more information.

```
BACKUP_IGNORETEXT = "^! Last configuration change at"
BACKUP_IGNORETEXT = "<Time"
```

**Pre-Login Keystroke:-** Some devices require keystrokes to be sent prior to login to 'wake up' the device. You can specify more than one 'Pre Login Keystroke' item so if, for example, you needed to press space twice prior to login you would include two 'Pre login keystrokes' lines, each sending a single space.

Examples:

```
PRE_LOGIN_KEYSTROKE = "CTRL-Y"
```

```
PRE_LOGIN_KEYSTROKE = "CTRL-Z"  
PRE_LOGIN_KEYSTROKE = "CR"  
PRE_LOGIN_KEYSTROKE = " "  
PRE_LOGIN_KEYSTROKE = "Y"
```

**Pre-Login Message:-** Some devices that require pre-login keystrokes will prompt with a message. If this is the case then the pre-login message should be set as this will ensure the keystrokes are not sent until the message appears.

```
PRE_LOGIN_MESSAGE = "Press CTRL-Y to continue"
```

**Post-Login Keystroke:-** Some devices require keystrokes to be sent after to login. For example at a menu to get to the CLI prompt. You can specify more than one 'Post Login Keystroke' item so if, for example, you needed to press space twice after login you would include two 'Post login keystrokes' lines, each sending a single space.

Examples:

```
POST_LOGIN_KEYSTROKE = "CTRL-Y"  
POST_LOGIN_KEYSTROKE = "CTRL-Z"  
POST_LOGIN_KEYSTROKE = "CR"  
POST_LOGIN_KEYSTROKE = " "  
POST_LOGIN_KEYSTROKE = "Y"
```

**Post-Login Message:-** Some devices that require post-login keystrokes will prompt with a message. If this is the case then the post-login message should be set as this will ensure the keystrokes are not sent until the message appears. For example this might be part of the text that appears in a menu prior to keys strokes being sent to enter the CLI.

```
POST_LOGIN_MESSAGE = "Press C to enter the CLI"
```

**Full Prompt Values:-** CatTools will normally determine the device hostname and from this and the specified prompts determine the full VTY, enable and config prompt. However should CatTools not be able to do this for your device you can specify the full values. If any of the below four values are used as an override CatTools will not try to determine the hostname for the device. Therefore generally speaking if you intend to use any of the below you should specify values for all four items.

```
FULL_HOSTNAMEID = "Cisco2950"  
FULL_VTYPROMPT = "Cisco2950>"  
FULL_ENABLEPROMPT = "Cisco2950#"  
FULL_CONFIGPROMPT = "Cisco2950("
```

**Data Timeout:-** When CatTools waits for a response from a device it will wait for a specified period of time (normally 30 seconds) and if no data is returned will timeout. For certain commands which are known to produce more output, such as 'show' commands, this default timeout will be increased using a multiplier (normally \*4). If you have a device which is particularly slow you can increase the default timeout from 30 seconds.

```
RESPONSE_TIMEOUT= "30"
```

**Strip Characters:-** Some devices include spurious escape \ANSI \Null characters in their output. Functionality to remove these from the buffer as the data is captured can be turned on if required.

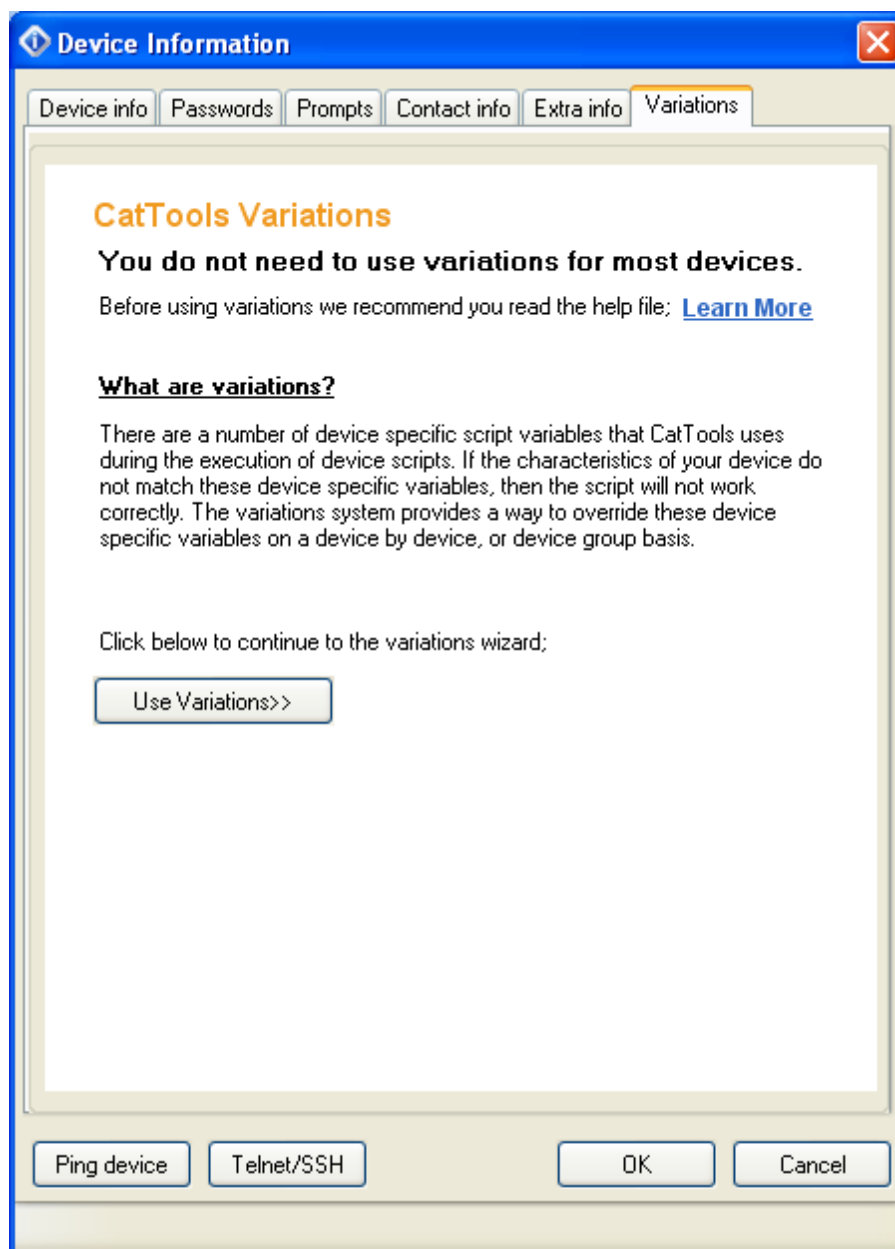
```
RESPONSE_STRIP_VT100ESC = "1"  
RESPONSE_STRIP_ANSICHARS="1"  
RESPONSE_STRIP_NULLS="1"
```

### 3.3.1.1 Variations GUI

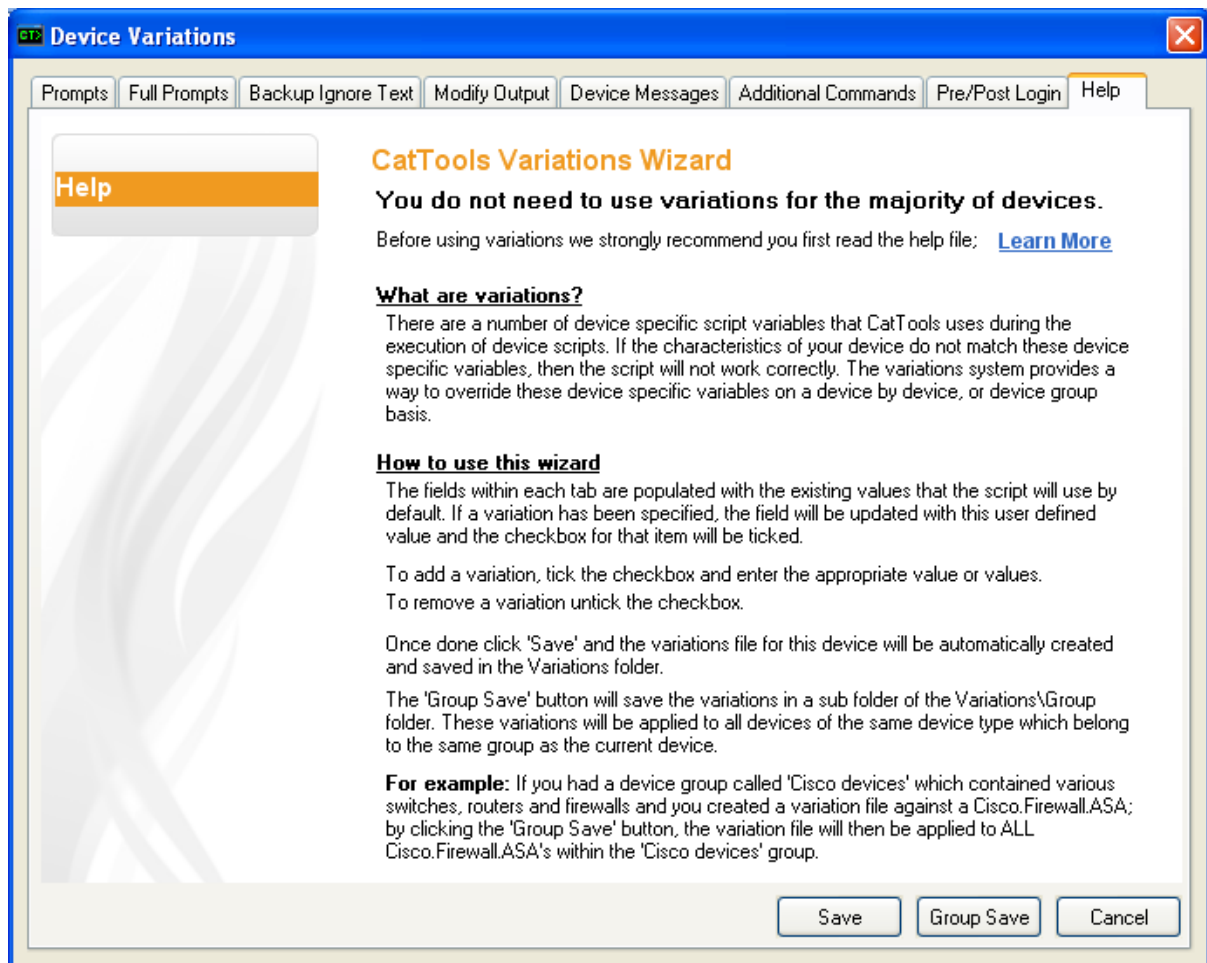
#### **The Variations GUI**

Although you can manually create a variations file in notepad.exe, the variations GUI will help automate this process.

If the device script supports variations then the **Variations** tab will be available on the **Device Information** panel.



To open the Variations GUI, click the '**Use Variations**' button on this tab. The Variations GUI will then be displayed, defaulting to the 'Help' tab.



Once opened, the tabs on the GUI will show any variations currently being used. If a field shows a grayed out value this is the value that the underlying script is currently using.

To select a variation tick the check-box and enter the variations value.



**Device Variations**

Prompts Full Prompts Backup Ignore Text Modify Output Device Messages **Additional Commands** Pre/Post Login Help

**Additional commands**

**Current field help**  
The command to show the device start-up configuration. Note: it is best not to use abbreviated commands such as "sh run" as some of the CatTools device scripts look for parts of the full command, such as 'show' and use this to determine whether to increase the default timeout for that command.

**Additional Commands**

<input type="checkbox"/> Enter Enable Mode	enable
<input type="checkbox"/> Exit Enable Mode	disable
<input type="checkbox"/> Enter Config Mode	configure terminal
<input type="checkbox"/> Exit Config Mode	CTRL-Z
<input type="checkbox"/> Disable Paging	
<input type="checkbox"/> Enable Paging	
<input type="checkbox"/> Show Running Config	
<input checked="" type="checkbox"/> Show Startup Config	show startup-config
<input type="checkbox"/> Save To NVRAM	write mem
<input type="checkbox"/> Disconnect	exit
<input type="checkbox"/> Issue Ping	ping

Save Group Save Cancel

**\*Note:** When focus moves to a text-box field, the the '**Current field help**' in the left pane is populated with additional helpful information about that field.

### **Saving a device variation**

Once you have selected and entered values for the fields you want to override click the **Save** button and the variations file will be automatically created for you in the variations folder for that device.

### **Applying a device variation to a group of devices**

If you want to apply the variations to all devices of the same device type as the current device, and that belong to the same group as the current device, then click the **Group Save** button instead.

### **Removing variations**

To remove all variations you can either delete the variations file from the variations folder\sub folder. Or you can remove all the variations in the variations GUI and then click either save or save group as appropriate.

See [Device Variations](#) for more information on the individual variations.

### 3.3.2 Dell devices

On some Dell devices (for example: PowerConnect switches 5224, 5324, 6024, etc. ), when capturing device output for configuration backups or reports, the device may be interrupted with system or port status messages.

Because these messages are sent to the terminal window, they will also be captured along with the rest of the data output and can therefore cause reporting issues or problems with the configuration backup activity compare function.

To prevent these messages from occurring, you can make a change to the device global configuration by sending the following command when in configuration mode:

**no logging console**

This prevents all system messages from being written to the console/terminal window. To turn on system messaging again, execute the command:

**logging console level**

*Level* - Limits the logging of messages displayed on the console to a specified level:  
**emergencies, alerts, critical, errors, warnings, notifications, informational, debugging**

#### **SSH2 connection issues:**

There appears to be an issue with the implementation of SSH2 for the Dell O/S version 2.0.0.12.

If you are using the SSH2 connection 'Method' in CatTools to connect to your Dell device and have set the Username and Password fields values correctly, but still see the Info Log error :

`"Failed to determine hostname - timeout"`

then check the version of the Dell O/S. This issue is isolated to the 2.0.0.12 version of the Dell O/S and versions prior to this are not effected.

The problem has been acknowledged by Dell and we have been informed (as of March 2008) that this will be resolved in their next firmware release.

This issue is not specifically limited to CatTools, and may also effect other clients used to connect to the device using the 2.0.0.12 version of the Dell O/S. For example: PuTTY V0.60 has the same problem (although PuTTY 0.58 does not).

### 3.3.3 ASA dap.xml file

The DAP (Dynamic Access Policies ) configuration of an ASA (v8.0) is stored in a file called dap.xml on the flash memory, not as part of the running config or startup-config file. A normal CatTools Backup activity will not backup this file.

To get CatTools to backup this file you need to create a second activity to TFTP the file off your device.

Creating a [Device.CLI.Send commands](#) activity with the following commands should complete the task.

**copy disk0:/dap.xml tftp://tftpserver-name/dap.xml  
dap.xml**

```
tftpserver-name  
%ctDeviceName.%ctDateISO.dap.xml
```

### 3.3.4 HP2500 series switches

CatTools supports 24xx, 25xx and 26xx switches in command line mode.

To make these switches enter command line mode automatically, you need to do the following...

- go to command line
- type 'setup'
- under Logon Default set to CLI
- save changes

### 3.3.5 3Com superstack switches

#### CatTools 3Com switch issues

#### What switches are supported?

3Com SuperStack Switches:

610 / 630 / 1100 / 3300 / 3300 FX / 3300XM / 3300TM / 3300SM / 3300MM

#### Background:

Although telnet is not the most elegant, nor efficient way to obtain the configuration of a 3Com switch, there aren't many other practical options. By capturing the output of various 'display' commands, CatTools provides a human readable list of settings in case of switch failure. Since 3Com doesn't provide a nice "show config" type command, this backup method is the next best thing. Because of all the commands required to obtain a complete list of settings, the backup can take up to 5 minutes per switch. A future version of CatTools may use SNMP or HTTP to obtain the config.

#### Warning!

While accessing the 3Com switch CLI, at odd times and for no obvious reason, the switch will perform a warm reset. This will cause the telnet session to disconnect. All network traffic will stop flowing until the switch has finished its start-up sequence. This can take around 15 seconds.

The switch appears to reset more often when the telnet session originates from a high speed port (100MBps). Once the switch has warm started, it is more prone to crashing again. A power reset (remove power cord) does help to return the switch to a stable state and reduce the chance of a crash occurring.

The crash does not appear to be associated with any particular command.

I have seen these crashes occur with all versions of switch agent software, up to and including version 2.62.

Using the Admin username to login instead of Security reduces the occurrence of crashes.

Hopefully this issue can be resolved in a future release of 3Com switch software. Any additional feedback would be appreciated.

**Recommended backup plan.**

Because of the chance of a switch crash, please only backup the devices during times of low traffic use. Also, reduce the frequency of your backups. Only make a backup when you believe the config might have changed. Use the Admin username instead of the Security username to login.

At this stage the config will always appear to have changed. This is because various packet counters and stats are included in the captured config. These values will change with each backup. A future version of CatTools may parse the captured data better and produce a cleaner list.

**What commands are issued?**

CatTools issues the following commands and captures the output to a file.

- 1 "bridge display"
- 2 "bridge multicastFiltering routerPort list"
- 3 "bridge port address list all"
- 4 "bridge port summary all"
- 5 "ethernet summary all"
- 6 "feature resilience detail"
- 7 "feature trunk summary"
- 8 "ip interface display"
- 9 "snmp trap display"
- 10 "system display"
- 11 "system inventory"
- 12 "system module display"
- 13 "system security access display"
- 14 "system security user display"
- 15 "bridge vlan summary all"
- 16 "bridge vlan detail" (Defined VLAN numbers)
- 17 "bridge port detail" (1 to number of switch ports)
- 18 "feature trunk detail" (1 to number of trunks)

**Device setup.**

Use the Username / Password fields to enter the username and passwords for the device. Leave the VTY and enable passwords blank as they are not used.

The standard 3Com usernames are: Monitor, Admin, Manager and Security. You will need to use the Security login if you want to make changes to passwords etc.

**Accessing switch stacks.**

A switch stack consists of up to 4 switches connected via matrix cables. Each switch has a unit number (1 to 4) which identifies it. The IP address will be the same for each device in the stack.

Once logged in, you switch between the stack units by issuing the 'system unit *N*' command where *N* is the number of the stack unit you want to session to.

Refer to the '[Connecting via a session](#)' section as to how to set up the devices in CatTools.

### **Sending commands to the CLI via CatTools.**

By using the [Device.CLI.Send commands](#) at enable prompt menu, you can send commands to the 3Com switch CLI. Some examples are given below.  
The registered version of CatTools allows you to capture the output to a file.

### **To change the password for the user "admin" use the commands:**

```
system security user modify admin  
newpass  
newpass  
private
```

### **To add port 3 to VLAN 2 with tagging use the command:**

```
bridge vlan addport 2 3 802.1Q
```

### **To drop back a menu item, use the "q" (quit) command.**

## **3.3.6 Connecting via a Cisco Terminal Server**

CatTools V3 supports a Cisco Terminal Server device type.

This device type is designed to enable CatTools to connect to devices physically connected to a terminal server type device via the console port. It should support authentication to both the terminal server device and to the device connected to it, or to just one or other of the devices. It should support either local or remote authentication. This type of connection is sometimes known as reverse telnet.

To set up a terminal server:

1. add a new device to CatTools using the Cisco Terminal Server device type
2. set the device Host Address to the correct IP address of the device
3. set the Connect via to Direct connect
4. set the authentication on this device to the passwords required when using a port of the terminal server

To use the terminal server to connect to a device make the following settings on the device that is attached to it:

1. set the CatTools device type to the type that corresponds to the actual device type
2. set the Connect via to connect via the terminal server device
3. set the Method to telnet
4. set the Port to the port on the terminal server the device is connected to. i.e. Async Port 1 = TCP port 2001
5. set the authentication required to access the device itself

To run an activity on the device that is connected to the terminal server, just select the device itself.

When CatTools wants to connect to the device, it knows to connect to the terminal server first. It connects to the terminal server using the port that is defined to the device that is connected to via the terminal server. If any authentication is required to use the port, it uses the authentication details from the terminal server. After CatTools authenticates to the terminal server, it sends a CR to it to activate the port. It will then authenticate to the device behind the terminal server using the credentials set to that device.

There are several things to bear in mind when utilising a terminal server with CatTools.

1. The connection is normally via a comm type port. The device may well be significantly slower to access from CatTools.
2. The terminal server must be directly connected to by CatTools. CatTools does not support the use of a terminal server that has to be connected to via another device.
3. If you need to connect to the terminal server itself to say issue commands or back it up, define a separate device to CatTools using its normal device type, such as Cisco router general for a Cisco 2509 router. Set its device details up with the correct Host Address, connection method, and the like. Set the port to the normal connection method port, such as port 23 for telnet.
4. After disconnecting from a device on the terminal server, the port connection may be left open before timing out. The duration of the timeout depends on the settings on the device. Where the device requires authentication, and you try to reconnect to it before the port connection has timed out, CatTools may fail to connect with a message like "Did not receive VTY entry prompt ...". This happens because CatTools is set to send authentication to the device and the device does not require it as the port connection is still available. You can unset the authentication options for the device in CatTools, and it should then connect.

To help if this is a problem, you may want to set the `exec timeout` (Cisco router) value low to ensure the console connection does not stay up for long periods of time.

### 3.3.7 Connecting via a session

The Session connection method was added to provide support for the many and varied methods by which interconnectivity can be achieved. Connections using this method can be achieved from one host device to any virtual device / blade / session or module.

To make use of this feature you need to setup two devices in CatTools. One that represents the parent device and one that represents the virtual device or session contained within it.

Connection to the parent device is as normal.

To connect to the virtual device you need to implement the following three fields on the device form.

The **Host Address** field: This should be used to execute any command that may be necessary to connect to the next device.

Examples include:

*Session 15*  
*Changeto System*  
*Session slot 4 proc 1*  
*Telnet*

The **ConnectVia** field: This should be set to the device you need to connect to first in order to establish the session.

The **Method** field: This should be set to *Session*.

This feature is currently limited to the following device types.  
If you need it added to a device type that is not listed then please let us know.

Cisco.Router.General  
Cisco.Switch.IOS  
Cisco.Switch.CatOS  
Cisco.Firewall.ASA  
Cisco.Firewall.PIX  
Cisco.ACE  
3COM.Switch.SSII

The following example illustrates how this concept works:

A Cisco 6509 has a firewall service module that you want to backup. The 6509 is CatOS and the FWSM is Cisco IOS.

First you create a device called My 6509 that would be defined as a Cisco.Switch.CatOS with all the necessary username and password information. The Host Address is the IP of the device, the Connect Via field is Direct Connect and the Method in this example is Telnet.

Next you create a second device called My FWSM defined as a Cisco.Switch.IOS. The Connect Via option is then set to be My 6509 and the Method as Session. In the Host Address field you enter the command that is necessary to access the module. In this example it is Session Slot 4 Proc 1.

Any necessary login details are entered on the Passwords tab.

Finally to backup the FWSM you would create a backup activity and assign My FWSM to it. When CatTools runs the backup activity it identifies that to get to My FWSM it needs to go through My 6509 and so logs into that device first. It then issues my custom command to establish the connection and then finally backs up the device as defined by the Cisco.Switch.IOS device type.

**DISCLAIMER / WARNING:** Normal communication between CatTools and the connected device is a process of sending commands and getting back known responses. We know what responses to expect because we know the OS of the device we are connected to. The Session connection method however can be used to connect from one device to another. In most cases these parent / child devices will be the same OS but in some cases they won't be. eg Cisco 6509. For that reason the logic behind this particular connection routine is different to all others. It works in reverse. As we do not know what a login prompt (if one even exists) might look like on the second device we can not use that as a known good response. Instead we look for the known bad responses of the first device at the time the connection command is issued. If anything other than a known error is received it is assumed that the response is the banner or login or prompt from the second device. At this point control is handed over to the script of the device-type of the second device to process authentication. It is possible that the connection will not be established and that an error message will be returned that is not trapped for. In this case CatTools will think that it is on the second device and start performing the activity scheduled. If the OS's are similar enough the commands may be valid ones and the instructions will be processed on the wrong device.

For this reason we strongly recommend that you thoroughly test all Session connections before putting them into production and that you monitor them closely.

### 3.3.8 SSH2 for Cisco and Netscreen

Many Cisco IOS devices support SSH2 from Version 12 on.

Netscreen devices with V5+ of the IOS also support SSH2.

To enable CatTools to connect to a Netscreen device there is a variant of the SSH2 protocol called **SSH2-nopty** that may need to be used.

This is in relation to the pseudo terminal that is used in \*nix when using SSH connections to a command processor, but may not be required by network devices.

### 3.3.9 Cisco VPN

If you are adding a Cisco VPN device in CatTools, we highly recommend you first try setting your device up using the **Cisco.VPN** device type.

The Cisco.VPN device type is more intelligent in determining which commands it need to send to navigate the menu driven system, which the Cisco VPN devices use.

By using the **Cisco.VPN** device type it should in most cases handle any differences within the number systems used for accessing each menu, sub menu or menu item. If you still have issues when running a configuration backup of a Cisco VPN, then please contact us using the [Technical Support](#) form.

For reference, Cisco VPN devices normally have a menu interface that CatTools accesses by sending a string of menu item numbers that invokes the display of the config.

An older Cisco 3002 may require 2.5.1. The **Cisco.Older.VPN3002** uses this by default.

The older VPN IOS prior to V4.0 uses 2.6.3 as the menu sequence. The **Cisco.Other.VPN3000** uses this string by default.

The later IOS, V4.1, uses 2.8.3 as the menu sequence. The **Cisco.V4.1.VPN3000** uses this by default.

The intermediate V4.0 IOS requires 2.7.3 (none of the above three Cisco VPN device types support this).

The **Cisco.VPN** device type should be able to handle ANY of the above command variations, so always try this device type first.

If the **Cisco.VPN** device type isn't able to backup your device, then try one of the other VPN types above. If none of the VPN3000 device types backup your device, try using the alternate command field of the backup activity and enter 2.7.3.

Note: that if you use an alternate command it overrides the default commands for all selected devices in the activity.

### 3.3.10 Cisco WAAS

Outlined below is a process to enable the backup of the database from a Cisco WAAS.

The general procedure used may be modified for other devices which also create a local backup file which then needs to be transferred to the host machine.

The process involves running two external scripts from the Device.CLI.Send commands activity;

1) The first script creates the backup file and stores the filename in a scripting dictionary for use later.

2) The second script issues commands to FTP the file to the local machine. (It relies on the host



machine running its own FTP server.)

The two scripts can be found at <http://thwack.com/media/p/61484.aspx>

The files should be copied into the UserScripts folder in the main CatTools folder. (You may need to create this folder)

The Cisco WAAS should be added to CatTools as a new device using the Cisco.Switch.IOS script.

A new Device.CLI.Send Commands activity should be created and the following lines added to the list of commands on the Options panel.

```
%ctRunExternalScript("C:\Program Files\CatTools3\UserScripts\WAASDumpName.txt","cms
database backup")
%ctRunExternalScript("C:\Program Files\CatTools3\UserScripts\FTPCopyWAASDump.
txt","<ftpHostname> 192.168.1.10 <ftpUsername> admin_backup <ftpPassword>
mypassword <ftpFileDirectory> backup.dump")
```

Note that the items in bold will need to be changed to the correct values for your FTP server.

The scripts themselves are well commented and could form the basis for further user customisation.

### 3.3.11 Backing up device via SFTP/SCP

Your device or network security policies may require transferring of files securely over SFTP or SCP.

Although CatTools does not provide a specific activity to do this (similar to the [Device.Backup.TFTP](#) activity), you may still be able to transfer using the [Device.CLI.Send commands](#) activity.

If you require an SFTP/SCP Server, you can download the [SolarWinds SFTP/SCP Server](#) from the SolarWinds web site.

The SolarWinds SFTP/SCP server runs as a service, but some basic configuration may be necessary to ensure the SFTP/SCP server behaves in a way that works best within your environment.

#### Notes:

1. An SFTP & SCP Server is not an FTP server. SFTP & SCP and FTP are different protocols. You cannot connect to a SFTP & SCP server with an FTP client.
2. Not all CatTools device type scripts will support SFTP/SCP transfer via the **Device.CLI.Send commands** activity. Some devices prompt with verification messages (for example: "Address or name of remote host"; "Destination filename"; "Destination username") which may need device script updates to handle. If you find your device does not work, please contact our CatTools Technical Support Team using the [Technical Support](#) form on our [web site](#).

## 3.4 Unsupported devices or device activity

Additional device types are being added all the time.

For an up to date list of the devices types currently available in CatTools and the activities supported for each of the device types, please see the [device matrix](#) on our [web site](#).

If the device type required to support your device is not listed in the matrix, then see our [Knowledge Base article](#) which lists the kind of details we will require in order to look into

adding support for your device, then inquire via [thwack](#), the SolarWinds online community site, for further information.

If there is a listing to support your device, but the activity you require is not currently supported by the device type then please [contact us via thwack](#), providing us with as much detail as you can and if possible a PuTTY capture of the relevant command(s).

## 4 Activities

CatTools Activities enable you to perform automated actions on your network devices.

- [Overview](#)
- [The Edit scheduled activity details form](#)
- [Activities list](#)
- [Internal functions](#)
- [Creating a custom activity](#) - how to create your own custom activities for CatTools
- [Unsupported activities for a device](#) - information and links

### 4.1 Overview

An Activity is an internal task that CatTools performs against one or many of your network devices that you have entered into the CatTools database.

It can be scheduled to run at specific times using the Timer or it can be run immediately by using the 'Run Now' button.

The Activity will launch clients to go out and query your devices and obtain the information needed to complete the task.

It will collect all of the information returned and then act upon it either storing it for future use or analysing it there and then.

It will then report to you via email the results.

A basic example would be to Backup the Running config of all of your network devices. To do this you would :

- Add a new activity of type [Device.Backup.Running Config](#)
- Click the 'Select All' button on the Devices tab to associate all of your network devices
- Click 'OK' to save the Activity
- Click 'Run Now' to launch the activity.

The progress of the Activity as it runs can be monitored using either the Info log or Display panes.

### 4.2 The Add/Edit scheduled activity details form

This section provides an overview of the interface used to add and edit an Activity. Below is an example of the **Edit scheduled activity details** form.

**Edit scheduled activity details**

Activity | Time | Devices | E-mail | Options

Type: Device.Backup.Running Config

Name: Device.Backup.Running Config

Description:

Persistence: Permanent

Report file: C:\CatTools V3\Reports\Device.Backup.Running Config.txt

☐ Overwrite existing report file.

Client threads: Maximum available

☒ Retry failed devices

Attempts: 2

Interval: 1 minutes

☒ Consolidate error reporting

OK Cancel

### 4.2.1 Activity tab

The Activity tab is where you define what sort of activity you are creating as well as its core details:

- **Type:** This drop-down enables you to choose the [type of Activity](#) you wish to set up.
- **Name:** This drop-down identifies the activity and should be unique.
- **Description:** A free text field where you can provide more detail about the purpose of the activity.
- **Persistence:** This drop-down toggles the lifetime of the activity between one that is permanent, or one that you only wish to run once.
- **Report file:** A path to a text file which in most cases holds the run history of the activity.
- **Overwrite existing report file:** This check box indicates whether the report file will be overwritten or appended to.
- **Client threads:** A drop-down which allows you to choose the number of simultaneous activities for load balancing purposes. This is limited by your license type but in essence can be thought of as the maximum number of concurrent devices that CatTools will talk to at any one time. This setting is only valid for activities that use clients to go out and talk to devices. Obviously the more threads that are running the faster CatTools will be able to complete the assigned task.
- **Retry failed devices:** This check box allows you the option to retry any devices that may have failed in the course of the activity. Between each attempt the activity will pause for an interval of X minutes. During this time the activity is still classed as running / busy.

- **Attempts:** The number of times that CatTools should retry any failed devices from this activity. Valid range is 1 to 10 attempts.
- **Interval:** The number of minutes to pause between each retry. Valid range is 1 to 60 minutes.
- **Consolidate error reporting:** This checkbox allows you to specify how level 1 errors are reported by this activity. This option is provided primarily for those that don't care what the specific error was, only whether an error occurred. For example, if you are backing up your devices and one of the devices fails to connect on the first attempt but then is successful on the second attempt you don't necessarily care about the initial failure. Your concern is "was a backup obtained or not". In this example you would have this option ticked. If do you want to see all errors that occurred on all retry's then you should untick this option.

## 4.2.2 Time tab

The Time tab lets you set up the scheduling of the activity.

### Times

You may select a Reoccurring schedule, or one that runs the activity at specific times.

### Days of the week

You can select which days of the week you wish to run the activity on by checking/ticking the appropriate box.

### Dates

You may select dates to start and end the running of an activity.

Custom schedule dates and times can be added, removed, or re-ordered using the Add, Remove, Move Up, Move Down, buttons.

You should exercise care when using this option:

- If an activity won't start running after you have set it up, check that the current date is not before the Start date you have set.
- If an activity stops running for some reason, check whether the current date is past the End date you have set.

### Favorite schedules:

You can set up Favourite schedules that can be applied to different activities.

Once you have created a schedule you can save it using the Save As button, and re-use this schedule for multiple activities by Loading it as required.

## 4.2.3 Devices tab

The Devices tab lets you associate Devices with the Activity.

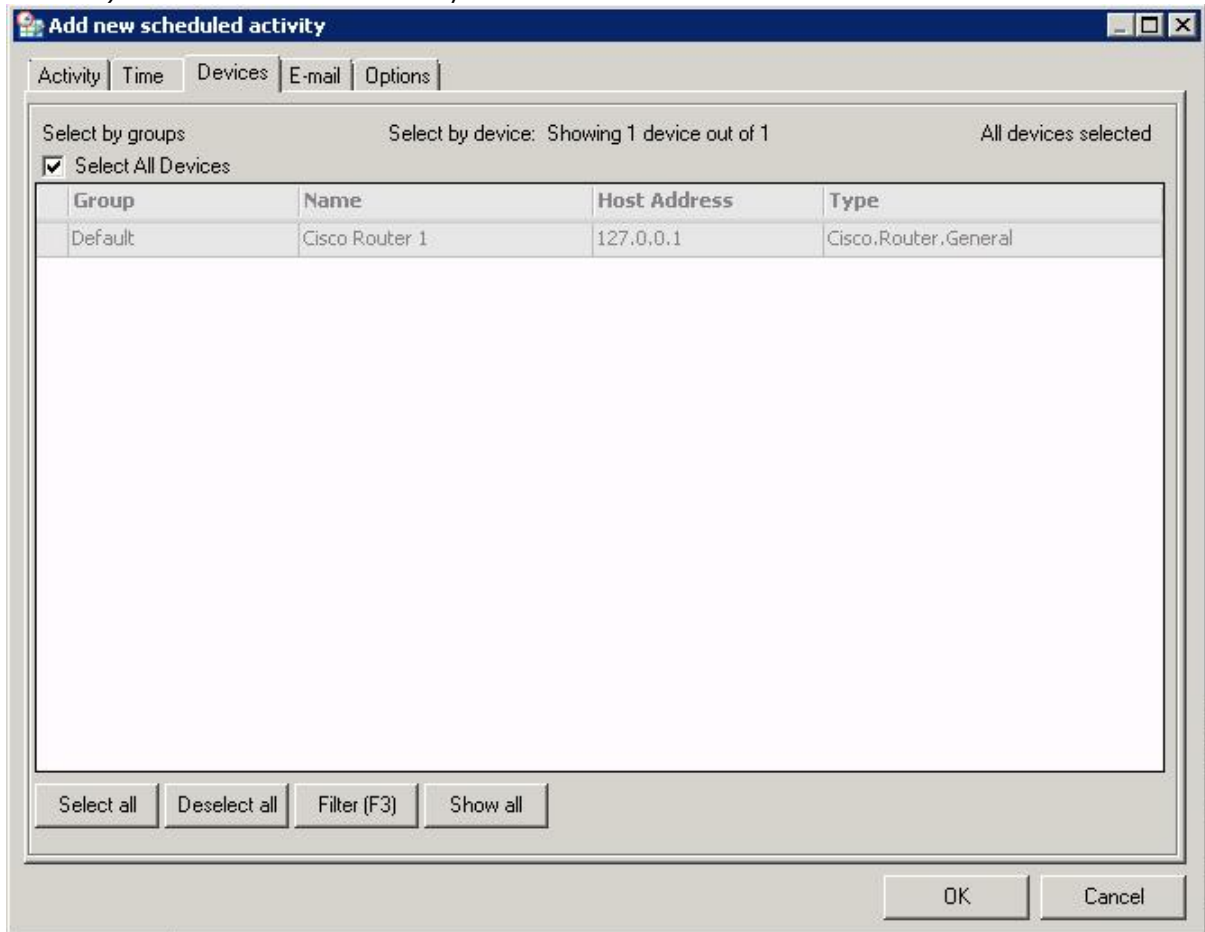
Each device has a tick box associated with it indicating whether or not the device is currently associated with this activity.

There are four buttons on the bottom of the tab:

- **Select all** Selects all the devices shown
- **Deselect all** Un-selects all the devices shown
- **Filter (F3)** Opens the "Database Filter" window, so you can filter the display of devices. Note that the text you enter in the Contains field is case sensitive. The operation of this function is described more fully [here](#)
- **Show all** Disables any active filters and shows all devices in the database

There are a number of methods you can use to select devices:

- You can simply click on the check box of each device.
- You can double click on the row containing the device.  
You can select a number of devices in the list using the standard Windows list selection methods and then right click to show a context menu. Selecting the **Enable selected** option will check all of the devices that are highlighted.
- You can use the **Filter** method to display a group of devices, such as including all devices whose group contains "Head Office". When the selected devices are displayed you can click on the **Select all** button to check all the displayed devices.
- **Select All Devices** Checking this option allows all devices within the database (current and future) to be added to this activity.



#### 4.2.4 Email tab

The E-mail tab allows you to select notification alerts for the current activity.

There are two types of email notifications. Errors and Reports. Both will send the email to the default recipient set in the general [CatTools Setup](#). You can however override this by selecting an Alternative e-mail address here.

**The Error email** is sent when a level 1 critical error is triggered at any point in the activity by any one of the devices the activity is being run against. If you wish to receive this sort of notification then check the Email errors from this activity box.

**The Report email** is sent at the end of the activity and summarises the results of the activity. An optional HTML formatted report can also be attached to the email, by checking the Attach additional report with sortable columns check box.

If you wish to receive this notification then check the Email statistics/reports from this activity box.

## 4.2.5 Options tab

The Options tab is specific to the activity type.

Refer to the specific activity type for more details.

[DB.UpdateDevice.Password field](#)

[DB.UpdateDevice.Text field](#)

[Device.Backup.Running Config](#)

[Device.CLI.Modify Config](#)

[Device.CLI.Send commands](#)

[Device.ConnectivityTest.Login](#)

[Device.ConnectivityTest.Ping](#)

[Device.InterDevice.Ping](#)

[Device.TFTP.Upload Config](#)

[Device.Update.Banner](#)

[Device.Update.Password](#)

[Report.ARP table](#)

[Report.CDP Neighbors table](#)

[Report.Compare.Running Startup](#)

[Report.Compare.Two files](#)

[Report.Error info table](#)

[Report.MAC address table](#)

[Report.Port info table](#)

[Report.SNMP.System summary](#)

[Report.Version table](#)

[Report.X-Ref.Port MAC ARP](#)

## 4.3 Activities list

In this section you will find a complete list of all of the CatTools Activities.

### 4.3.1 DB.Backup.CatTools

The **DB.Backup.CatTools** activity allows you to automate the backing up of your CatTools database.

It does not run against your physical network devices as most of the other Activities do. Instead it takes a backup of your existing CatTools database and makes a backup to the specified location. It will also try to squeeze the backup after completing the copy.

For more information see [Setting up the activity](#)

#### 4.3.1.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [DB.Backup.CatTools](#) as the Activity Type:

**Time** tab:

If you wish to run the Activity at a scheduled time set it here.

**Devices** tab:

You need not enter any devices.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

Select the location to backup the database to, you may use [Filename Variables](#) as part of the filename.

### 4.3.2 DB.UpdateDevice.Password field

The **DB.UpdateDevice.Password field** activity allows you to maintain your passwords in the CatTools database device password fields.

It does not run against your physical network devices as all the other Activities do but is instead used to do batch processing against multiple devices within the CatTools database Device table.

An example use of this Activity might be:

Your network devices use a RADIUS or TACACS server for remote authentication. You have just changed the password on the RADIUS/TACACS server and you now need to update all of your devices within CatTools with this new information.

Instead of editing each device individually you would create a new **DB.UpdateDevice.**

**Password field** activity which will apply the new password to all of your Devices at once.

For more information see [Setting up the activity](#)

#### 4.3.2.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [DB.UpdateDevice.Password field](#) as the Activity Type:

**Time** tab:

If you wish to run the Activity at a scheduled time set it here.

**Devices** tab:

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

- **VTY Password:** tick box. Allows you to change the CatTools database field value for the VTY Password field.
- **AAA Password:** tick box. Allows you to change the CatTools database field value for the AAA Password field\*
- **Enable Password:** tick box. Allows you to change the CatTools database field value for the Enable Password field.
- **Console Password:** tick box. Allows you to change the CatTools database field value for the Console Password field.
- **SNMP Read:** tick box. Allows you to change the CatTools database field value for the SNMP Read field.
- **SNMP Write:** tick box. Allows you to change the CatTools database field value for the SNMP Write field.

**Stop on error:** tick box. Should an error be encountered at any stage of the processing the activity will quit the job and return control to CatTools.

\* Note: the **AAA Password** field in the CatTools database corresponds to the 'Password:' field (below the 'Username:' field) on the [Passwords](#) tab of the Device information/setup form. This field can be used to store AAA/TACAC/RADIUS/Local authentication or SSH passwords.

### 4.3.3 DB.UpdateDevice.Text field

The **DB.UpdateDevice.Text field** activity allows you to update any of the text fields in the CatTools database Device table.

It does not run against your physical network devices as all the other Activities do but is instead used to do batch processing against multiple devices within the CatTools database Device table.

An example use of this Activity might be:

Your network devices use a RADIUS or TACACS server for remote authentication. You have just changed the username on the RADIUS/TACACS server and you now need to update all of your devices within CatTools with this new information. Instead of editing each device individually you would create a new **DB.UpdateDevice.Text field** activity which will apply the new username to all of your Devices at once.

For more information see [Setting up the activity](#)

#### 4.3.3.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [DB.UpdateDevice.Text field](#) as the Activity Type:

**Time** tab:

If you wish to run the Activity at a scheduled time set it here.

**Devices** tab:

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.



**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Database Field:** (x4) tick boxes activate database field dropdown box and corresponding database value text field. This allows you to select an item from the device details you wish to update in the database.

examples:

**Info\Port** refers to the 'Port:' field on the [Device info](#) tab of the Device Information/setup form.

**Username\Password\*** refers to the 'Username:' field on the [Passwords](#) tab of the Device Information/setup form

**New database value:** (activated by the above) allows you to enter the value you wish to change the selected database item to.

**Stop on error:** tick box. Should an error be encountered at any stage of the processing the activity will quit the job and return control to CatTools.

\* Note: the Username\Password database field in the activity will update the **AAA Username** field in the CatTools database.

The **AAA Username** database field corresponds to the 'Username:' field on the [Passwords](#) tab of the Device information/setup form.

This field can be used to store AAA/TACAC/RADIUS/Local authentication or SSH usernames.

### 4.3.4 Device.Backup.Running Config

The **Device.Backup.Running Config** activity will make a backup of a device's running configuration and compare it to the current config file on disk. If there are differences, the current config is moved to the "Dated Configs" folder and the filename is appended with the current date. The newly downloaded config then becomes the current config. An HTML "diff" report is created in the "Reports" folder and a copy is e-mailed to the nominated person.

**How it works. Step by step:**

1. A connection is made via SSH or telnet to the device
2. The "Show running configuration" (or similar) command is issued
3. The config is collected and stored on disk in a temporary file
4. A check is made to see if there is an existing config file on disk for this device
5. If not, the newly downloaded config is placed in the "Configs" folder as the current device backup
6. If a current config file is present, it is compared to the newly downloaded config
7. A text and HTML report is made of the differences (if any) and stored in the "Reports" folder
8. If differences are found, a copy of the difference report is sent via e-mail
9. The current config file is moved into the "Dated configs" folder and appended with the current date
10. The newly downloaded config file is put in the "Configs" folder and becomes the current device backup

For more information see [Setting up the activity](#)

#### 4.3.4.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Device.Backup.Running Config](#) as the Activity Type:

**Time tab:**

If you wish to run the Activity at a scheduled time set it here.

**Devices tab:**

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:****Use alternate command:**

The command issued to backup the running config will depend on the device type. For example, a Cisco router will be sent the command "Show running-config". If you wanted to send a different command, enter it in this field. Remember that the command must be suitable for all the devices selected for this activity. If your alternate command is only suitable for a Cisco VPN, then make sure only Cisco VPN devices are selected under the devices tab.

**Current config file:**

Specifies the folder and file name of where to store the current device configuration.

Default: ...\\Configs\\%GroupName%\\Config.Current.Running.%BaseFile%.txt

The variables %GroupName% and %BaseFile% will be replaced at run time and produce a folder and file name.

For example:

C:\\Program files\\Cattools\\Configs\\Default\\Config.Current.Running.Sales\_Router.txt

This value is where any new configuration files will be stored. If an existing file exists a comparison will be done against the newly downloaded file. If there are no changes, no files will be modified.

**Dated config file:**

Specifies the folder and file name of where to store dated configuration files. The current file is moved to the dated config folder when changes are detected.

Default: ...\\Dated Configs\\%GroupName%\\Config.Dated.Running.%BaseFile%.%DateISO%-%TimeHHMM%.txt

The variables %GroupName%, %BaseFile%, %DateISO% and %TimeHHMM% will be replaced at run time and produce a folder and file name.

For example:

C:\\Program files\\Cattools\\Dated Configs\\Default\\Config.Dated.Running.Cisco805.20031006-1914.txt

See the section [Filename Variables](#) for a list of variables you can use.

**HTML compare report:**

Specifies the folder and file name of where to store the HTML diff report. This diff report file contains a side by side comparison of the old and new configuration. Changes are highlighted in different colors.

Default: ...\\Reports\\ConfigChanges\\%GroupName%\\Config.Changes.Running.%BaseFile%.%DateISO%-%TimeHHMM%.htm

The variables %GroupName%, %BaseFile%, %DateISO% and %TimeHHMM% will be replaced at run time and produce a folder and file name.

For example:

C:\Program files\Cattools\Reports\ConfigChanges\Default\Config.Changes.Running.  
Cisco805.20031006-1914.htm

**Text summary diff report:**

Specifies the folder and file name of where to store the text based diff report. This diff report file contains a list of the changes found in the old and new configs. This report is a smaller and simpler way to view the changes. Unlike the HTML report, where both the old and new configs are shown in full. The text report only shows the lines that are different.

Default: ...\\Reports\\ConfigChanges\\%GroupName%\\Config.Changes.Running.%BaseFile%.%DateISO%-%TimeHHMM%.txt

The variables %GroupName%, %BaseFile%, %DateISO% and %TimeHHMM% will be replaced at run time and produce a folder and file name.

For example:

C:\Program files\Cattools\Reports\ConfigChanges\Default\Config.Changes.Running.  
Cisco805.20031006-1914.txt

**Only notify by e-mail if configs have changed:**

This allows you to run configuration backups as often as you want, but only receive an e-mail notification when changes are detected.

**Attach reports to e-mail:**

Sub options:

- HTML compare report only (default)
- Text diff report only
- Both reports

Allows you to choose what level of diff reports you are sent via e-mail.

If you want the full comparison reports, have it set to send you the HTML compare report. If you just want the changes sent, use the text report.

If you are sending the reports via an insecure medium (corporate network or internet), be aware that the HTML reports contain your full device configuration. It is recommended that you either don't send HTML reports over an insecure medium, or you protect the reports by using a zip password. The longer the password, the harder it is to crack.

**Zip attachments:**

To reduce network traffic and allow for password protection, you can choose to zip the attachments. This is the recommended and default option. It keeps the e-mail smaller and places all the files in a single attachment (great when you have 400 devices).

**Password protect zip file:**

Protect your configs and reports by adding a password to your zip file. The longer the password, the harder it is to crack the protection. The password protection will protect your configs from prying eyes while it is in transit to your e-mail box.

#### 4.3.4.2 Why you should backup your configs

Why it is important to backup your configs?

The [Device.Backup.Running Config](#) activity is one of a number of default activities provided with the CatTools program. It is designed to copy the currently running configuration from the selected device, into a text file on your local file system.

This is an important distinction: it copies the configuration parameters that are currently running on the device. Many devices have their own local storage, where one *or more* versions of their configuration files may be stored. Most network devices managed by CatTools will have a start-up config file that they load and run when the power is switched on.

Because this running config can be changed in real-time, by someone logged into the device via a telnet session for example, the running config may no longer be the same as the start-up config. Capturing these different versions of your device configurations is exactly the sort of thing that CatTools is for.

Say you have a night-shift engineer who encounters a problem on your network, hurriedly telnets into a router and makes some changes to that device that restores service to your internal customers, but in the panic forgets to document or save these changes. If you power cycle that device at some later time, those changes will be lost, potentially recreating the problem, only this time it might be during your busiest period of the day. Now you have the problem back, but the fix is gone and the engineer is at home in bed. You may have to start the whole problem resolution process from scratch, costing your organisation money and costing you (and your department) credibility.

There are several ways that the activity might be useful in this reasonably common scenario.

Firstly, it is best practice in change management to always have a roll-back contingency. Naturally, such a plan is no good if you can't actually recover and re-install the old configuration. You should make it a habit to always capture a restorable copy of the current configuration before making any changes to it, and CatTools is a very effective means of doing this.

Secondly, CatTools contains a Diff Report function which runs as a part of the activity. This compares the configuration that has just been downloaded to the previous copy stored on file. If any changes are detected then it creates a report in HTML format that highlights the lines that are different (that's why it's called a Diff Report). The default setting is for this report to be emailed to you as soon as it is created, as well as being stored in the Reports directory.

So if our hypothetical night-shift engineer had run a [Device.Backup.Running Config](#) before and after they had made changes to the router, then you'd have a report showing you exactly what changes had been made, plus you'd have two copies of the config in two text files; one before the changes and one after.

You could use either of these files to restore the router to the configuration you want. Or, you could use the Diff Report to manually identify and recreate the changes.

#### 4.3.5 Device.Backup.TFTP

PLEASE NOTE: We would recommend using the [Device.Backup.Running config](#) activity to backup your config files. However, the Device.Backup.TFTP activity may serve as a useful alternative in certain circumstances.

The **Device.Backup.TFTP** activity will make a backup of the specified file and optionally compare it to the current file on disk. If there are differences, the current file is moved to the

"Dated Configs" folder and the filename is appended with the current date. The newly downloaded file then becomes the current file. A HTML "diff" report is created in the "Reports" folder and a copy is e-mailed to the nominated person.

Although this activity is designed to mostly replicate the Device.Backup.Running config activity via TFTP, it is not limited to the device-config. You can specify other files, such as VLAN.dat. You also have the option whether to do the compare. Some files may be in binary format and so can't be compared. Finally you can also specify your own command set to TFTP the file.

**How it works. Step by step:**

1. A connection is made via SSH or telnet to the device
2. The commands to TFTP the specified file are issued
3. The file is collected and stored in the TFTP folder
4. A check is made to see if there is an existing file on disk for this device
5. If not, the newly downloaded file is placed in the specified folder as the current file backup
6. If a current file is present, it is compared to the newly downloaded file
7. A text and HTML report is made of the differences (if any) and stored in the "Reports" folder
8. If differences are found, a copy of the difference report is sent via e-mail
9. The current file is moved into the "Dated configs" folder and appended with the current date
10. The newly downloaded file is put in the specified folder and becomes the current file backup

For more information see [Setting up the activity](#)

**NOTE:** - This activity has only been added to the following scripts.

AlliedTelesis.AlliedWare.Plus  
Cisco.Router.General  
Cisco.Switch.IOS  
Cisco.Switch.CatIOS  
Cisco.Firewall.PIX  
Cisco.Wireless.Lan  
Fortinet.FortiOS.General  
Generic.Device  
HP.Switch.2500  
NEC.Univerge.IX  
Nortel.Switch.Ethernet  
Nortel.Switch.NoCLI

#### 4.3.5.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Device.Backup.TFTP](#) as the Activity Type:

**Time tab:**

If you wish to run the Activity at a scheduled time set it here.

**Devices tab:**

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Optional alternate list of commands:**

To use this option remove the tick from the 'File to write to TFTP Server' checkbox (see below). Some of the CatTools device scripts will have a default set of commands that can be used to generate and TFTP the config for this activity. However, should there be no default commands for your device you can use this box to enter your own. Below is an example of the sort of commands you may enter;

```
copy running-config tftp:
10.190.2.98
%ctDeviceName-Running-Config
```

NOTE: When specifying the name of the file to be written it must include the %ctDeviceName filename variables as CatTools will use this to correctly identify the file for the compare.

**Enter commands in enable mode:**

Selecting this check box tells CatTools to enter enable mode before entering any of the commands to the device.

**Answer [yes] to any confirmation prompts:**

Some commands have confirmations prompts ("are you sure? Y or N") – so the program will automatically answer them in the affirmative, on the assumption that if you entered the commands you actually want them to be executed. If this option is unchecked, then the activity will enter an "n" in response to any confirmation requests.

**File to write to TFTP server:**

If you want to use the default list of commands (where available) to TFTP the file tick this box. Enter the name of the file you want to TFTP in the text box. i.e. running-config, vlan.dat etc.

**Current file:**

Specifies the folder and file name of where to store the current file. (By default this will be in the configs folder and use 'Config.Current.Running' as part of the file name as the most common use case for this activity is to backup the config by TFTP.) This will also be the filename that the file downloaded by TFTP will be compared against.

Default: ...\\Configs\\%GroupName%\\Config.Current.Running.%BaseFile%.txt

The variables %GroupName% and %BaseFile% will be replaced at run time and produce a folder and file name.

For example:

C:\\Program files\\Cattools\\Configs\\Default\\Config.Current.Running.Sales\_Router.txt

This value is where any new configuration files will be stored. If an existing file exists a comparison will be done against the newly downloaded file. If there are no changes, no files will be modified.

**Compare, move to dated file:**

Tick this checkbox if you wish to run a compare between the file downloaded and an existing file (as specified by the current file).

Specifies the folder and file name of where to store dated files. The current file is moved to the dated config folder when changes are detected.

Default: ...\\Dated Configs\\%GroupName%\\Config.Dated.Running.%BaseFile%.%DateISO%-%TimeHHMM%.txt

The variables %GroupName%, %BaseFile%, %DateISO% and %TimeHHMM% will be replaced at run time and produce a folder and file name.

For example:

C:\\Program files\\Cattools\\Dated Configs\\Default\\Config.Dated.Running.Cisco805.20031006-1914.txt

See the section [Filename Variables](#) for a list of variables you can use.

**Ignore Text:** Should the new file contain any lines of text that are likely to differ from the original file every time the comparison is run (such as a timestamp) and you would like to ignore these lines, they can be added here. More information is available on the [Ignore Text](#) field.

**Note:** to ignore "certificate self-signed" changes on Cisco routers, see the [block text ignore example](#).

**Only notify by e-mail if configs have changed:**

This allows you to run configuration backups as often as you want, but only receive an e-mail notification when changes are detected.

**Attach reports to e-mail:**

Sub options:

- HTML compare report only (default)
- Text diff report only
- Both reports

Allows you to choose what level of diff reports you are sent via e-mail.

If you want the full comparison reports, have it set to send you the HTML compare report. If you just want the changes sent, use the text report.

If you are sending the reports via an insecure medium (corporate network or internet), be aware that the HTML reports contain your full device configuration. It is recommended that you either don't send HTML reports over an insecure medium, or you protect the reports by using a zip password. The longer the password, the harder it is to crack.

**Zip attachments:**

To reduce network traffic and allow for password protection, you can choose to zip the attachments. This is the recommended and default option. It keeps the e-mail smaller and places all the files in a single attachment (great when you have 400 devices). Protect your configs and reports by adding a password to your zip file by typing it in the text box. The longer the password, the harder it is to crack the protection. The password protection will protect your configs from prying eyes while it is in transit to your e-mail box.

### 4.3.6 Device.CLI.Modify Config

The **Device.CLI.Modify Config** activity allows you to modify the configuration of device or a selection of devices.

CatTools will Login to each selected device in turn, execute the configuration changes you describe, and will log any failures to the report file:

...\Reports\Device.CLI.Modify Config.txt

Whereas the [Device.CLI.Send commands](#) activity does this from the normal command line prompt (or Enable mode if that option is checked) this activity executes commands in the Config mode (Config terminal).

**Note:** A simple way to test this activity and get more used to it yourself is to do a change of interface description, using the commands in the example below:

```
int e0
description this is a test
```

Try copying it and using a different description, so you can run them in sequence and flip the description back and forth.

Maybe combine it with a [Device.Backup.Running Config](#), so you can see the changes in the config files, the email reports it generates, and so on.

For more information see [Setting up the activity](#)

#### 4.3.6.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Device.CLI.Modify Config](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "At Midnight" as the Recurring option in the Times box to run the activity every night at midnight.

Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to configure from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Config commands to be entered into the device (Max 10,000 chrs):**

This field is where you enter the specific commands to change the configuration of the selected device(s). On a Cisco IOS router, for example, these are commands that you would normally enter in config mode (after entering "config term" at the Enable prompt).

So in a simple example, if you want to change the VTY login password for the router you would enter the following commands in this field:

```
Line vty 0 4  
password mynewpassword
```

CatTools will enter config mode and execute these commands exactly as you write them in this field, so be sure to write them exactly as you would if you were configuring the device manually via a telnet session.

**Each command must be entered on a separate line.** If you enter a blank line between commands then this will be applied just as if you pressed enter in a telnet session.

There is a limit on the size of this field, which is 10,000 characters. This gives you considerable room for putting together quite sophisticated configuration changes.

Within the commands, you can use a number of variables that are replaced by actual values when the commands are presented to the device. See the section [Command Variables](#) for a list of variables you can use.

You may also enter meta commands in the list of commands. See the section [Meta Commands](#) for an explanation.



The next option is: **Or, read commands from file:**

This option is preceded by a check-box and is followed by a normal path/filename entry field.

(This option and the one above it are mutually exclusive. If you check the box then the config commands field is deactivated, and the commands are only read from the text file you specify here.)

The rules for entering commands in the text file are the same as they are for entering them in the config commands field. Every line is read as a command to be entered in the devices config mode. A blank line between commands is read as an <Enter>.

The major difference between the two methods is that the text file doesn't have a limit on the number of characters like the config command field does.

**Save device output to file:** also has a check box and a path/filename field. It is checked by default, and the default filename is based on the device name and the date.

This is basically a device-specific record of each time the activity runs. All output from the device is recorded in the text file, exactly as it would appear in a telnet session.

Each device selected creates its own unique output file.

The following options are all check boxes and all are checked by default.

**Overwrite existing capture file:** Overwrites the file created by the output from the last run of this Activity. If unchecked, the file is appended, but a new file is created each time the date changes.

**Answer [yes] to any confirmation prompts:** some commands have confirmations prompts ("are you sure? Y or N") – so the program will automatically answer them in the affirmative, on the assumption that if you entered the commands you actually want them to be executed. If this option is unchecked, then the activity will enter an "n" in response to any confirmation requests.

**Stop on error:** Aborts the activity immediately on any error condition.

**Save running config to start-up config when complete:** copies the running config on the device to non-volatile memory, so that the modified config becomes the start-up config.

#### 4.3.6.2 When to use the Device.CLI.Modify Config activity

This activity is a useful network management tool for several reasons:

- It allows you to have very fine-grained timing control over your network
- It enables you to automate these controls, yet still use exactly the same commands as you would enter if you did the task manually.

The benefit is that you learn and use the same command line instructions in both cases, and the CatTools program itself is very simple to learn to use in quite sophisticated ways because it simply applies those commands on your behalf, just as you would via telnet.

The time controls are perhaps the most useful aspect of this activity. With different combinations of scheduled activities you can tune your network to the changing needs of your organization over the course of a typical business day or week.

CatTools is designed to manage the repetitive scheduled configuration tasks on your behalf. It removes the possibility of human error from such tasks. It provides you with a much greater confidence in the currency of your backup processes, and in your ability to roll back to a previous state using your historical backups. It provides for active management of your network and its condition, by enabling the remote execution of command line instructions across your network, at times determined by you.

For example:

If you have some switch ports that are wired to a shared conference room that is only in use during the day, you could schedule a config change to shutdown the specific interfaces at 6 p.m. and then another activity to enable the interfaces at 7 a.m. This will ensure that unauthorized staff can not hack into the network from the shared conference room after hours.

Scheduled config changes could also be used to add or remove access-list entries at specified times. Allowing access to the payroll system during business hours only.

The uses are only limited by your imagination.

#### 4.3.6.3 How to: Enter different commands onto each device

If you want to configure each of the selected devices in a different way, there are two ways to achieve this.

1. Create a separate [Device.CLI.Modify Config](#) activity for each device and set the commands to be entered in the options tab.

This method involves having a separate scheduled activity for each device you want to modify.

2. Create a single [Device.CLI.Modify Config](#) activity and select all the devices you want to modify. Then use the **Or, read commands from file** option and specify the %BaseFile% value as part of the file name. Then create a separate text file for each device that contains the commands you want to have entered into the config.

For example.

Specify a filename of `C:\CatTools\%BaseFile%.txt`

The list of devices might be...

Device name	Base filename	Config commands filename
CiscoRouter1	CiscoRouter1	C:\CatTools\CiscoRouter1.txt
CiscoRouter2	CiscoRouter2	C:\CatTools\CiscoRouter1.txt
Sales Switch	Sales_Switch	C:\CatTools\Sales_Switch.txt

\* Note that any spaces and invalid characters are replaced by and underscore ( `_` ) in the base filename.

In each of the text files, enter a list of commands that you want to enter.

For example, in the file: `C:\CatTools\CiscoRouter1.txt`, enter the commands...

```
Interface s 0/0  
Description New WAN link to head office  
No shutdown
```

Enter different commands in each text file and then run the activity. CatTools will automatically match the correct text file with the device based on the %BaseFile% value.

### 4.3.7 Device.CLI.Send commands

The **Device.CLI.Send commands** activity allows you to issue a series of CLI commands to the selected devices as if you were typing them from the normal command line prompt (or Enable mode if that option is checked).

CatTools automatically performs the following functions for this activity:

1. Login to the device
2. Handles any terminal display window length issues
3. Optionally enter enable or privileged mode
4. Enter the specified commands in sequence and wait for a valid response
5. Optionally Log each command entered and the device response to a text log file
6. Allows you to control whether to continue or stop if errors occur
7. Disconnect from the device

For more information see [Setting up the activity](#)

#### 4.3.7.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Device.CLI.Send commands](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time tab:**

Select, for instance, "At Midnight" as the Recurring option in the Times box to run the activity every night at midnight.  
Leave all other options on this tab at their defaults.

**Devices tab:**

Select the Devices you want to configure from your CatTools Host.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**List of commands to be entered into the device (Max 10,000 chrs):**

This field is where you enter the specific commands to be sent to the selected device(s).

Examples of some CLI commands that can be entered:  
(On a Cisco router)  
Clear counters  
Clear interface

(On a Cisco Catalyst CatOS switch)

Set Port 1/1 disable

Set Port 2/12-24 enable

Each command must be entered on a separate line. If you enter a blank line between commands then this will be sent just as if you pressed enter in a telnet session.

There is a limit on the size of this field, which is 10,000 characters. This gives you considerable room for putting together quite sophisticated configuration changes.

**Or, read commands from file:**

This option is preceded by a check-box and is followed by a normal path/filename entry field.

(This option and the one above it are mutually exclusive. If you check the box then the commands field is deactivated, and the commands are only read from the text file you specify here.)

The rules for entering commands in the text file are the same as they are for entering them in the commands list. Every line is read as a command to be entered in the devices config mode. A blank line between commands is read as an <Enter>.

The major difference between the two methods is that the text file doesn't have a limit on the number of characters like the command list does.

Within the commands, you can use a number of variables that are replaced by actual values when the commands are presented to the device. See the section [Command Variables](#) for a list of variables you can use.

**Enter commands in enable mode:**

Selecting this check box tells CatTools to enter enable mode before entering any of the commands to the device.

**Save device output to file:**

Selecting this check box tells CatTools to write file of all output from the device to the file specified in the filename field. It is checked by default, and the default filename is based on the device name and the date.

This is basically a device-specific record of each time the activity runs. All output from the device is recorded in the text file, exactly as it would appear in a telnet session.

Each device selected creates its own unique output file based on the %DeviceName% and %DateISO% variables.

See the section [Filename Variables](#) for a list of variables you can use.

The following options are all check boxes and all are checked by default.

**Overwrite existing capture file:**

Overwrites the file created by the output from the last run of this Activity. If unchecked, the file is appended, but a new file is created each time the date changes.

**Answer [yes] to any confirmation prompts:**

Some commands have confirmations prompts ("are you sure? Y or N") – so the program will automatically answer them in the affirmative, on the assumption that if you entered the commands you actually want them to be executed. If this option is unchecked, then the activity will enter an "n" in response to any confirmation requests.

**Stop on error:**

Stops the activity immediately if any error occurs when a command is sent to the device.

#### 4.3.7.2 Running External Scripts

**Overview**

It is possible to run an external VB script from within the [Device.CLI.Send commands](#) activity and use the return value from the external script as part of a CLI command to be executed.

When creating external scripts for use within CatTools, you will need to edit the script files using a [script editor](#).

A number of useful [Functions](#) and [Variables](#) are exposed for use by external scripts.

**Supported Device Scripts**

Cisco.Switch.IOS  
Cisco.Router.General  
Cisco.Other.CUE  
Cisco.Wireless.LAN  
Cisco.WAE  
Aruba.ArubaOS.General

**Usage**

To run an external script from within the [Device.CLI.Send commands](#) activity a command should be entered in the following format.

```
%ct RunExternal Script ( " PathToScript " , [ " VariableString " ] )
```

PathToScript;

Is the full path to the script to be run.  
It must be wrapped in quotes as shown.

VariableString;

Is a string containing values to pass to the external script.  
It must be wrapped in quotes as shown.  
It cannot contain any quotes.

It is optional as the script may not require variables.

If you need to pass more than one variable to the script then it is up to you how you achieve this within the confines of the variable string. For example you could pass "255|127|1|2" as your variable string and then use the external script to parse these into their individual number values.

Only one %ct RunExternal Script command can exist per line.

**The External Script**

The external script must have a function Main which is the scripts entry point.

Function Main must accept a string parameter. eg. Function Main( VariableString)

The script must always return a value even if it is a nul value.

The script may need to parse VariableString to separate it into individual variables for use

within the script.

The script has these [Functions](#) and [Variables](#) exposed to it.

The script can store values for use later using the functionality exposed by a scripting dictionary.

### **Example Usage**

1) In the below example a script is run which deletes all files from the TFTP folder before the running config is copied into it. The external script doesn't require any parameters and returns a nul value, after which the next command, copy running tftp is executed.

```
%ctRunExternalScript("C:\DeleteFiles.txt")  
copy running tftp  
192.168.1.200  
%ctGroupName/%ctDeviceName-Config.txt
```

2) In the below example a script is run which returns a value of the ports to enable based on the value of the StringVariable sent. The variable is 5 in this case, which could, for example, represent the slot number.

**Set Port %ctRunExternalScript("C:\PortToEnable.txt","5") enable**

So after execution of the script, the command which would actually be processed by the [Device.CLI.Send commands](#) activity in CatTools may be,

**Set Port 5/1-12 enable**

Where the writing in red is the returned value from the function, which is then inserted into the final command used.

3) In this example the hostname is changed to a new hostname. (No variables are passed )

```
Conf term  
hostname %ctRunExternalScript("C:\hostnames.txt")  
wri mem
```

### **Example External Scripts**

**(see example 2 above)**

In the example below the function returns a value based on that of the input string.

```
Function Main(inputString)  
    Dim SelectionValue  
    SelectionValue = Val(inputString)  
    Main = ""  
    Select Case SelectionValue  
    Case 1  
        Main = " 1/ 1- 12"  
    Case 2  
        Main = " 2/ 1- 12"  
    Case 5  
        Main = " 5/ 1- 12"  
    End Select  
End Function
```

**(see example 3 above)**

In this example, although the user is not passing an input string, CatTools will pass a default null string so the 'inputString' variable is still required in the function definition.

The function itself is referencing **cl.DeviceHostnameId** which is one of the variables made available to external scripts.

```
Function Main(inputString)
    Main=""
    If cl.DeviceHostnameId="MyRouter1" Then Main="MyNewRouterName1"
    If cl.DeviceHostnameId="MyRouter2" Then Main="MyNewRouterName2"
End Function
```

**4.3.7.3 Example Generic Script**

The script below is a more complete framework which you can use as a starting point for your external scripts.

It works in the following way;

- 1) Send a command to the device.
- 2) Wait until the device finishes the output generated by the command and returns the device prompt.
- 3) Parse the stored output from the device and take appropriate action.

Depending on what you need to achieve you may want to add additional 'mults' to the `local SendCommandSingle` function. By also adding matching case statements you can respond to output from the device which is not handled by the standard device script.

**Option Explicit**

' The starting place for the external script is always Function Main.

```
Function Main(inputString)
    Dim commandResult
    Dim processBufferResult
    Main = ""
    ' Call the function to send the command to the device. In this case we
    ' are using input string as the command.
    commandResult = local SendCommandSingle(inputString)
    ' If the command was processed OK then take action.
    If commandResult Then
        processBufferResult = ProcessBuffer()
        If processBufferResult Then
            ' Take action based on the result of processing the buffer
        End If
    End If
End Function
```

**Function local SendCommandSingle(CmdToSend)**

' This is a simplified version of the send command single activity used in the main CatTools client.

' The idea is that it can be used to handle specific output not catered for by the main script.

```
Dim retVal
Dim Mult(20)
Dim Choices
Dim StoredBuffer
local SendCommandSingle = False
```

```

StoredBuffer = ""

' Set up Mults - The text returned from the device is checked for any of
the mults and
' if a match is found different actions are taken depending on the mult.
Mult(1) = "--More--"
Mult(2) = "Invalid Command"
' Custom mults can be inserted here
' Mult(3) = "Device specific mult"
Mult(18) = cl.DeviceVTYPrompt
Mult(19) = cl.DeviceEnablePrompt
Mult(20) = cl.DeviceConfigPrompt
Choices = 20 'the number of mults

' Clear the buffer
cl.FlushRXBuffer

' Send the command to the device (but dont press enter yet)
If cl.WaitForEcho Then
    If cl.SendAndWaitForEcho(CmdToSend) = False Then
        cl.Log 4, "Did not receive echo of " & CmdToSend
        Exit Function
    End If
Else
    cl.SendData CmdToSend
End If

' We dont want to record the command being sent or the echo so clear the
buffer ready for a reply from device
cl.FlushRXBuffer

' Send <cr> to execute the command
cl.SendData vbCr

' Process return results
cl.Log 4, "Waiting for a response to: " & CmdToSend
Do
    ' Wait for a response from the device
    If cl.WaitForTime = 0 Or cl.WaitForTime = "" Then
        ' If the user has not specified a WaitForTime
        retval = cl.WaitForMultData(Mult, Choices, cl.RecDataTimeOut * 5)
    Else
        ' If the user has specified a WaitForTime
        retval = cl.WaitForMultData(Mult, Choices, cl.WaitForTime)
    End If

    ' Store any output from the device
    StoredBuffer = StoredBuffer & cl.RXBuffer

    ' Analyse results for the first matching mult found
    Select Case retval
    Case 0
        ' No valid data received
        If 2 > 0 Then
            cl.Log 2, "Did not receive expected response to command: " &
            CmdToSend
        End If
        Exit Do
    Case 1
        ' --more-- prompt found so send a cr to get the next block of
        data
        cl.SendData vbCr
    Case 2
        ' Invalid command - error with syntax

```



```

        If 2 > 0 Then
            cl.Log 2, "Error with syntax: " & CmdToSend
        End If
        Exit Do
    ' Case 3
    ' Take action based on the "Device specific mult" found.
    ' This may involve sending a command or keystrokes to the device.
    ' You may also need to flush the buffer before or after the above
    ' by using a cl.FlushRxBuffer command.
    ' Note: You may not want to exit the loop at this point because
    ' you will want to leave the device ready
    ' to receive another command so you should ensure you are at a
    ' device prompt when you exit the loop.
    Case 18, 19, 20
    ' Prompt found so the device has finished its output so exit the
    ' loop.
    local SendCommandSingle = True
    Exit Do
End Select
Loop
' Put the total output of the command in the RxBuffer so that it is
' available for use elsewhere.
cl.RxBuffer = StoredBuffer
End Function

Function ProcessBuffer()
' Do what ever processing of the data in cl.RXBuffer you need to here
ProcessBuffer = True
End Function

```

#### 4.3.7.4 How to: Enable/Disable ports on a Catalyst CatOS switch

The Cisco Catalyst CatOS switches (4000, 5000, 6000) use the 'Set' type commands to modify the configuration.

You can easily enable or disable specific ports on switches at certain times of the day via a scheduled [Device.CLI.Send commands](#) activity.

Create a scheduled activity to enable the ports in the morning.

Activity Time settings:

Custom Schedule: 07:00 Monday to Friday

Activity Option settings:

**[X] List of commands to be entered on device:**

**Set Port 3/1-24 enable** (enable ports 1 to 24 on slot 3)

**Set Port 5/1-12 enable** (enable ports 1 to 12 on slot 5)

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

Then, create another scheduled activity to disable the ports again during the evening.

Activity Time settings:

Custom Schedule: 19:00 Monday to Friday

Activity Option settings:

**[X] List of commands to be entered on device:**

**Set Port 3/1-24 disable** (disable ports 1 to 24 on slot 3)

**Set Port 5/1-12 disable** (disable ports 1 to 12 on slot 5)

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

#### 4.3.7.5 How to: Backup Cisco IOS image to TFTP server

Use the [Device.CLI.Send commands](#) activity to enter the following commands at the enable prompt.

Activity Option settings:

**[X] List of commands to be entered on device:**

**copy flash: tftp:**

**c2500-ik8s-l.122-5.bin** (name of image file to backup)

**192.168.1.200** (IP address of remote TFTP server)

**c2500-ik8s-l.122-5.bin** (name of image file to use on TFTP server)

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

#### 4.3.7.6 How to: Download Cisco IOS running config to TFTP server

Use the [Device.CLI.Send commands](#) activity to enter the following commands at the enable prompt.

Activity Option settings:

**[X] List of commands to be entered on device:**

```
copy running tftp
192.168.1.200 (IP address of remote TFTP server)
%ctGroupName/%ctDeviceName-Config.txt (filename to store config to on server)
```

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

Note - the use of the variables:

%ctGroupName = Device Group Name  
%ctDeviceName = Device Name

Note - the use of the forward slash in the filename to specify a subfolder within the TFTP Download folder

#### 4.3.7.7 How to: Download Cisco IOS running config to SCP server

Use the [Device.CLI.Send commands](#) activity to enter the following commands at the enable prompt.

Activity Option settings:

**[X] List of commands to be entered on device:**

```
copy running-config scp://Destination username:Destination
password@Destination HostIP/Destination filename
%ctCR
%ctCR
%ctCR
```

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

Example:

```
copy running-config scp://Admin:Password@10.190.1.10/%ctDeviceName-Config.txt
%ctCR
%ctCR
%ctCR
```

The device will normally respond with the prompts:

Address or name of remote host [**10.190.1.10**]?

Destination username [**Admin**]?

Destination filename [*Device-Config.txt* ]?

which by using the **%crCR** [Meta Variable](#) will tell CatTools to send a CR (Carriage Return) to answer each prompt

Note also the use of the meta variable:  
%ctDeviceName = Device Name

Also refer to [Backing up device via SFTP/SCP](#)

#### 4.3.7.8 How to: Upload Cisco IOS config from TFTP to device

To illustrate how to upload an IOS to a device using the [Device.CLI.Send commands](#) Activity please refer to the following example:

##### Activity Options tab settings:

[X] List of commands to be entered on device:

copy tftp running	
192.168.1.200	(IP address of remote TFTP server)
%ctGroupName\%ctDeviceName-Config.txt	(Filename to store config to on server)
%ctCR	(Required to start device TFTP)

[X] Enter commands in enable mode

[X] Save device output to file [Specify the log file name here]

[X] Overwrite existing capture file

[X] Answer yes to any confirmation prompts

[X] Stop on error

##### Please Note :

- 1) The use of [Meta Variables](#).  
 %ctGroupName = Device Group Name  
 %ctDeviceName = Device Name  
 %ctCR = Carriage Return
- 2) The use of the backward slash in the filename to specify a subfolder within the TFTP Upload folder.
- 3) The use of the %ctCR variable to send a <CR> to the device to activate the TFTP session.
- 4) The [Device.TFTP.Upload Config](#) Activity could also be used to upload an IOS to a device.

#### 4.3.7.9 How to: Upgrade Cisco IOS via TFTP

Use the [Device.CLI.Send commands](#) activity to enter the following commands at the enable prompt.

Activity Option settings:

**[X] List of commands to be entered on device:**

**erase flash:** (erase the existing flash files and make space for the new files)

**copy tftp: flash:**

**192.168.1.200**

(IP address of remote TFTP server)

**c2500-ik8s-l.1.122-5.bin**

(source filename)

**c2500-ik8s-l.1.122-5.bin**

(destination filename)

**[X] Enter commands in enable mode**

**[X] Save device output to file [Specify the log file name here]**

**[X] Overwrite existing capture file**

**[X] Answer yes to any confirmation prompts**

**[X] Stop on error**

#### 4.3.7.10 How to: Enter different commands onto each device

If you want to configure each of the selected devices in a different way, there are two ways to achieve this.

1. Create a separate [Device.CLI.Send commands](#) activity for each device and set the commands to be entered in the options tab.

This method involves having a separate scheduled activity for each device you want to modify.

2. Create a single [Device.CLI.Send commands](#) activity and select all the devices you want to modify. Then use the **Or, read commands from file** option and specify the %BaseFile% value as part of the file name. Then create a separate text file for each device that contains the commands you want to have entered into the config.

For example.

Specify a filename of C:\CatTools\%BaseFile%.txt

The list of devices might be...

Device name	Base filename	Config commands filename
CiscoRouter1	CiscoRouter1	C:\CatTools\CiscoRouter1.txt
CiscoRouter2	CiscoRouter2	C:\CatTools\CiscoRouter1.txt
Sales Switch	Sales_Switch	C:\CatTools\Sales_Switch.txt

\* Note that any spaces and invalid characters are replaced by and underscore (\_) in the base filename.

In each of the text files, enter a list of commands that you want to enter.

For example, in the file: C:\CatTools\CiscoRouter1.txt, enter the commands...

```
Interface s 0/0
Description New WAN link to head office
No shutdown
```

Enter different commands in each text file and then run the activity. CatTools will automatically match the correct text file with the device based on the %BaseFile% value.

#### 4.3.7.11 How to: Save output from multiple devices into a single file

Normally, the [Device.CLI.Send commands](#) activity will save the data from each device into a separate file.

The default capture filename is: ..\Captured Data\%GroupName%\%DeviceName%. DeviceOutput.%DateISO%.txt

To save the output from multiple devices into a single file, set the capture filename to something like: ..\Captured Data\MyActivityData.txt

On the options tab, ensure that you uncheck the **Overwrite existing capture file** option. This allows the data from each device to be appended to the common capture file.

On the Activity tab, set the **Client threads** to use a single thread only. This ensures that the output file is written to by only one device at a time.

To have start and finish tags placed between the output from each device, you will need to place some comments into the commands you send to the device.

For example, if the command you are entering is "Show ARP", you could use the following...

```
! %ctDeviceName - Start
Show ARP
! %ctDeviceName - Finish
```

When the command is sent to the device, the %ctDeviceName will be replaced with the current device name (SalesRouter-1 for example).

Since the first and last commands start with a !, they are ignored by Cisco routers.

For Catalyst switches (CatOS), use the # character instead.

The captured data file could look something like:

```
! Cisco2950Switch1 - Start
Protocol  Address          Age (min)  Hardware Addr  Type   Interface
Internet  192.168.1.1      3          0008.02b9.1111  ARPA   Vlan1
Internet  192.168.1.2      1          000d.9d89.2222  ARPA   Vlan1
! Cisco2950Switch1 - Finish
! Cisco2950Switch2 - Start
Protocol  Address          Age (min)  Hardware Addr  Type   Interface
Internet  192.168.1.3      6          0008.a118.3333  ARPA   Vlan1
Internet  192.168.1.4      0          0008.02cf.4444  ARPA   Vlan1
! Cisco2950Switch2 - Finish
```

The data captured from each device is wrapped with a Start and Finish tag.

### 4.3.8 Device.ConnectivityTest.Login

The **Device.ConnectivityTest.Login** activity allows you to Login to a selection of devices.

(It is similar to a [Device.ConnectivityTest.Ping](#) in that it does not change anything on the device but simply confirms connectivity plus the ability to Login).

CatTools will Login to each selected Device in turn, and will log any failures to the Report file:  
...\Reports\Device.ConnectivityTest.Login.txt

In general, the reasons for performing Login tests are similar to those for performing Ping tests, with some differences.

The Login test goes a little bit further than a simple Ping, confirming that interactive terminal access to the device is working, as well as confirming the network operation between you and the device, in the process. It also confirms that your passwords have not changed. You may want to check that your staff are still able to login to the selected device, or you may want to check that the login passwords have not been altered without your knowledge or authorization, or both.

The report file provides you with an historical record of the status of the tested devices, which may be useful for later forensic purposes.

For more information see [Setting up the activity](#)

#### 4.3.8.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Device.ConnectivityTest.Login](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time tab:**

Select, for instance, "Every 10 minutes" as the Recurring option in the Times box to run the activity every 10 minutes.  
Leave all other options on this tab at their defaults.

**Devices tab:**

Select the Devices you want to test for Login.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Enter enable mode if possible:** tick box, if the Device has an Enable mode and the appropriate passwords are configured in the Device file, then CatTools will attempt to enter enable mode each time it logs in to each Device.

**Only record login failures:** tick box, keeps a record only of failures of the Activity. This keeps the Report file to a minimum size, conserving hard-drive space.

#### 4.3.9 Device.ConnectivityTest.Ping

The **Device.ConnectivityTest.Ping** activity allows you to Ping to a selection of devices.

CatTools will Ping each selected Device in turn, and will log any failures to the Report file:  
...\Reports\Device.ConnectivityTest.Ping.txt

The Ping test is a simple ICMP Echo request, sent from the machine running CatTools to the selected device(s).

It tests the operation of your network between CatTools and the selected device, but only at the IP level. It may or may not be a useful test of your network, or your users' ability to use the network, depending on your specific network design.

The report file provides you with an historical record of the status of the tested devices, which may be useful for later forensic purposes.

For more information see [Setting up the activity](#)

#### 4.3.9.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Device.ConnectivityTest.Ping](#) as the Activity Type: Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "Every 15 minutes" as the Recurring option in the Times box to run the activity every 15 minutes.

Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to Ping test from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Response timeout (seconds):** is the time in seconds that the activity waits until it assumes that the Ping request has failed. The default is 2 seconds.

**Ping packet send count:** is the number of Ping requests the activity sends to each Device each time the Activity runs. The default is 5 packets.

**Only record ping failures:** keeps a record only of failures of the activity. This keeps the Report file to a minimum size, conserving hard-drive space.

**Log Error if 100% not reported:** raises an error if all pings are not 100% successful.

#### 4.3.10 Device.InterDevice.Ping

The **Device.InterDevice.Ping** activity allows you to Ping a list of Hosts from the selected Devices.

CatTools will log into each device in turn and perform a ping test to each device listed. Alternatively, it will ping all other devices that have been selected.

There are two options available for which devices are pinged.

1. You may enter a list of Host names and IP Addresses in the "Ping a list of Hosts" field, under the Options tab. CatTools will login to each selected device and ping each listed Host in turn.

2. You may check the "Or, ping all other selected devices" box (also under the Options tab). CatTools will then start with the first selected Device and ping all the others in turn, move to the second Device and ping all others in turn, and so on until all selected Devices have sent Ping requests to all other selected Devices. This test can ensure that you have full connectivity from each device to all other devices.



CatTools will then create a Report in the form of a table listing each Device and the results of each Ping originated by that Device.

Under the Options tab you can also check the option to have CatTools report only on failed Ping requests. This will keep the Report as small as possible and draw attention only to fault conditions on your network.

Note:

The Hostname, IP Address information can be entered in two formats:

- If you don't know the Hostname then simply enter the IP Address instead. Each IP Address must be entered on a new line (marked by a Carriage Return or Enter).
- If you know the Hostname then enter both Hostname and IP Address, using a comma and space to separate them, with each Hostname and IP Address again listed on a new line.

The Hostname is used in the Report, in its own column, which simply makes it easier to read.

For example:

Main Server, 192.168.1.1  
Sales router, 192.168.2.1  
ISP DNS, 202.30.45.21

For more information see [Setting up the activity](#)

#### 4.3.10.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Device.InterDevice.Ping](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "Every minute" as the Recurring option in the Times box to run the activity every minute.  
Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to perform ping testing from.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

Either, enter a list of Host names and IP Addresses in the **Ping a list of hosts (Display name, IP address)** text box or simply check the **Or, ping all other selected devices** option (checking this option will disable the Host names, IP Addresses field – the two are mutually exclusive).

**Only record ping failures:** tick box, will only log ping failures.

#### 4.3.10.2 When to use the Device.InterDevice.Ping

The [Device.InterDevice.Ping](#) activity is a useful network management tool because it allows you to perform and report on multi-directional testing across your whole network.

In many router networks, pinging the routers is not a sufficient test to provide a high level of confidence in the proper working of the network, because the router is closer to you than the devices that actually need to use the network.

It is reasonably common, for example, for pings to remote routers to use the WAN interface IP Address. So as long as the WAN and Router are functional the ping request will succeed, but this does not tell you that the LAN on the far side of the Router is operating as you intend. It also only tests your ability to Ping the Router from a particular point in your network: it does not test from other points, nor does it test the ability of Devices on the far side to successfully use your network.

If you have a Linux Device on each remote LAN segment (or at least "behind" each remote router) you can access each of those Devices from CatTools in turn, and use it to Ping all other significant parts of your network. This provides better evidence that the network is operating properly than simply pinging the routers WAN interface from a single central point.

Also, by default the Activity appends reporting information to the same file:  
...\\Reports\\Device.InterDevice.Ping.txt

This file could be very useful for providing forensic data in the event of a service provider outage or other event that affects your network.

If you Schedule this Activity on a regular basis, say every 5 minutes (unless you have a quite large number of Devices in your network this would not be a significant loading), then you should be able to use the time-stamped record of ping failures to connect the two events. Although that might not prove the source of the failure, it should provide helpful evidence.

#### Checking availability of a central device

This test will allow you to test branch office connectivity back to the central office server. Simply select all the branch office routers (or a Linux machine sitting in the branch office) and have them ping the server back in central office. This will ensure that all of your customers can access the central server.

### 4.3.11 Device.TFTP.Upload Config

The **Device.TFTP.Upload Config** activity enables you to upload text config files to the specified device.

Please note the current [Devices supported](#) by this activity.

For more information see [Setting up the activity](#)

#### 4.3.11.1 Setting up the activity

##### **Activity** tab:

Create a new Activity and select [Device.TFTP.Upload Config](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

##### **Time** tab:

Select a time to run the activity.  
Leave all other options on this tab at their defaults.

##### **Devices** tab:

Select the Devices you want to report on from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**TFTP Server:** set the address of the TFTP server

**TFTP FileName:** set the name of the file containing the config data or use the default

**Save device output to file:** tick box lets you save the output from the session with the device to a file

**Overwrite existing capture file:** tick box lets you overwrite or append to the capture file

**Save to NVRAM:** tick box allows you to save the new config to NVRAM for non-CatOS devices

#### 4.3.11.2 Devices supported

To see a full list of the currently supported device types for the [Device.TFTP.Upload Config](#) activity, please see the **Activities** tab of the [device matrix](#) on our [web site](#).

#### 4.3.12 Device.Update.Banner

The **Device.Update.Banner** allows you to easily apply a banner to your devices.

Please note the current [Devices supported](#) by this activity.

CatTools will pass the banner command to each selected Device in turn. Success or Failure is logged to the Report file:

...\Reports\Device.Update.Banner.txt

This activity supports the use of [Command Variables](#)

The text provided either through the on screen text box or through an input file should be plain text. No delimiters or syntax command words are needed.

For more information see [Setting up the activity](#)

#### 4.3.12.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Device.Update.Banner](#) as the Activity Type:

**Time** tab:

If you wish to run the Activity at a scheduled time set it here.

**Devices** tab:

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Banner text:** tick box, indicates if the Banner text is to be read from the text box or from a file. Ticking the checkbox allows for free text to be entered. While unticking it allows you to select a text file which will contain the Banner text to be applied.

**Which banner is to be supported:** Select which banner to apply. Note that not all banners are supported on all devices.

**Save device output to file:** tick box lets you save the Banner command that was used to the file specified.

**Overwrite existing capture file:** tick box lets you overwrite or append to the capture file

**Save running config to start-up config when complete:** Is the Banner to be saved to NVRAM? Note that some devices do not support this option.

#### 4.3.12.2 Devices supported

To see a full list of the currently supported device types for the [Device.Update.Banner](#) activity, please see the **Activities** tab of the [device matrix](#) on our [web site](#).

#### 4.3.13 Device.Update.Password

The **Device.Update.Password** activity enables you to change some of the passwords on devices.

Please note the current [Devices supported](#) by this activity.

You can change the Enable mode password, the Enable Secret password, the VTY password, and the Console password.

The options enable you to select which of the passwords you want to change, and to enter a new password for it. You can also enter a blank password in the field for either of the Enable mode passwords. This enables you to reset the password to nothing.

NOTE: This is not an allowed option for the VTY and Console passwords.

A report gives you the status of each device selected for the activity after the activity is completed. The report shows both the new and old passwords. These passwords are shown by default in plain text on the report. You may elect to hide the reported passwords by selecting the [Hide password change report passwords](#) option in the Misc tab in the CatTools Setup dialogue.

For more information see [Setting up the activity](#)

##### 4.3.13.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Device.Update.Password](#) as the Activity Type:

**Time tab:**

If you wish to run the Activity at a scheduled time set it here.

**Devices tab:**

Select the devices that you want this activity applied to by checking the tick box next to each device in the grid.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**New enable:** tick box, to enable New enable password field.

**New enable secret:** tick box, to enable New enable secret field.

**Choose password for CatTools:** drop down list, choose to select which of the Enable mode passwords you want CatTools to remember and use for the devices.

**New VTY:** tick box, to enable New VTY password field.

**Console password:** tick box, to enable Console password field.

**Save running config to start-up config when complete:** tick box to copy the running config on the device to non-volatile memory, so that the modified config becomes the start-up config.

#### 4.3.13.2 Devices supported

To see a full list of the currently supported device types for the [Device.Update.Password](#) activity, please see the **Activities** tab of the [device matrix](#) on our [web site](#).

#### 4.3.14 Report.ARP table

The **Report.ARP table** activity builds a report using the ARP table from selected devices as source data.

On a Cisco IOS device it uses a simple "Show IP ARP" command and builds a table of results over time.

The data is stored in a text file whose default location is:

...\Reports\Master ARP table.txt

It automatically indexes MAC addresses against IP addresses and device interfaces, and then resolves their host names via DNS if required. By default the table is updated with each run of the activity, so it provides a historical record of the devices attached to your network over time. Each entry is time stamped, and "First Seen" and "Last Seen" columns included in the report.

By importing the file into a spreadsheet program like Excel, you can search for specific MAC addresses or host names, or maybe list all devices on a particular port.

The Options tab allows you to use an alternate command, to set the expiry date of MAC entries in the report, and to select whether to resolve IP addresses.

For more information see [Setting up the activity](#)

##### 4.3.14.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.ARP table](#) as the Activity Type:

Leave the other options on this tab at their defaults.

**Time tab:**

Select, for instance, "Every hour" as the Recurring option in the Times box to run the activity every hour.

Leave all other options on this tab at their defaults.

**Devices tab:**

Select the Devices you want to report on from your CatTools Host.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Use alternate command:** tick box activates command entry field, to allow you to set an alternate command if the default command issued by CatTools does not produce the correct results. This may be useful if you have some Devices not yet supported in the CatTools default device list. The field is not checked by default.

**Days until entries expire:** tick box activates associated numeric field (whos default is 30 days) . Here you may enter the number of days you want entries to remain in the table. Every time this activity is run, the existing MAC address table entries that are older than the expire time will be removed. This keeps the table to a workable size.

**Resolve IP addresses to host names:** tick box activates resolution of host names to IP addresses for report. Each time the activity runs, the program will scan the table for entries that have a blank host name field. A list of IP addresses will be created and then resolved in bulk. The resolver will attempt up to 100 DNS lookups at once. If a hostname is returned for the IP address, the table will be updated.

### 4.3.15 Report.CDP Neighbors table

The **Report.CDP.Neighbors table** activity produces a report of all networked devices that can be seen by the specified device using the CDP neighbor command.

Please note the current [Devices supported](#) by this activity.

For more information see [Setting up the activity](#)

#### 4.3.15.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Report.CDP Neighbors table](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time tab:**

Select a time to run the activity.

Leave all other options on this tab at their defaults.

**Devices tab:**

Select the Devices you want to report on from your CatTools Host.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Use alternate command:** tick box activates command entry field, and allows you to enter

alternate commands if the default command issued by CatTools does not produce the correct results. This may be useful if you have some Devices not yet supported in the CatTools default device list. The field is not checked by default.

**Hide duplicate IP address entries:** tick box allows you to hide duplicate IP address entries.

#### 4.3.15.2 Devices supported

To see a full list of the currently supported device types for the [Report.CDP Neighbors table](#) activity, please see the **Reports** tab of the [device matrix](#) on our [web site](#).

#### 4.3.16 Report.Compare.Running Startup

The **Report.Compare.Running Startup** activity compares the running and the startup configs of your device(s) and reports on the differences found.

##### **How it works**

The activity works by first connecting to your selected device(s) then issuing the commands which will show the running and startup configs.

These are saved into the ClientTemp folder while the activity is running.

When the configs have been retrieved from the device(s) a compare is run against the two captures and a report is generated. The compare process is very similar to the [Report.Compare.Two files](#) activity.

##### **Report Generation**

The reporting of any differences found can be any combination of the following three reports.

- 1) As with other Activities the Report File on the main Activity tab records a history of the jobs that have been run and their result.
- 2) An HTML report can be created which will allow you to see the differences between the two files next to each other in column format and is colour coded.
- 3) A txt file report can be generated which shows differences in top down format.

The two file paths are capable of receiving [Filename Variable](#) parameters.

**Note** Not all devices currently support this activity. See the [Devices supported](#) list for more information.

For more information see [Setting up the activity](#)

#### 4.3.16.1 Setting up the activity

##### **Activity** tab:

Create a new Activity and select [Report.Compare.Running Startup](#) as the Activity Type:

##### **Time** tab:

If you wish to run the Activity at a scheduled time set it here.

##### **Devices** tab:

Select the devices you wish to run this activity against by ticking the check box against each in the list. Please note that not all [devices support](#) this activity.

##### **E-mail** tab:

Set preferences for reporting on this Activity.

##### **Options** tab:

**Alternate Startup command:** tick box, enables text field to enter a command which when issued to your device will return the startup config.

**Alternate Running command:** tick box, enables text field to enter a command which when issued to your device will return the running config.

**Ignore Text:** Should the new file contain any lines of text that are likely to different from the original file every time the comparison is run (such as a timestamp) and you would like to ignore these lines, they can be added here. More information is available on the [Ignore Text](#) field.

**Note:** to ignore "certificate self-signed" changes on Cisco routers, see the [block text ignore example](#).

**HTML compare report:** Specifies the folder and file name of where to store the HTML diff report. This diff report file contains a side by side comparison of the Running and Startup configs. Changes are highlighted in different colors.

**Text summary diff report:**

Specifies the folder and file name of where to store the text based diff report. This diff report file contains a list of the changes found in the Running and Startup configs. This report is a smaller and simpler way to view the changes. Unlike the HTML report, where both the Running and Startup configs are shown in full. The text report only shows the lines that are different.

**Only notify by e-mail if differences found:** Check this option to receive emails reporting any difference found between the startup and running configs.

**Attach reports to email:** The email you receive is only notification that differences were found. If you would like to see what those differences are then choose which reports you would like attached.

**Zip Attachments:** The name of the zip file into which the reports will be added. Path is not necessary as this zip file is emailed not saved.

**Password protect Zip file:** A password to encrypt the Zip file if the Zip attachments option has been selected.

#### 4.3.16.2 Devices supported

To see a full list of the currently supported device types for the [Report.Compare.Running Startup](#) activity, please see the **Reports** tab of the [device matrix](#) on our [web site](#).

#### 4.3.17 Report.Compare.Two files

The **Report.Compare.Two files** activity runs a compare against two files which you define and reports on the differences it finds.

**Report Generation**

The reporting of any differences found can be any combination of the following three reports.

- 1) As with other Activities the Report File on the main Activity tab records a history of the jobs that have been run and their result.
- 2) An HTML report can be created which will allow you to see the differences between the two files next to each other in column format and is colour coded.



3) A txt file report can be generated which shows differences in top down format.

The two file paths are capable of receiving [Filename Variable](#) parameters.

**Note**

This activity does not require the selection of any devices. Therefore, Filename variables relating to devices such as %GroupName%, %BaseFile% , %DeviceName% will not work.

For more information see [Setting up the activity](#)

#### 4.3.17.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Report.Compare.Two files](#) as the Activity Type:

**Time tab:**

If you wish to run the Activity at a scheduled time set it here.

**Devices tab:**

This type of Activity does not run against any devices.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Source File:** The first file for the comparison to use. This can be thought of as the *new file*.

**File to compare against:** The second file for the comparison to use. This can be thought of as the *original file*.

**Ignore Text:** Should the new file contain any lines of text that are likely to different from the original file every time the comparison is run (such as a timestamp) and you would like to ignore these lines, they can be added here. More information is available on the [Ignore Text field](#).

**HTML compare report:** If you would like the results shown in HTML format then indicate here where the report should be saved.

**Text summary diff report:** If you would like the results shown in Text format then indicate here where the report should be saved.

**Only notify by e-mail if differences found:** Check this option to receive emails reporting any difference found between the new and old files.

**Attach reports to email:** The email you receive is only notification that differences were found. If you would like to see what those differences are then choose which reports you would like attached.

**Zip Attachments:** The name of the zip file into which the reports will be added. Path is not necessary as this zip file is emailed not saved.

**Password protect Zip file:** A password to encrypt the Zip file if the Zip attachments option has been selected.

### 4.3.18 Report.Error info table

The **Report.Error info table** activity collects and reports error counter information for all interfaces of each selected device.

Please note the current [Devices supported](#) by this activity.

The default location for the report is: ...\\Reports\\Master Error info table.txt

This activity contains four options:

The first option allows you to specify the order of the report headers, decide which columns you wish to display and also what names you want the columns headers to be if you don't want to use the defaults.

The second option will calculate column totals. Providing you with a summary line for each of the devices in the activity.

The third option allows you to clear the interface counters after collecting the data.

And finally, if enabled, the fourth option will allow you to collect the error counters from your VLANs. This last option is only supported by Cisco IOS devices.

For more information see [Setting up the activity](#)

#### 4.3.18.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.Error info table](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "Every 4 hours" as the Recurring option in the Times box to run the activity every 4 hours.  
Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to report on from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Specify Report Headers:** table allows you to specify the order of the report headers, decide which columns you wish to display and what names you want the column headers to be. The first row of this matrix shows all the information that may be reported on. You can select to report on each column or not as you wish. You may also select and drag each column to the position you want for formatting the report layout.

**Calculate column totals:** tick box to calculate column totals. This provides you with a summary line for each of the errors columns. Each device in the activity has a separate totals summary line.

**Clear the interface counters:** tick box allows you to clear the devices error counters after collecting all the current data.

**Collect VLAN information:** tick box allows you to collect the error counters from your VLANs. This last option is only supported by Cisco IOS devices.

#### 4.3.18.2 Devices supported

To see a full list of the currently supported device types for the [Report.Error info table](#) activity, please see the **Reports** tab of the [device matrix](#) on our [web site](#).

#### 4.3.19 Report.MAC address table

The **Report.MAC address table** activity gathers MAC address entries (CAM table entries) from switches and bridging routers and can builds a table over time.

The data is stored in a text file whose default location is:  
...\Reports\Master MAC address table.txt

The Options tab allows you to use an alternate command and to set the expiry date of MAC entries in the report.

This activity by default may issue more than one command depending on the device type. In these cases, not every command issued may be valid for every device selected. This does not cause problems or errors as CatTools gathers information from every command that succeeds, and just continues after any command that does not produce a valid result.

Entering an alternate command to use overrides all default commands.

For more information see [Setting up the activity](#)

##### 4.3.19.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [Report.MAC address table](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time tab:**

Select, for instance, "Every hour" as the Recurring option in the Times box to run the activity every hour.  
Leave all other options on this tab at their defaults.

**Devices tab:**

Select the Devices you want to report on from your CatTools Host.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Use alternate command:** tick box activates command entry field, to allow you to set an alternate command if the default command issued by CatTools does not produce the correct results. This may be useful if you have some Devices not yet supported in the CatTools default device list. The field is not checked by default.

**Days until entries expire:** tick box activates associated numeric field (whos default is 30 days) . Here you may enter the number of days you want entries to remain in the table. Every time this activity is run, the existing MAC address table entries that are older than the expire time will be removed. This keeps the table to a workable size.

### 4.3.20 Report.Port info table

The **Report.Port info table** activity creates a snapshot of the interface configuration and state of the selected devices and stores the data in a tab delimited report file.

The data is stored in a text file whose default location is:  
...\Reports\Master Port info table.txt

The information gathered includes name, VLAN, port type (Ethernet, Fast Ethernet, Full Duplex, etc), status, and speed.

An option in the Option tab allows you to enter an alternate command should the default command not produce the correct output.

For more information see [Setting up the activity](#)

#### 4.3.20.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.Port info table](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "At Midday" as the Recurring option in the Times box to run the activity every day at noon.

Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to report on from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Use alternate command:** tick box activates command entry field, to allow you to set an alternate command if the default command issued by CatTools does not produce the correct results. This may be useful if you have some Devices not yet supported in the CatTools default device list. The field is not checked by default.

### 4.3.21 Report.SNMP.System summary

The **Report.SNMP.System summary** activity produces a summary report of data available from any device that supports standard system SNMP.

All columns are optional and may be renamed. You can add up to 5 optional columns to the report and enter your own OID to produce the column data.

Please note the current [Devices supported](#) by this activity.

For more information see [Setting up the activity](#)

#### 4.3.21.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.SNMP.System summary](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select a time to run the activity.  
Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to gather SNMP information from.

**E-mail** tab:

Set preferences for reporting on this Activity.

**Options** tab:

**Specify Report Headers:** table allows you to specify the order of the report headers, decide which columns you wish to display and what names you want the column headers to be .

**Show Uptime in Ticks:** tick box allows you to show the device up time in ticks rather than days\hours\minutes etc.

**Collect Custom SNMP Date:** 5 tick boxes and entry boxes to set your own OID to collect data with. Tick the corresponding report header to show it in the report.

#### 4.3.21.2 Devices supported

The [Report.SNMP.System summary](#) activity should be able to be run against any device that provides SNMP MIB-2 system information.

It should be able to be run against devices not otherwise supported by CatTools. You can add a new device and call it any device type as CatTools does not use device specific logic for this activity. The only details CatTools uses are the IP address to find the device and the SNMP read community string to access the device.

#### 4.3.22 Report.Version table

The **Report.Version table** activity creates a tab delimited report of all the software and hardware version information from the selected devices.

The data is stored in a text file whose default location is:  
...\Reports\Master Version table.txt

The Version table contains the following information columns:

- Group
- Device Name
- Serial Number
- Processor
- IOS (version number)
- ROM (revision)
- Boot
- Uptime
- Flash
- NVRAM (size)
- Memory

- Image (filename)

When the report is viewed in the Report tab, the columns are sortable by clicking on the column name. Clicking again will reverse the listed order.

For more information see [Setting up the activity](#)

#### 4.3.22.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.Version table](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, "At Midnight" as the Recurring option in the Times box to run the activity every day at midnight.  
Leave all other options on this tab at their defaults.

**Devices** tab:

Select the Devices you want to report on from your CatTools Host.

**E-mail** tab:

Set preferences for reporting on this Activity.

#### 4.3.23 Report.X-Ref.Port MAC ARP

The **Report.X-Ref.Port MAC ARP** activity creates a cross reference report of MAC and IP addresses against ports on a network. If IP addresses have been resolved to host names on the ARP report, the host names are reported as well.

This report uses the [Device.Port info table](#), [Report.MAC address table](#) and [Report.ARP table](#) reports as its information base. It collects all port information from the Port report and matches MAC and ARP information against the ports.

These report activities **must** be run before the cross reference report can be run.

This report **does not require any devices** to be selected as it uses the devices selected for the port report as its starting point.

Options allow you to select the source reports, limit the number of MAC addresses displayed, and ignore specified interfaces.

For more information see [Setting up the activity](#)

#### 4.3.23.1 Setting up the activity

**Activity** tab:

Create a new Activity and select [Report.X-Ref.Port MAC ARP](#) as the Activity Type:  
Leave the other options on this tab at their defaults.

**Time** tab:

Select, for instance, the Custom schedule option in the Times box, set the time to 14:00, and select Monday only from the Days box. This will run the activity at 2pm every Monday to create the cross reference report.  
Leave all other options on this tab at their defaults.

**Devices tab:**

There is no need to select any devices for this activity as it uses existing reports for its data.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options tab:**

**Current Port info table:** the path of the port info table to use for the x-ref

**Current MAC address table:** the path of the MAC address table to use for the x-ref

**Current ARP table:** the path of the ARP table to use for the x-ref

**Max MAC entries per port:** enables a limit of the number of MAC entries to be displayed in the report on a port - useful for trunks and similar ports

If you select 5 entries here, if there are more than 5 MAC entries the 5 latest MAC entries only are displayed.

**Interfaces to exclude:** for example, these may be trunk ports that link devices and that may have a large number of MAC entries linked to them.

To exclude an interface you can enter an exact port name, or you can enter a regular expression that may evaluate to exclude a number of similarly named ports. An example of this might be "Giga|Serial" to remove any ports with the text "Giga" or "Serial" in their name.

#### 4.3.24 System.File.Delete

The **System.File.Delete** activity allows you to automate the deletion of files. It is primarily designed to allow the management of the dated config files, although it can be run against any folder.

It does not run against your physical network devices as most of the other Activities do. Instead, from the specified folder and optionally sub folders, it will;

Delete all files except a specified number of device specific files.  
Delete files older than a certain number of days.

Or a combination of the above two options.

For more information see [Setting up the activity](#)

##### 4.3.24.1 Setting up the activity

**Activity tab:**

Create a new Activity and select [System.File.Delete](#) as the Activity Type:

**Time tab:**

If you wish to run the Activity at a scheduled time set it here.

**Devices tab:**

If you choose to retain a certain number of files for each device (see options below) then you must specify which devices are to be included. In most cases you will want to select all.

**E-mail tab:**

Set preferences for reporting on this Activity.

**Options** tab:

**Select Folder:**

Select a file from the folder which you want to run the activity against.

**Include sub folders:**

Select this if you want the activity to run against all sub folders of the selected folder.

**Number of device specific dated files to retain:**

You may want to clean the dated configs but ensure that you always keep the last 'x' number of dated configs'. This option allows you to do this. Enter the number of files of each device specific type you want to keep.

This works by using the CatTools Device Filename (for devices specified in the devices tab above) and comparing this to the file names in the folder to identify files that relate to the same device. It will then keep 'x' number of files for each device type.

Note: see [Troubleshooting](#) if your System.File.Delete activity is not retaining 'x' number of files.

**Delete files older than how many days:**

Select this option to delete any files older than the specified number of days. This option is of lower precedence than the above option so that if both are specified you will still keep 'x' number of files even if they are older than the specified number of days.

#### 4.3.24.2 Troubleshooting

**1) 'Number of device specific dated files to retain' option not working:**

If you have set the 'Number of device specific dated files to retain' option but the activity is deleting all files, then it is most likely due to the file naming convention you have used (over the default CatTools file naming convention).

CatTools default naming convention will use for example the naming convention (for the backup activity):

```
Config.Current.Running.Cisco_Router_2.txt
```

However, if you are using a convention which does not place a period mark (.) at the end of the device name (i.e. Cisco\_Router\_2. ) then CatTools may not be able to group the files relating to the device correctly.

For example, if you are using a file naming convention such as:

```
Cisco_Router_2-backup.txt
```

CatTools will not be able to group the Cisco\_Router\_2 files together correctly and so all files will be deleted.

The solution therefore is to change the '-' in your own naming convention to a '.' and this should fix the problem.

For example:

```
Cisco_Router_2-backup.txt
```

should become

```
Cisco_Router_2.backup.txt
```



## 4.4 Internal functions

This section contains CatTools internal functions and methods that can be used to help you customise your Activity.

### 4.4.1 Ignore Text

When comparing files using either the [Report.Compare.Two files](#) or [Report.Compare.Running Startup](#) activities you are provided with the option to supply ignore text.

Lines that match the ignore text instruction will be classed as ignored lines. Ignored lines are still highlighted in the diff report but they are not flagged as changes / differences and so will not trigger alert emails to be sent.

The Ignore text syntax is made up of two parts.

1. The first part is an instruction which defines the context of the ignore text.
  - ^ is the instruction to ignore lines with the sample text occurring anywhere in them
  - < is the instruction to ignore lines that begin with the sample text
  - > is the instruction to ignore lines that end with the sample text
2. The second part is a block of sample text which will be searched for when the line comparison is being made.

**The following is an example of how this feature might be used:**

^Time	Will ignore any lines with the text "Time" in them
<Time	Will ignore any lines which begin with the word "Time"
>Time	Will ignore any lines which end with the word "Time"

#### **Examples:**

^! Last configuration change at  
This will ignore only lines that contain the text "Last configuration change at"

<Current configuration  
This will ignore only lines that begin with "Current configuration"

^! Last configuration change at|<Current configuration  
This will ignore lines that contain the text "Last configuration change at" as well as lines that begin with "Current configuration"

Multiple ignore text instructions may be concatenated by using the | (pipe) symbol.

E.g.

^some text|^some other text|<some more text

#### **Ignoring changes that span multiple lines:**

To have a block of changes ignored, you can use the following ignore syntax...

{start of block text match}-{end of block text match}

In our example below, the running-config contains a code block that is missing from the startup config.

#### ***Cisco Running-Config:***

!

```
crypto ca certificate chain TP-self-signed-12345678
certificate self-signed 01
3082022B 30820194 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D31 37363538
528BD5A8 E7E26C51 10BAB609 5B60228F C8DE0299 7BE85C2D 9769FF05 C295706F
3082022B 30820194 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D31 37363538
528BD5A8 E7E26C51 10BAB609 5B60228F C8DE0299 7BE85C2D 9769FF05 C295706F
quit
Username joe password bloggs
!
```

### **Cisco Startup-Config:**

```
!
crypto ca certificate chain TP-self-signed-12345678
certificate self-signed 01 nvram:IOS-Self-Sig#4567.cer
Username joe password bloggs
!
```

To ignore this entire block of changes, you could use the following ignore text:

```
{certificate self-signed 01$}-{quit}|^certificate self-signed
```

This matches "certificate self-signed 01" at the top and ignores all changes until it matches the "quit" at the end of the block.

The text contained between the {}-{} uses the Regular Expression syntax. So in our example, the \$ means match the end of the line.

#### **Note:**

Because the ignore text field uses | (pipe) as a delimiter between multiple inputs, you can not use the | (pipe) in the regular expression syntax.

E.g.

```
^some text|{crypto block start}-{crypto block end}|<some more text
```

## **4.4.2 Filename Variables**

There are a number of variables that can be used within the filename fields throughout CatTools which will help you create unique names for your backups and reports.

These variables are resolved to actual values at the time the activity is started.

Example:

C:\CatTools V3\Captured Data\%GroupName%\%DeviceName%.DeviceOutput.%DateISO%.txt

Filename variables should not be confused with [Meta Data](#) variables.

Variable	Value
%GroupName%	Device group name as a folder name
%BaseFile%	Device specific file name
%HostAddress%	Device specific Host Address
%DateISO%	ISO format date
%DateYYYY%	Date as yyyy (year only)

%DateMM%	Date as mm (month only)
%DateDD%	Date as dd (day only)
%DateWW%	Day of the week (values 1 through 7)
%DateWDN%	Week Day Name (e.g. Monday)
%TimeHHMM%	Time as hhmm (e.g. time of 13:01 will display as 1301)
%TimeHH%	Time as hh (e.g. time of 13:01 will display as 13)
%TimeMM%	Time as mm (e.g. time of 13:01 will display as 01)
%DeviceName%	File system friendly device name
%AppPath%	File system path of the CatTools executable including a trailing "\"

It should be noted that %GroupName% and %DeviceName% are device specific variables. As such they can not be resolved against an Activity (summary report file) since an Activity is related to multiple devices.

### 4.4.3 Meta Data

Meta data is a term used to describe the method by which a user can access internal CatTools variables or functions from within an Activity.

Meta data should not be confused with [Filename Variables](#).

The meta data commands / variables are placed within the stream of commands defined by the user for the Activity.

They may be the only command for an Activity but would normally be interspersed amongst actual device commands.

The commands are evaluated by CatTools for each device in the Activity list and tell CatTools to either :

- 1) Perform a CatTools internal action at that point in time. A [Meta Command](#).
- 2) Or replace the variable with an actual data value. A [Meta Variable](#).

Activities that support the use of Meta data include :

[Device.CLI.Send commands](#)  
[Device.CLI.Modify Config](#)  
[Device.Update.Banner](#)

#### 4.4.3.1 Meta Commands

%Meta Commands are a subset of CatTools [Meta Data](#).

They are used to tell CatTools to perform a piece of internal functionality at a particular point in an Activity.

Below is a list of the currently available Meta Commands.

Meta Command	Description
<a href="#">%ctDB</a>	Updates the CatTools Device table which holds the properties for each device.
%ctUM: EchoOff	See the <a href="#">Device Table</a> section for a list of the device fields you can update. Sets a flag so commands are sent without waiting for an echo back of the command.
%ctUM: EchoOn	Sets a flag so commands are sent and wait for an echo back of the command.
%ctUM: Timeout	Sets the maximum time to wait for the response to a command.
%ctUM: PauseTime	Sets a period to wait before sending a command.

## 4.4.3.1.1 %ctDB Command

The %ctDB meta command enables you to directly update a field in a table in the CatTools database as part of an Activity.

**Syntax**

```
%ctDB:tablename:fieldname:new field value
```

The tablename is the name of the table in the database you wish to update a field in.

The fieldname is the name of the field you wish to update. See the [Device Table](#) section for a list of fields.

The new field value is the content you wish to place into the database field. This value may contain a colon - : - in the content as the command parser only looks for the first 3 colons as separators.

For instance, you may wish to change the password for devices on the remote authentication server. This would be done on the server itself, and CatTools would not be aware that the password has changed. You can set up an [Device.CLI.Send commands](#) activity with a meta command that will change the password in the CatTools database for the selected devices. The command would look like:

```
%ctDB:Device:AAAPassword:thenewpassword
```

In this case, you need to enter only the one meta command as the commands for the [Device.CLI.Send commands](#) activity. When the activity runs, it sets the AAAPassword field of the Device table to "thenewpassword" for all the selected devices. CatTools would then use this new password whenever it needed to login to one of the devices.

You might also use this type of command if you were setting up the local username and password properties on a set of devices. You would place the meta commands to update the CatTools database fields after the device commands that set the properties on the devices. It is recommended that you select the option to Stop on error in this situation to ensure that CatTools does not have a field set if the device command does not complete successfully.

In a [Device.CLI.Modify Config](#) activity you might enter the command

```
username joe password fred ...
```

to set a local username and password. You could use the meta commands to update the CatTools database to reflect these changes:

```
%ctDB:Device:AAAUsername:joe
%ctDB:Device:AAAPassword:fred
```

CatTools then knows to use these values when logging into the device in future.

## 4.4.3.1.1.1 Device Table

This is a list of the fields you may update in the Device table using the [%ctDB](#) meta command:

Field Name	Max Size - characters
HostAddress	255
Filename	255
Model	255
Telnet	255
TelnetPort	255
Session	255

VTYPass	255
ConsolePass	255
EnablePass	255
PrivilegeLevel	255
AAAUsername	255
AAAPassword	255
SNMPRead	255
SNMPWrite	255
RequireVTYLogin	255
LoginUsesAAA	255
EnableUsesAAA	255
VTYPrompt	255
ConsolePrompt	255
EnablePrompt	255
AAAUserPrompt	255
AAAPassPrompt	255
Address1	>255
Address2	>255
Address3	>255
ContactName	>255
ContactPhone	>255
ContactEmail	>255
ContactOther	>255
AlertEmail	>255

While all the fields are character format fields, some such as RequireVTYLogin, LoginUsesAAA, EnableUsesAAA contain numbers to indicate whether they are on or off. 0 is off, 1 is on.

#### 4.4.3.1.2 %ctUM

The %ctUM meta commands enable you to set various options for use in the Device.CLI.Send commands and Device.CLI.Modify Config activities.

##### **Commands**

##### **%ctUM: EchoOff**

Normally when commands are sent out, CatTools will wait for an echo of that command before proceeding. Some commands, however, will not produce an echo and so the activity will fail at this point. (Password changing activities are notorious for this.)

The %ctUM: EchoOff command will tell CatTools to send out any further commands in this activity without waiting for an echo.

##### **%ctUM: EchoOn**

Although in some cases it is necessary to switch off the echoing of commands it is advisable to switch echoing back on as soon as possible. By waiting for an echo of commands CatTools can better synchronise data sent and received from the device minimising the scope for errors.

The %ctUM: EchoOn command will tell CatTools to wait for an echo of any further commands during this activity.

##### **%ctUM: Timeout xx**

When waiting for a response to a command CatTools will wait for a default time of 30 seconds. Some commands on some devices will take longer than this to respond. The %ctUM: Timeout XX command allows you to specify the time to wait for a response for all subsequent commands for this activity.

For example you may want to set the timeout for the show version command to 50 seconds, to do this you would do the following.

```
%ctUM: Timeout 50
show version
%ctUM: Timeout 0
```

Note the use of the command with a zero value(%ctUM: Timeout 0) sets the timeout back to the default value of 30 seconds.

#### **%ctUM:PauseTime xx**

This will cause CatTools to pause for the specified time in seconds before issuing the next command. The maximum time that can be specified is 3600.

For example, the below commands will issue a ping wait for 50 seconds then issue another ping.

```
ping 127.0.0.1
%ctUM: PauseTime 50
ping 127.0.0.1
```

#### **4.4.3.2 Meta Variables**

Several Activities allow you to enter a list of commands which will be issued to your selected devices.

[Device.CLI.Send commands](#), [Device.CLI.Modify Config](#) and [Device.Update.Banner](#) are examples of these activities.

Within the commands you send, you can include a number of CatTools [Meta Data](#) variables that will be translated when the Activity is run.

Example: In the Device.Update.Banner Activity you could set the following text and apply it to all your devices.

Hello, I am %ctDeviceName. Contact %ctContactName on %ctContactPhone to gain access to me.

Below is a table of the variables available.

Variable	Value
%ctDeviceName	The device name
%ctGroupName	The device group name
%ctHostName	The device host address
%ctModel	The device model field
%ctAddress1	The device Address1 field
%ctAddress2	The device Address2 field
%ctAddress3	The device Address3 field
%ctContactName	The device Contact name field
%ctContactPhone	The device Contact phone field
%ctContactEmail	The device Contact email field
%ctTimeHHMM	current time in HHMM format
%ctDateISO	current date in YYYYMMDD format
%ctBaseFile	device base file name
%ctAppPath	file system path of the CatTools executable including a trailing "\"
%ctCR	carriage return
%ctLF	line feed
%ctTAB	tab
%ctCTRL-Z	control key & Z. Normally used to exit

```
config mode.
```

**Note:** The variable names are case sensitive and must be entered as shown.

## 4.5 Creating a custom activity

CatTools provides a facility to create your own custom activities and activity script files should the CatTools built-in activities not suffice your requirements.

### Pre-requisites

A reasonable understanding or experience of Visual Basic Scripting is assumed in order to successfully add custom scripts to CatTools. However, the help file documentation and comments within the example code template files found in the /Templates sub folder of the CatTools root directory, should provide a reasonable level of assistance for a technically competent novice to follow.

### Overview

To add support for a custom activity in CatTools, four files are required. Three activity files and one custom device file:

#### Activity files:

- 1) The activity type file ([.ini](#) file), which defines the following:
  - activity name,
  - activity ID,
  - activity [main script](#) filename (associated with the activity),
  - activity [client script](#) filename (associated with the activity),
  - the user interface field values and defaults which are displayed in the activity form [Options](#) tab when adding or editing an activity.
- 2) The activity [main script](#) file ([.txt](#) file), which contains code to read the activity options from the CatTools database, prepare folders and files to store output data, set variables, marshal the CatTools Client threads and do any post processing of results in order to create reports or send messages to the CatTools main program.
- 3) The activity [client script](#) file ([.txt](#) file), which contains a number of common function calls to the device scripts, i.e. the scripts that send device specific commands in order to get the device to log in, issue the commands required to perform the activity, then log out of the device again.

#### Device file:

- 4) The device script file ([.custom](#) file), which contains device type specific code for the custom activity, for example, the commands to send to the device and any parsing of the data before sending the results back to the client activity script.

The activity client and main script files also contains function calls and references to variables within the internal CatTools program code. These are prefixed with 'cl.' in the client script and 'ct.' in the main script. A list of these cl. and ct. functions and variables have also been made available within this chapter to help assist in the development of your custom activity scripts.

- [How to create a custom activity](#) - a simple step-by-step guide on how to create a custom activity
- [The custom activity type file \(.ini\)](#) - information and how to create the custom activity type file
- [The custom activity main script file \(.txt\)](#) - information and how to create the custom activity main script file
- [The custom activity client script file \(.txt\)](#) - information and how to create the custom activity client script file
- [The custom activity device script file \(.custom\)](#) - information and how to create the custom activity device script file
- [cl. / ct. variables and functions](#) - information on the CatTools internal variables and functions exposed to the custom activity script files
- [Testing your custom activity](#) - help and tips on testing your custom activity

## 4.5.1 How to create a custom activity

### Quick reference guide to creating a custom activity in CatTools

This is a simple step-by-step guide on creating a custom activity in CatTools. It contains brief instructions on how to add a custom activity type to the CatTools GUI, and how to create a the custom main script and custom client script activity files, and how to make your custom activity available to device types.

The custom activity templates files provided by CatTools and referenced in this guide, are based on creating a simple Version report for a Cisco Router device. Once you have created the script (.txt and .custom) files, you will need to modify them for your particular activity or report.

Further information regarding the [custom activity type file \(.ini\)](#), [custom activity main script file \(.txt\)](#), [custom activity client script file \(.txt\)](#), [custom activity device script file \(.custom\)](#), [ct. code examples](#) and [cl. code examples](#) of how to use the CatTools internal functions and sub procedures in your activity scripts, can be found in the other sub pages of this chapter.

#### STEPS TO CREATE A CUSTOM ACTIVITY:

##### 1) Create a custom activity type file (.ini)

Take a COPY of the Custom.Activity.Template.ini in the \Templates sub folder of the CatTools root directory and save to the \Activities sub folder, giving it a new file name using the syntax: *Custom.Type.Name* (e.g. Custom.Report.Version).

##### 2) Edit the .ini file

Using a text file editor (such as Notepad), open the .ini file created in [step 1](#). You must change the following items in the .ini file from the template default values:

```
[Activity]
name=Custom.Activity.Template
script_main=Main.Custom.Activity.Template.txt
script_client=Client.Custom.Activity.Template.txt
report_file=%AppPath%Reports\Master Custom Activity Template.txt
id=3999
```

Change 'name' item to a UNIQUE activity name. Example: Custom.Report.PortStatus. Note: do not use spaces, use a 'period' mark (.) instead.

Change 'script\_main' item to the name of the associated Main script file (use activity name and prefix with 'Main.' to keep UNIQUE).



Change 'script\_client' item to the name of the associated Client script file (use activity name and prefix with 'Client.' to keep UNIQUE).

Change 'report\_file' item to a UNIQUE report name.

Change 'id' item to a UNIQUE number (i.e. one not used in any other activity .ini file). Number must be within the range of 4000 to 4999.

### 3) Save & restart

Once you have made your changes to the .ini file, save it back to the \Activities sub folder and close.

Restart CatTools to populate the 'Type' drop-down field in the activity setup screen with your new custom activity type.

### 4) Create a custom activity main script file (.txt)

Take a COPY of the Main.Custom.Activity.Template.txt in the \Templates sub folder of the CatTools root directory and save to the \Scripts sub folder, giving it a new file name as specified in the 'script\_main' item within the activity type .ini file **[Activity]** section in [step 2](#) above. Ensure you retain the .txt file type suffix.

### 5) Edit the main script .txt file

Using a text file editor\*, open the .txt file created in [step 4](#). Follow the instructions in the SCRIPT NOTES section of the .txt file to begin customizing the script for your activity.

### 6) Save the main script .txt file

Once you have made your initial changes to the main script .txt file, save it back to the \Scripts sub folder and close.

### 7) Create a custom activity client script file (.txt)

Take a COPY of the Client.Custom.Activity.Template.txt in the \Templates sub folder of the CatTools root directory and save to the \Scripts sub folder, giving it a new file name as specified in the 'script\_client' item within the activity type .ini file **[Activity]** section in [step 2](#) above. Ensure you retain the .txt file type suffix.

### 8) Edit the client script .txt file

Using a text file editor\*, open the .txt file created in [step 7](#). Follow the instructions in the SCRIPT NOTES section of the .txt file to begin customizing the script for your activity.

### 9) Save the client script .txt file

Once you have made your initial changes to the client script .txt file, save it back to the \Scripts sub folder and close.

### 10) Create a custom activity device script file (.custom)

Take a COPY of the Custom.Device.Template.txt.custom in the \Templates sub folder of the CatTools root directory and save to the \Scripts sub folder, giving it a new file name. You need to ensure you save the file using the same file name (including the .txt file type suffix) as its associated encrypted device script file (or a custom device script file), and then append an additional suffix '.custom'

Example: (creating a custom activity device script file for a Cisco Router)

Cisco router encrypted script file provide by CatTools: *Cisco.Router.General.txt*

Custom activity device script file name will therefore be: *Cisco.Router.General.txt.custom*

**11) Edit the device script .custom file**

Using a text file editor\*, open the .custom file created in [step 10](#). Read the SCRIPT NOTES section of the .custom file and begin customizing the script for your device. An [example](#) of the steps you need to follow to add a custom activity to the .custom device file can be found in [The custom activity device script file \(.custom\)](#) help file page.

**12) Save the device script .custom file**

Once you have made your changes to the device script .custom file, save it back to the \Scripts sub folder and close.

**13) Testing your custom activity scripts**

When all the custom activity scripts and the device .custom script files are set up, you can test the new activity within CatTools by setting up the activity to run with the device you created the .custom file for. Use the 'Run Now' button to test the custom activity manually rather than by starting the timer.

Check the [Info Log pane](#) for errors and edit your .txt files and .custom file as necessary.

See [Testing your custom activity](#) for more information and tips.

Creating custom activity checklist

- Create activity type file (.ini) \_\_\_\_\_
- Create activity Main script file (.txt) \_\_\_\_\_
- Create activity Client script file (.txt) \_\_\_\_\_
- Create activity device script file (.custom) \_\_\_\_\_

(\*note: although the .txt files can be opened and edited using 'NotePad', a syntax highlighting [script editor](#) may make reading and editing the .txt file much easier)

**4.5.2 The custom activity type file (.ini)****Activity types in CatTools**

Activity types are defined within CatTools using text files (.ini files) and are stored in the \Activities sub folder within the root CatTools directory. Each .ini file represents a separate item in the 'Type' drop-down list in the [Add/Edit scheduled activity details form](#) setup form.

When CatTools starts, it searches for all the .ini files in the \Activities sub folder and reads their contents. All the activity types found are then available for selection in the activity Type drop down when defining activities.

To add a new custom activity type, you need to create a new .ini file defining the activity and its characteristics.

**The .ini file sections overview**

The contents of the .ini file are divided up within [sections]. The main sections inside the INI file are as follows:

**[info] section**

This is required and identifies the .ini file as a CatTools file.

```
[info]
cookie=CatTools
version=3
author=SolarWinds
```

**[Activity]** section

This is required and defines the name used within the CatTools user interface (i.e. the Activity type drop-down list field) and within scripting. It also defines the unique key (id) of the activity type in the CatTools database.

```
[Activity]
name=Custom.Activity.Template
script_main=Main.Custom.Activity.Template.txt
script_client=Client.Custom.Activity.Template.txt
description=Collects information from devices and creates a custom report
report_file=%AppPath%Reports\Master Custom Activity Template.txt
report_overwrite=1
client_threads=0
id=3999
```

Although the file name of the .ini file may be similar (or the same) to the [Activity] section *name* item, it is the *name* item that defines the activity name within the user interface and not the .ini file name.

CatTools reserves id's from 0 to 3999 for predefined activity types. You can use the number range from 4000 to 4999 for the custom activity types you create, however you must ensure the number is unique.

**[item]** sections (Activity Options)

Each item section defines the input fields used within the CatTools user interface for the Activity setup screens.

Example:

```
[item_value1]
name=Use alternate command
type=string
checkbox_show=1
checkbox_default=0
value_default=
required=0
info="Alternative command to capture data from device."
```

Each input field in the CatTools user interface for Activity options has a separate [item\_xxx] type section.

The *name* item sets the text (or label) to be displayed next to the input or selection field.

The *type* item describes the type of input or selection field adjacent to the *name* text/label.

The *checkbox\_show* and *checkbox\_default* items describe whether a checkbox should be displayed and its default value of checked or unchecked.

The *value\_default* item sets the default value to be used when adding a new activity to the database.

The *required* item tells the user interface if the field must contain valid data or not (for example: must be an item from within a predefined list or whether the user can enter their own values). 0=not required, 1=required.

The *info* item is the description associated with this field. This text is displayed in the status bar of the user interface when a field has focus.

The above [item\_value1] section is an example of a checkbox and standard text box input field

within the user interface.

If you need to limit a range of numerical values that can be entered into an option field, then use the type or 'range'.

The range option example below, contains a checkbox and an input field that accepts numerical values between 1 and 2000.

```
[item_value2]
name=Days until entries expire
type=range
checkbox_show=1
checkbox_default=0
value_default=30
range_min=1
range_max=2000
required=1
info="Remove entries older than X days from the Master report"
```

Check box input fields can only accept values of 0 or 1. Any value other than a 1 is considered a 0. A check box example is below.

The check box is defaulted to 1 (checked).

```
[item_value3]
name=Resolve IP addresses to hostnames
type=checkbox
checkbox_show=1
checkbox_default=1
info="Resolve all IP addresses found to hostnames via reverse DNS lookup"
```

The size (length of the data that can be entered) of each field is determined by the design of the CatTools database. Any data entered via the CatTools user interface that is too long for the field will be truncated accordingly.

### **The custom activity type .ini template**

CatTools provides a starting point template file to help assist in the creation of a new custom activity type.

The template file is called *Custom.Activity.Template.ini* and can be found in the \Templates sub folder.

This template file is included as part of the predefined activity types in CatTools. As for all of the CatTools predefined activity types, it may (as and when required) be subject to updates and modifications with each new release of the CatTools product.

For this reason, you should never modify the template file itself in order to create a new custom activity type.

If you need to create a new custom activity type, take a copy the template file and save it to the \Activities sub folder giving it a new file name.

### **Creating your custom activity type .ini file**

If adding a new custom activity type to CatTools, the first step will therefore be to take a copy of the template file *Custom.Activity.Template.ini* in the \Templates sub folder and save it to the \Activities sub folder giving it a new file name.

This will give you a new starting point activity type file to work with. When naming the file, try to use the naming convention:

*Custom.Type.Name*

<i>Custom</i>	being the text "Custom" to distinguish custom activity types from CatTools predefined types. They will also be grouped together in the activity 'Type' drop-down list field.
<i>Type</i>	the type of activity (Report, CLI, Connectivity, Update)
<i>Name</i>	the name of activity (ARPTable, SendCommands, PingTest, BannerText)

**Editing the .ini file**

The next step is to open the .ini file and make the required changes to the values within each section.

Below is an example of an .ini file with a number of items values highlighted.

The most important item values which *MUST* be changed are **highlighted in Red**. These items **must be unique** otherwise your activity type may not appear in the CatTools activity 'Type' drop-down list field.

Items values which *may* be changed if desired are **highlighted in Green**.

Additional comments are **highlighted in Blue** (*Note*: do not include any comments in your .ini file)

It is recommended that anything else that is not highlighted be left at its original setting. You can however, customise some of the items such as the Name and Info items to make them more relevant to your activity.

The text value within the 'Name' item is the field Label that is displayed within the form. The text value within the 'Info' item is the text that is displayed in the Status bar when you select the field in the form (i.e. gets focus).

Once you have made your amendments to the .ini file, remember to save it back to the \Activities sub folder.

You will then need to restart CatTools in order for your new custom activity type to be read into the activity 'Type' drop-down list field.

```
[info]
cookie=CatTools
version=3
author=SolarWinds enter your name or leave as the default

[Activity]
name=Custom.Activity.Template change to a unique activity name. example: Custom.Report.PortStatus Note: do not use spaces. Use a 'period' mark (.)
script_main=Main.Custom.Activity.Template.txt change to associated Main script name (use activity name and prefix with 'Main.' to keep unique)
script_client=Client.Custom.Activity.Template.txt change to associated Client script name (use activity name and prefix 'Client.' to keep unique)
description=Collects information from devices and creates a custom report change if required
report_file=%AppPath%\Reports\Master Custom Activity Template.txt change to a unique report name
report_overwrite=1
client_threads=0
id=3999 select a number within the range of 4000 to 4999. The number used here must be unique.

# Activity options

[item_value1]
name=Use alternate command
```

```
type=string
checkbox_show=1
checkbox_default=0
edit_button=0
value_default=
required=0
info="Alternative command to capture data from device."
```

[\[item\\_value2\]](#)

*... add more options items as required (to a maximum of 10 options).*

### 4.5.3 The custom activity main script file (.txt)

#### Activity scripts in CatTools

Activity scripts are defined within CatTools using text files (.txt files) and are stored in the \Scripts sub folder within the root CatTools directory.

When an activity is run, CatTools reads the *script\_main* item value in the **[Activity]** section from the activity type .ini file to determine which activity main script file it needs to use.

(Its recommended that in order to save confusion, the .ini file and .txt files have the same file name apart from the file prefix of 'Main.' for the .txt script file)

For all the predefined activity types in CatTools, the associated activity main script .txt files have been encrypted. They are encrypted for two reasons. The first is to protect our intellectual property. The other is to prevent unauthorised modification of these files which may cause the scripts to fail at runtime.

#### The custom activity main script .txt template

CatTools provides a starting point main script template file to help assist in the creation of a new custom activity main script.

The template file is called *Main.Custom.Activity.Template.txt* and can be found in the \Templates sub folder within the root CatTools directory.

This template file is included as part of the predefined activity scripts in CatTools, but is unencrypted. As for all of the CatTools predefined activity scripts, it may (as and when required) be subject to updates and modifications with each new release of the CatTools product.

For this reason, you should never modify the template file itself in order to create a new activity custom main script.

If you need to create a new custom activity main script, take a copy the template file and save it to the \Scripts sub folder giving it a new file name.

#### Creating your custom activity main script .txt file

If adding a new custom activity main script to CatTools, the first step will therefore be to take a copy of the template file *Main.Custom.Activity.Template.txt* in the \Templates sub folder and save it to the \Scripts sub folder, giving it with a new file name. You need to ensure the file name you use is the value entered for the *script\_main* item within the activity type .ini file **[Activity]** section.

This will give you a new starting point device script file to work with.

#### Editing the main script .txt file

The next step is to open the custom main script .txt file and make any necessary changes in order to get the script to work with your device(s).

The .txt file is well commented (documented) to provide instructions and assistance in making your modifications, however there are a few important sections at the top of the script file which require further mention.

You may see the following line at the very top of the script:

```
Attribute VB_Name = "CustomActivity_Template"
```

This is the Visual Basic module name for the custom activity main script file when viewing within the Visual Basic (or equivalent) development environment. This line won't appear if editing the file in the VB development environment, but may do in others.

"CustomActivity\_Template" is the name given to the custom template file module.

If you have a number of custom activity main script files, you may want to consider changing this name to reflect your custom activity main script file names. By doing so, you can then open multiple custom activity main scripts (say within a project) in your VB development environment.

If you are using a text based editor to modify your scripts (such as Notepad.exe), then changing the module name is optional, although to avoid confusion you may prefer to change the name anyway.

#### Notes:

```
--- SCRIPT NOTES ---
```

The SCRIPT NOTES section provides useful information and lists tasks that are required to be carried out during the initial script creation phase. You can edit this section as and when you feel it is appropriate. An example may be deleting the initial setup tasks block of text from the SCRIPT NOTES section after you have carried out the initial script setup tasks, as it no longer applies.

Any complex routines or script specific behaviour which may be of assistance to any person maintaining the activity scripts, should be mentioned in the SCRIPT NOTES section. They should be either fully documented in the SCRIPT NOTES section, or a reference made to the relevant function where they are fully documented.

```
--- ACTIVITY NOTES ---
```

The ACTIVITY NOTES section is an area where you can enter any activity specific behaviour or quirks which may be of assistance to any person maintaining the activity scripts. An example may be: *"The activity only works for Cisco IOS devices"*.

#### Required functions:

There is only one function that must exist within your custom activity main script in order for CatTools to run the activity:

```
Function Main()
```

You may also define your own Functions or Sub routines in the activity main script, however they must be called from within Function Main(). Remember, it is good practice to comment your Functions and Sub routines, so that you or other developers can get an idea of what the function does at a later date.

#### **Saving the script file**

Before CatTools can access the script file you have built, you must save it to the \Scripts sub

folder. Normally you do not have to stop and restart CatTools in order for script file code changes to be picked up, however there may be occasions during testing that you find you may have to do this if the changes are not being recognised.

### Testing your custom activity scripts

Once you have all the custom activity scripts set up, you can test the new activity with CatTools by setting up the activity to run with an existing device.

The device selected must have had a [.custom](#) file created and contain the necessary code relevant to run your activity, otherwise you will see a error be logged to the Info Log of:

```
"Client script error: Type Mismatch: [ activity name] on line: [ line number] "
```

See [Testing your custom activity](#) for more information and tips.

### Send us your working scripts

Once you have successfully created support for your custom activity in CatTools, if you would like us to consider adding it as a predefined activity type to ship with the product, then please send your custom client and main script .txt files, the custom activity type .ini file and also an example of a device .custom file which references the custom activity.

Please send as an attachment using the [Technical Support](#) form on our [web site](#).

There are no guarantees as to when or if the activity will be added to the predefined activity types in CatTools, as there are a number of factors to consider: e.g. the number of requests for the activity, complexity of the script, technical resources available, etc; however all scripts that are sent in will be cataloged for future reference.

## 4.5.4 The custom activity client script file (.txt)

### Activity scripts in CatTools

Activity scripts are defined within CatTools using text files (.txt files) and are stored in the \Scripts sub folder within the root CatTools directory.

When an activity is run, CatTools reads the *script\_client* item value in the **[Activity]** section from the activity type .ini file to determine which activity client script file it needs to use.

(Its recommended that in order to save confusion, the .ini file and .txt files have the same file name apart from the file prefix of 'Client.' for the .txt script file)

For all the predefined activity types in CatTools, the associated activity client script .txt files have been encrypted. They are encrypted for two reasons. The first is to protect our intellectual property. The other is to prevent unauthorised modification of these files which may cause the scripts to fail at runtime.

### The custom activity client script .txt template

CatTools provides a starting point client script template file to help assist in the creation of a new custom activity client script.

The template file is called *Client.Custom.Activity.Template.txt* and can be found in the \Templates sub folder within the root CatTools directory.

This template file is included as part of the predefined activity scripts in CatTools, but is unencrypted. As for all of the CatTools predefined activity scripts, it may (as and when



required) be subject to updates and modifications with each new release of the CatTools product.

For this reason, you should never modify the template file itself in order to create a new activity custom client script.

If you need to create a new custom activity client script, take a copy the template file and save it to the \Scripts sub folder giving it a new file name.

### Creating your custom activity client script .txt file

If adding a new custom activity client script to CatTools, the first step will therefore be to take a copy of the template file *Client.Custom.Activity.Template.txt* in the \Templates sub folder and save it to the \Scripts sub folder, giving it with a new file name. You need to ensure the file name you use is the value entered for the *script\_client* item within the activity type .ini file **[Activity]** section.

This will give you a new starting point device script file to work with.

### Editing the client script .txt file

The next step is to open the custom client script .txt file and make any necessary changes in order to get the script to work with your device(s).

The .txt file is well commented (documented) to provide instructions and assistance in making your modifications, however there are a few important sections at the top of the script file which require further mention.

You may see the following line at the very top of the script:

```
At t r i but e VB_Name = " Act _Cust om_Templ at e"
```

This is the Visual Basic module name for the custom activity client script file when viewing within the Visual Basic (or equivalent) development environment. This line won't appear if editing the file in the VB development environment, but may do in others.

" Act \_Cust om\_Templ at e" is the name given to the custom template file module.

If you have a number of custom activity client script files, you may want to consider changing this name to reflect your custom activity client script file names. By doing so, you can then open multiple custom activity client scripts (say within a project) in your VB development environment.

If you are using a text based editor to modify your scripts (such as Notepad.exe), then changing the module name is optional, although to avoid confusion you may prefer to change the name anyway.

### Notes:

```
- - - SCRI PT NOTES - - -
```

The SCRIPT NOTES section provides useful information and lists tasks that are required to be carried out during the initial script creation phase. You can edit this section as and when you feel it is appropriate. An example may be deleting the initial setup tasks block of text from the SCRIPT NOTES section after you have carried out the initial script setup tasks, as it no longer applies.

Any complex routines or script specific behaviour which may be of assistance to any person maintaining the activity scripts, should be mentioned in the SCRIPT NOTES section. They should be either fully documented in the SCRIPT NOTES section, or a reference made to the relevant function where they are fully documented.

### --- ACTIVITY NOTES ---

The ACTIVITY NOTES section is an area where you can enter any activity specific behaviour or quirks which may be of assistance to any person maintaining the activity scripts. An example may be: "*The activity allows 3 Options:...*" (then go on to list what each option does and its input values).

#### Required functions:

There is only one function that must exist within your custom activity client script in order for CatTools to run the activity:

```
Function Client()
```

You may also define your own Functions or Sub routines in the activity client script, however they must be called from within Function Client(). Remember, it is good practice to comment your Functions and Sub routines, so that you or other developers can get an idea of what the function does at a later date.

#### **Saving the script file**

Before CatTools can access the script file you have built, you must save it to the \Scripts sub folder. Normally you do not have to stop and restart CatTools in order for script file code changes to be picked up, however there may be occasions during testing that you find you may have to do this if the changes are not being recognised.

#### **Testing your custom activity scripts**

Once you have all the custom activity scripts set up, you can test the new activity with CatTools by setting up the activity to run with an existing device.

The device selected must have had a [.custom](#) file created and contain the necessary code relevant to run your activity, otherwise you will see a error be logged to the Info Log of:

```
"Client script error: Type Mismatch: [ activity name] on line: [ line number] "
```

See [Testing your custom activity](#) for more information and tips.

#### **Send us your working scripts**

Once you have successfully created support for your custom activity in CatTools, if you would like us to consider adding it as a predefined activity type to ship with the product, then please send your custom client and main script .txt files, the custom activity type .ini file and also an example of a device .custom file which references the custom activity.

Please send as an attachment using the [Technical Support](#) form on our [web site](#).

There are no guarantees as to when or if the activity will be added to the predefined activity types in CatTools, as there are a number of factors to consider: e.g. the number of requests for the activity, complexity of the script, technical resources available, etc; however all scripts that are sent in will be cataloged for future reference.

## 4.5.5 The custom activity device script file (.custom)

### Device scripts in CatTools

Devices scripts are defined within CatTools using text files (.txt files) and are stored in the \Scripts sub folder within the root CatTools directory.

When an activity is run, CatTools reads the *name* item value in the **[device]** section from the device type .ini file to determine which device script .txt file it needs to use. Therefore each .txt file must be given the same file name as the *name* item in the corresponding .ini file.

For all the predefined device types in CatTools, the associated device script .txt files have been encrypted. They are encrypted for two reasons. The first is to protect our intellectual property. The other is to prevent unauthorised modification of these files which may cause the script to fail at runtime.

### Device '.custom' file extension scripts

With the introduction of custom activity scripting in CatTools, in order to facilitate the adding of custom activities to the predefined device types, CatTools now checks for the existence of an associated device .custom file in the \Scripts folder, after loading the encrypted device .txt script file.

If when running an activity an associated device .custom file is found for a device type, the contents of the .custom file are read and appended to the predefined device type encrypted script contents, creating a temporary extended device type script, which is then used for the activity. The device .custom file and the device encrypted files are merged in this manner every time an activity is run.

A device .custom file is associated with its encrypted script by file name, so for example, to add a .custom file for a Cisco Router (*Cisco.Router.General.txt*), you need to create a file called *Cisco.Router.General.txt.custom* in the \Scripts folder.

Whenever you upgrade or reinstall CatTools on your system, the predefined device type and activity type script files are overwritten. By defining your custom activity function calls for devices in a separate .custom file, it ensures that your custom functions do not get inadvertently lost.

### The device '.custom' template

CatTools provides a starting point template file to help assist in the creation of a new custom activity device .custom file.

The template file is called *Custom.Device.Template.txt.custom* and can be found in the \Templates sub folder.

This template file is included as part of the predefined scripts in CatTools, but is unencrypted. As for all of the CatTools predefined scripts, it may (as and when required) be subject to updates and modifications with each new release of the CatTools product.

For this specific reason, you should never modify the template file itself in order to create a new device .custom script.

If you need to create a new device .custom script, then take a copy the template file and save it to the \Scripts sub folder giving it a new file name.

### Creating your device '.custom' script file

If adding a new .custom script to CatTools, the first step will therefore be to take a copy of the

template file *Custom.Device.Template.txt.custom* in the \Templates sub folder and save it to the \Scripts sub folder, giving it with a new file name. You need to ensure you save the file using the same file name (including .txt extension) as its associated encrypted device script file, and append a suffix '**.custom**'

This will give you a new starting point device script file to work with.

*For example:*

You have created a new custom activity called "*Custom\_Report\_PortStatus*" and want to use this activity with your Cisco IOS switch devices.

Your custom activity Client script file *Client.Custom.Report.PortStatus.txt* needs to call the function *Custom\_Report\_GetPortStatus()* in order to send a command, capture port data and parse the results to send back to the activity Client script.

To add this custom activity to the Cisco IOS switches (device type script file *Cisco.Switch.IOS.txt*), you need to perform the following steps:

- 1) Create a **.custom** file for the Cisco IOS switch device type. Copy the .custom template file *Custom.Device.Template.txt.custom* from the \Templates sub folder and save it to the \Scripts sub folder with the file name of *Cisco.Switch.IOS.txt.custom* in order for it to be merged with the *Cisco.Switch.IOS.txt* script at run-time.
- 2) Open the .custom file and delete the example functions.
- 3) Add your custom activity function *Custom\_Report\_GetPortStatus()* and code.

Initially, you may just want to test the function is being called correctly, so below is a simple example of the code needed to add an 'Info' message in the Info Log from within your function.

```
Function Custom_Report_GetPortStatus()
    cl.Log 3, "Testing custom activity: Custom_Report_PortStatus"
End Function
```

- 4) Save the file.

### Editing the .custom file

Basically, there are only three issues to be aware of with regards to editing the .custom script file.

After copying the .custom template file in the \Templates sub folder to the \Scripts sub folder, please ensure that:

- 1) Do not delete the first line in the template file

```
Attribute VB_Name = "Dev_CustomDeviceTemplate_custom"
```

You can modify the value within the " " quotes, but it must always be defined as the Attribute VB\_Name, otherwise a run-time error is likely to occur.

- 2) It is recommended you do not enter any uncommented code (program code) in the header section and most importantly DO NOT add any above the Attribute VB\_Name line, otherwise your code may be stripped out or alternatively you may encounter a run-time error.

The SCRIPT NOTES section within the .custom file header is there for your own reference. Please feel free to add your own comments in here if it will be of assistance to yourself or

another developer at a later date.

3) Ensure that all program code added is contained within either a Function or a Sub routine. To avoid run-time errors or the risk of redefining a variable or object which is used in the predefined encrypted device script, try keep variable and object declarations limited to the scope of a Function or Sub routine. Remember, it is good practice to comment your Functions and Sub routines, so that you or other developers can get an idea of what the function does at a later date.

### **Saving the .custom script file**

Before CatTools can access the .custom script file you have built, you must save it to the \Scripts folder. Normally you do not have to stop and restart CatTools in order for script file code changes to be picked up, however there may be occasions during testing that you find you may have to do this if the changes are not being recognised.

### **Testing your custom device .custom scripts**

Once you have the scripts set up you can test the custom activity with CatTools by setting up the activity and selecting the device you have created the .custom file for. See [Testing your custom device](#) for more information and tips.

### **Send us your working scripts**

Once you have successfully created support for your custom device in CatTools, if you would like us to consider adding it as a predefined device type to ship with the product, then please send your custom client script .txt file and custom device type .ini file as an attachment using the [Technical Support](#) form on our [web site](#).

There are no guarantees as to when or if the device will be added to the predefined device types in CatTools, as there are a number of factors to consider: e.g. the number of requests for the device, complexity of the script, technical resources available, etc; however all scripts that are sent in will be cataloged for future reference.

## **4.5.6 Custom activity scripts cl. and ct. variables and functions**

The custom activity client script and main script files contain references to the internal CatTools variables, functions and sub routines.

These variables, functions and sub routines can be identified by their 'cl.' (client) or 'ct.' (main CatTools program) prefixes.

Although these functions, variables and sub routines are not documented within the template scripts themselves (mainly to cut down the size of the script files), in order for you to understand what each one is used for and in the case of the functions understand the parameters they require and what their return values are; each variable, function and sub routine exposed in the custom activity template files are detailed within the following sub pages of this chapter.

- [Client script file cl. functions and variables](#)
- [Main script file ct. functions and variables](#)

#### 4.5.6.1 Client script - cl. variables and functions

Below is a list of CatTools client 'cl.' variables, functions and sub routines, with some examples of how to implement them in your custom activity client script files. This is not a full set of all the internal client variables and functions used within the CatTools application, however it contains details of those from within the client script template files, plus a few additional ones which may be useful when creating your custom activity client scripts.

##### VARIABLES

cl.AppPath	The application path (install directory) of the CatTools program
cl.ScheduleNumber	The current schedule number
cl.DeviceName	The name of the device currently connected to
cl.TelnetConnectionStatuscontrol	The connection status for the instance of the (protocol engine control) client thread.

Example: Test current client connection status and exit function if status is not 'OK'.

```
If StrComp( cl . Tel net Connect i onSt at us, " OK" ,  
vbText Compare) <> 0 Then Exit Funct i on
```

cl.QuitNow	A variable that is checked in a number of places within an activity to stop it running. It is set to 1 (for example) when the 'STOP' button is pressed while an activity is running. This variable can only be referenced from within activity Client Scripts.
------------	--

Use the *cl.QuitNow* variable to reference within Main Scripts

##### FUNCTIONS & SUB ROUTINES - Summary list

###### **General & file**

cl.Log	Sends a line of text to the InfoLog.txt file and Info Log pane
cl.LogToFile	Writes data to a file
cl.FileExists	Checks for the existence of a file
cl.CreateFile	Create a new file using a given filename and path
cl.DeleteFile	Delete a file using a given filename and path
cl.ReadFromFile	Read in contents of a given file
cl.ReplaceClientFilenameVariables	Replace filename variables in client report filenames or meta variables
cl.SaveResults	Save results to a temporary .txt file within the \ClientTemp folder

###### **Activity**

cl.DBCheckScheduleOption	Determine whether a particular option has been selected within a given activity
cl.DBScheduleGetField	Get activity field value from the CatTools database

###### **Device**

cl.DisconnectHost	Disconnect instance of the (protocol engine control) client thread
-------------------	--

from the end device

### FUNCTIONS & SUB ROUTINES - Details and examples

The functions and sub routines listed in the above table are further detailed below with their input parameters, etc. and (in some cases) examples of their usage are provided.

#### **cl.Log (iPriority, sMessage)**

This sub routine sends a line of text to the Infolog.txt file and [Info Log pane](#).

It has two input parameters:

<i>iPriority</i>	Integer range 1 to 4 representing the logging 'level' for the line being sent. 1=Error, 2=Warning, 3=Information, 4=Debug
<i>sMessage</i>	Message text of the line being sent

Example: log a level 3 (info) message if the 'Use alternate command' field in the activity [Options tab](#) is empty.

```
Dim sAlt Command

sAlt Command = cl.DBScheduleGetField(cl.ScheduleNumber,
"OptionsString1")
If Len(Trim(sAlt Command)) = 0 Then
    cl.Log 3, "No alternate command specified"
End if
```

#### **cl.LogToFile (sFilename, sData, bAppend)**

This sub routine writes data to a given file.

It has three input parameters:

<i>sFileName</i>	String of the filename and path of the file to write data to
<i>sData</i>	String of data to write
<i>bAppend</i>	Boolean value. If bAppend is NOT 0 (i.e. not False), then the file will be appended to

Example: log a level 2 (warning) message to the Info Log and write a line to a text file in the Reports folder if the 'Use alternate command' field in the activity [Options tab](#) is empty.

```
Dim sLogResultsFile
Dim sAlt Command

sLogResultsFile = cl.AppPath & "Reports\" & cl.UniqueField & ".txt"
sAlt Command = cl.DBScheduleGetField(cl.ScheduleNumber,
"OptionsString1")

If Len(Trim(sAlt Command)) = 0 Then
    cl.Log 2, "No alternative command specified"
    Call cl.LogToFile(sLogResultsFile, "No alternative command
specified in activity Options tab", 1)
End if
```

#### **cl.FileExists(sFileName) As Boolean**

This function checks for the existence of a given file. The function returns **True** if the file was

found, **False** if not.

It has one input parameter:

*sFileName*            String of the filename and path of the file that existence is being tested for

#### **cl.CreateFile(sFileName) As Boolean**

This function is used to create a new file using the given filename and path. The function returns **True** if the file already exists, or the file creation was successful, else it returns **False**.

It has one input parameter:

*sFileName*            String of the filename and path of the file to create

#### **cl.DeleteFile(sFileName) As Boolean**

This function is used to delete a file using the given filename and path. The function returns **True** if the file has been deleted successfully, else it returns **False**.

It has one input parameter:

*sFileName*            String of the filename and path of the file to delete

#### **cl.ReadFromFile(sFileName) As String**

This function is used to read in the contents of a given file. The function returns a string of the file data.

It performs the same operation as the CatTools function ct.ReadFromFile, but can only be called from within an activity Client Script.

It has one input parameter:

*sFileName*            String of the filename and path of the file to read in the contents of

Example: read in the contents of a given file (in this case a list of CLI commands).

```
Dim sFileName
Dim sCommandList

If cl.DBCheckScheduleOption(cl.ScheduleNumber, 1) = 0 Then
    ' Option selected to load CLI commands to send to device from a file
    sFileName = cl.DBScheduleGetField(cl.ScheduleNumber,
    "OptionsString2")
    sCommandList = cl.ReadFromFile(sFileName)
End If
```

#### **cl.ReplaceClientFilenameVariables(sData) As String**

This function is used to replace any [filename variables](#) in the client report filenames or [meta variables](#) in the commands to be sent to the device, with the actual text values. It returns the modified string with the variables replaced.

It has one input parameter:

*sData*                String of the filename and path of the file; or command list being checked



**Example:** replace any filename variables of a given file, then read in its contents (in this case a list of CLI commands). Next, replace and meta variables within the command list.

```

Dim sFileName
Dim sCommandList

If cl.DBCheckScheduleOption(cl.ScheduleNumber, 1) = 0 Then

    ' Option selected to load CLI commands to send to device from a file
    sFileName = cl.DBScheduleGetField(cl.ScheduleNumber,
    "OptionsString2")

    ' Replace any filename variables
    sFileName = cl.ReplaceClientFilenameVariables(sFileName)

    ' Read in the file contents
    sCommandList = cl.ReadFromFile(sFileName)

    ' Replace meta (command) variables with values
    sCommandList = cl.ReplaceClientFilenameVariables(sCommandList)

End If

```

#### **cl.SaveResults(sData, bAppend)**

This sub routine saves the results to a temporary .txt file within the \ClientTemp folder

It has two input parameters:

<i>sData</i>	String of data to write
<i>bAppend</i>	Boolean value. If bAppend is 0 (False), then the file will be overwritten, otherwise it is appended to

**Example:** save current device response (sResults) to a temporary .txt file overwriting any existing data.

```
Call cl.SaveResults(sResults, 0)
```

#### **cl.DBCheckScheduleOption(IScheduleNumber, IOptionNumber) As Long**

This function is used to determine whether particular options have been selected within a given activity.

It has two input parameters:

<i>IScheduleNumber</i>	The current schedule number as a Long
<i>IOptionNumber</i>	The option item within the activity we are checking as a Long. <i>IOptionNumber</i> must be between 1 and 10 (inclusive)

**Example:** check an activity (in this instance a [Device.CLI.Send commands](#) activity) to see if the output of the command(s) should be emailed.

```

I RetVal = cl.DBCheckScheduleOption(cl.ScheduleNumber, 8)
If I RetVal = 1 then
    ' code to email commands goes here...
End If

```

**cl.DBScheduleGetField(IScheduleNumber, sFieldName)**

This function is used to get a field value from the CatTools database for a given activity. It performs the same operation as the CatTools function ct.DBScheduleGetField, but can only be called from within an activity Client Script.

It has two input parameters:

*IScheduleNumber* The current schedule number as a Long  
*sFieldName* The field name in the database we want to get the value of as a String

**Example:** (used in conjunction with the **cl.DBCheckScheduleOption** function above) check an activity to see if the 'Use alternate command' check-box is selected in the activity [Options tab](#). If it is, it gets the value from the database for the alternate command we want to send.

```
' Check the option 1 check-box to see if selected - i.e. set to 1
If cl . DBCheckScheduleOption( cl . ScheduleNumber , 1) = 1 Then
    ' Get value from database for associated string field (being field "OptionsString1"
    in the database) to set a variable value
    sAltCommand = cl . DBScheduleGetField( cl . ScheduleNumber ,
    "OptionsString1" )
End If
```

**cl.DisconnectHost**

This sub routine is used to disconnect the instance of the (protocol engine control) client thread from the end device.

If should be called after the device has been logged out of to disconnect the client thread gracefully.

**4.5.6.2 Main script - ct. variables and functions**

Below is a list of CatTools application '**ct.**' variables, functions and sub routines, with some examples of how to implement them in your custom activity main script files. This is not a full set of all the internal ct. variables and functions used within the CatTools application, however it contains details of those from within the main script template files, plus a few additional ones which may be useful when creating your custom activity main scripts.

**VARIABLES**

ct.ScheduleNumber	The current schedule number
ct.Devices	A pipe ( ) delimited string of devices to connect to (i.e. the devices associated with the current schedule)
ct.ScriptFileClient	A string variable to store the file name of the client script file that is used by the activity
ct.QuitNow	A variable that is checked in a number of places within an activity to stop it running. It is set to 1 (for example) when the 'STOP' button is pressed while an activity is running. This variable can only be referenced from within an activity Main Script. Use the <i>cl.QuitNow</i> client variable is referencing within the Client Scripts

## FUNCTIONS & SUB ROUTINES - Summary list

### **General & file**

ct.Log	Sends a line of text to the Infolog.txt file and Info Log pane
ct.ReadFromFile	Read in contents of a given file
ct. ReplaceFilenameVariables	Replace filename variables in report filenames to real text values
ct. CreateMasterTable	Creates a master table file
ct. CreateAndSendReport	Creates report files (.txt .html, etc.) and sends by email
ct. RemoveClientResults	Removes existing client temporary .txt files within the \ClientTemp folder

### **Activity**

ct.MarshalThreads	Manages the client threads used within the activity schedule
ct. DBScheduleGetField	Get activity field value from the CatTools database

### **Device**

ct.RemoveHeader	Removes the first line of data from within a data block
-----------------	---

## FUNCTIONS & SUB ROUTINES - Details and examples

The functions and sub routines listed in the above table are further detailed below with their input parameters, etc. and (in some cases) examples of their usage are provided.

### **ct.Log(sDevice, iPriority, sMessage)**

This sub routine sends a line of text to the Infolog.txt file and [Info Log pane](#).

It has three input parameters:

<i>sDevice</i>	The name of the device as a string. If an empty string (or VBNullString) then the string value will be "CatTools" referring to an internal application related log message rather than a device specific message
<i>iPriority</i>	integer range 1 to 4 representing the logging 'level' for the line being sent. 1=Error, 2=Warning, 3=Information, 4=Debug
<i>sMessage</i>	Message text of the line being sent

**Example:** log a level 1 (error) message for CatTools (rather than any specific device, i.e. sDevice = "") with the text "No devices have been selected".

```
ct . Log " ", 1, " No devices have been selected"
```

### **ct.ReadFromFile(sFileName) As String**

This function is used to read in the contents of a given file. The function returns a string of the

file data.

It performs the same operation as the client function `cl.ReadFromFile`, but can only be called from within an activity Main Script.

It has one input parameter:

*sFileName* String of the filename and path of the file to read in the contents of

**Example:** read in the contents of a file called "ReportData", removing the first row header row.

```
Dim sReportFile
Dim sReportData

' Get the file name and path from the activity settings and replace any filename
variables.
sReportFile = ct.DBScheduleGetFileId(ct.ScheduleNumber, "ReportFile")
sReportFile = ct.ReplaceFilenameVariables("", sReportFile)

' Read the contents of the report file, remove the first line of data (header row) and
save the results to sReportData variable.
sReportData = ct.RemoveHeader(ct.ReadFromFile(sReportFile))
```

#### **ct.ReplaceFilenameVariables(sDeviceName, sFileName) As String**

This function is used to replace any [filename variables](#) in the report filename to the actual text values. It returns the modified string with the filename variables replaced.

It has two input parameters:

*sDeviceName* String value for the device name. Set *sDeviceName* to be an empty string ("") when you are translating an activity Report file name, as these should not contain device specific filename variables

*sFileName* String of the filename and path of the file being checked

**Example:** get the report filename for an activity and perform any filename variable replacements, then store the value in a string variable.

```
sReportFile = ct.DBScheduleGetFileId(ct.ScheduleNumber, "ReportFile")
sReportFile = ct.ReplaceFilenameVariables("", sReportFile)
```

#### **ct.CreateMasterTable(sReportFile, sHeader, bOverwrite, sDeviceList, sClientFilePrefix, sTableFormat) As Boolean**

This function is used to create the master table file. If successful, it returns a value of **True**, else if not then returns **False**.

It has six input parameters:

*sReportFile* String value of the report/master table file name and path to create

*sHeader* Tab delimited string for the report header as defined in the activity Main Script

*bOverwrite* Boolean value. If 0 (False) the file will be appended to, otherwise it is overwritten

*sDeviceList* Pipe (|) delimited string of devices associated with the current schedule (use the "ct.Devices" variable)

*sClientFilePrefix* File name prefix string for the temporary client (device) files that the activity creates in \ClientTemp in order to consolidate for the Master table file

*sTableFormat* Pipe (|) delimited string used for those activities where you can modify the table definition output (change column names; include/exclude columns)

from the output; and change the table column order:  
For example, the Options tab of the [Report.SNMP.System Summary](#) activity)

**Example:** create the master table for a version report. Get the Report file name and Overwrite values from the activity settings in the database; and define the report header. Use a prefix of "CustomReportTemplate" for each temporary client (device) file.

```
Dim sReportFile
Dim bOverwrite
Dim sHeader

sReportFile = ct.DBScheduleGetField(ct.ScheduleNumber, "ReportFile")
sReportFile = ct.ReplaceFilenameVariables("", ReportFile)

bOverwrite = ct.DBScheduleGetField(ct.ScheduleNumber,
"ReportOverwrite")

sHeader = "Group" & vbTab & "Device Name" & vbTab & "IP Address" &
vbTab & "Serial #" & vbTab & _
"Processor" & vbTab & "IOS" & vbTab & "Uptime"

' Create the master table
bRetVal = ct.CreateMasterTable(sReportFile, sHeader, bOverwrite, ct.
Devices, "CustomReportTemplate", "")
```

#### **ct.CreateAndSendReport(sScheduleName, sHeader, sData, sReportFile)**

This sub routine is used to create the report files (.txt .html, etc.) and if the relevant options are set, it will also send the reports by email.

It has four input parameters:

<i>sScheduleName</i>	The schedule name used for the report title, and email message subject title
<i>sHeader</i>	Tab delimited string for the report header as defined in the activity Main Script
<i>sData</i>	String of the report data
<i>sReportFile</i>	String value of the report file name and path to create

**Example:** create a simple version report. Get the Activity name and Report file name from the activity settings in the database; and define the report header.

```
Dim sClientFile
Dim sScheduleName
Dim bOverwrite
Dim sReportFile
Dim sHeader

' Prefix for temporary client file filenames
sClientFile = "CustomReportTemplate"

' Get the schedule name from the activity settings
sScheduleName = ct.DBScheduleGetField(ct.ScheduleNumber, "Name")

' Get overwrite option value
bOverwrite = ct.DBScheduleGetField(ct.ScheduleNumber,
"ReportOverwrite")

' Get report file name
```

```

sReportFile = ct.DBScheduleGetField(ct.ScheduleNumber, "ReportFile")
sReportFile = ct.ReplaceFilenameVariables("", ReportFile)

' Define the report header
sHeader = "Group" & vbTab & "Device Name" & vbTab & "IP Address" &
vbTab & "Serial #" & vbTab & _
        "Processor" & vbTab & "IOS" & vbTab & "Uptime"

' Create the master table
Call ct.CreateMasterTable(sReportFile, sHeader, bOverwrite, ct.Devices, sClientFile, "")

' Call CatTools function to create and send the report
Call ct.CreateAndSendReport(sScheduleName & " (Custom Template Report)", sHeader,
                           ct.RemoveHeader(ct.ReadFromFile(sReportFile)), sReportFile)

' Finally, clean up by deleting any temp client files
Call ct.RemoveClientResults(sClientFile)

```

#### **ct.RemoveClientResults(sClientFile)**

This sub routine removes any existing client temporary .txt files within the \ClientTemp folder, with a filename beginning with the string value of sClientFile.

It has one input parameter:

*sClientFile*            String value of the file names to find

#### **ct.MarshalThreads(ByVal sDeviceList As String, ByVal sScriptFile As String, ByVal sFunctionName As String, ByVal IMaxProcessCount As Long, ByVal sUniqueFileID As String)**

This sub routine manages the client threads used within the activity schedule.

It has five input parameters:

*sDeviceList*    Pipe (|) delimited string of devices associated with the current schedule (use the "ct.Devices" variable for this parameter)  
*sScriptFile*    String of the filename of client script file to use (specify the "ct.ScriptFileClient" variable for this parameter)  
*sFunctionName* String of the Function name within the client script file to use (normally this should be "Client")  
*IMaxProcessCount* Maximum number of client threads to use for the activity (this will be automatically restricted by your Edition of CatTools)  
*sUniqueFileID* File name prefix string for the temporary client (device) files that the activity creates in \ClientTemp within the activity

**Example:** start the client threads for a simple version report. Get the maximum client threads from the activity settings in the database.

```

Dim IMaxClientThreads

' Get the number of client threads from the activity settings
IMaxClientThreads = ct.DBScheduleGetField(ct.ScheduleNumber,
"ClientThreads")

' Start the threads

```

```
Call ct.MarshalThreads(ct.Devices, ct.ScriptFileClient, "Client",
    lMaxClientThreads, "CustomReportTemplate")
```

### **ct.DBScheduleGetField(IScheduleNumber, sFieldName)**

This function is used to get a field value from the CatTools database for a given activity. It performs the same operation as the client function `cl.DBScheduleGetField`, but can only be called from within an activity Main Script.

It has two input parameters:

*IScheduleNumber* The current schedule number as a Long  
*sFieldName* The field name in the database we want to get the value of as a String

Example: get the report filename for an activity and store the value in the string variable `sReportFile`.

```
sReportFile = ct.DBScheduleGetField(ct.ScheduleNumber, "ReportFile")
```

### **ct.RemoveHeader(sData) As String**

This function is used to remove the first line of data in a given block of data. It does so by locating the first instance of a `vbCrLf`, and removes everything up to and including it. The function returns a string with the first data line removed.

It has one input parameter:

*sData* String of the data to manipulate

## **4.5.7 Testing your custom activity**

Testing your custom activity is relatively straight forward.

Once you have the activity type (`.ini`) file, the activity [main script](#) and [client script](#) (`.txt`) files setup and have also created the device [.custom](#) file containing the device specific code for your custom activity; you simply create a new activity in CatTools, selecting your custom activity from the activity 'Type' drop-down list field, then in the 'Devices' tab you select the device which you created the `.custom` file for.

There are several testing aids available in CatTools.

### **1) Info Log messages**

Within the Info Log pane you see messages appear while an activity is running. The level of messages that appear can be filtered by the drop-down list near the bottom of the Info Log window. By selecting level **"4) Show Debug events and above"** you get to see all the [cl.log](#) messages being displayed during the running of an activity.

You should be aware that all the Info Log messages are logged to a file called `InfoLog.txt` in the root CatTools folder. This file can get very large very quickly, so can be purged by selecting the [File | Delete | Delete InfoLog.txt file](#) menu item from the CatTools File menu.

The client script function [cl.Log 4,"message"](#) is found throughout device scripts to display level 4 messages which help assist in troubleshooting device specific issues.

The most likely Info Log error (level 1 - error) you will encounter when testing your new custom activity script is:

- "Client script error: Type Mismatch: [activity name] on line: [line number]"

This error will occur if:

- the device you have selected to run your custom activity against does not have a [.custom file](#) defined for it, or
- the .custom file for the device does not contain a function which has been defined in your custom activity client script, or
- a function defined in the .custom file contains an error within the code.

If CatTools finds an associated .custom file for the device, you should see an Info Log (level 4 - debug) message of

```
"Loading custom activities for device"
```

If you do not see this message for your device (note: you must have set your Info Log pane to "**4) Show Debug events and above**"), then CatTools was unable to find a .custom file either because it did not exist, it could not be opened (check permissions), or a typo error has been made when naming the .custom file.

If you see the debug message above, then check the contents of the .custom file to ensure that the custom activity function exists, has been spelled correctly and that the code does not contain errors (maybe add additional `cl.log 4, "..."` Info Log debug messages at strategic points in your code to find where the script is failing).

If the selected device does not support the custom activity, either deselect the device in the activity setup Devices tab, or add the following code to the custom activity Function within the .custom file for the device:

```
Function [YourCustomActivityFunctionName]()
    Call cl.CatToolsNoSupport
End Function
```

Now when you run this activity, the function will be called, but the [cl.CatToolsNoSupport](#) internal CatTools function will return an Info Log message (level 2 - Warning) of:

```
"Device type: [SCRIPT_NAME constant] has not had this functionality
added yet. Skipping this device"
```

Other errors you may encounter are:

- "Client script error: Variable is undefined: '[variable name]' on line: [line number]"

This error will occur if:

- your code contains a variable which has not been declared. Ensure you **Dim** or **Redim** your variables correctly (check for typo errors in your variable declarations)
- your code contains a call or reference to an internal CatTools variable or function



(cl. or ct.), or to a class or type you have defined, but you mistyped 'cl.' or 'ct.' or the class name.

- "Client script error: Object doesn't support this property or method: '[*property or method*]' on line: [ *line number*]"

This error will occur if:

- you are making a Function call or Sub routine call within the [.custom](#) file, but the Function or Sub routine does not exist. (check for typo errors in your code)

- "Client script error: Wrong number of arguments or invalid property assignment: '[*Function or Sub name*]' on line: [ *line number*]"

This error will occur if:

- you are making a Function call or Sub routine call within the [.custom](#) file, but have supplied an incorrect number of arguments to the Function or Sub routine.

## 2) Debug log

Under the File menu of CatTools there is an entry [Enable Capture Mode](#). This turns on the logging of the conversation between CatTools and the end device and creates a disk file in the \Debug sub folder of the communications. This can be extremely useful when the script does not appear to be communicating correctly with the device after making its initial connection. The scripting mechanism waits for known prompts and issues commands to the device at these points. It eventually times out if an expected prompt fails to appear. The debug log capture file shows what the device is actually sending to CatTools, so you can adjust your code accordingly.

## 4.6 Unsupported activities for a device

Device types are updated all the time.

For an up to date list of the devices types currently available in CatTools and the activities supported for each of the device types, please see the [device matrix](#) on our [web site](#).

If there is a listing to support your device, but the activity you require is not currently supported by the device type then please contact us using the [Technical Support](#) form, providing us with as much detail as you can and if possible a PuTTY capture of the relevant command(s).

## 5 Menus

The following menu items are available from the CatTools main menu:

- [File](#)
- [View](#)
- [Options](#)
- [Interface](#)

- [Help](#)

## 5.1 File

The following menu items are available from the CatTools File menu:

- [Database](#)
- [Delete](#)
- [Enable capture mode](#)
- [Debug](#)
- [Exit](#)

### 5.1.1 Database

You can perform a number of operations on the CatTools database.

#### 5.1.1.1 Export

You can export activity and device details to flat files.

##### 5.1.1.1.1 Export devices to tab delimited file

Click on this option to export your device information to a tab-delimited file (suitable for reporting using tools like MS-Excel or similar).

This will pop up a normal file selection box, where you browse to select the folder you want the file created in, and then enter the name of the file you want to create.

##### 5.1.1.1.2 Export activities to tab delimited file

Click on this option to export your activity data to a tab-delimited file (suitable for reporting using tools like MS-Excel or similar).

This will pop up a normal file selection box, where you browse to select the folder you want the file created in, and then enter the name of the file you want to create.

#### 5.1.1.2 Import

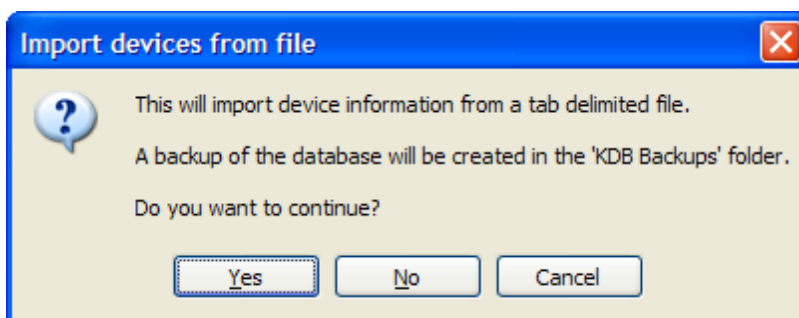
You can import activity and device details from flat files.

##### 5.1.1.2.1 Import devices from tab delimited file

This menu allows you to import devices from a tab delimited file.

The current database is backed up before the import operation occurs.

This first screen that appears will prompt you to confirm the import operation.

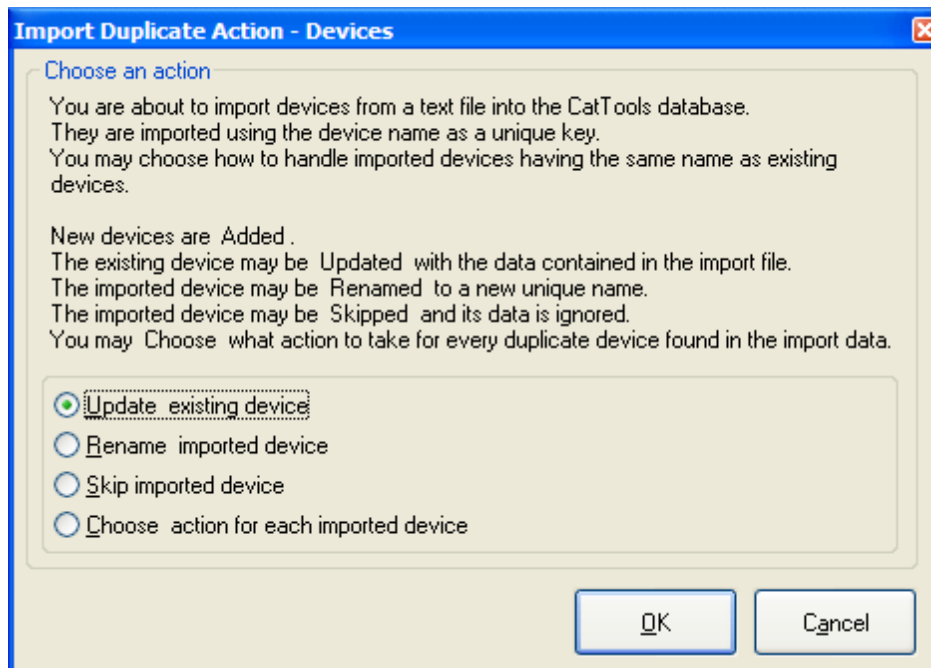


Press **Yes** to continue. Press **No** or **Cancel** to exit the import operation.

Next, you need to select the file to import from. By default, the file mask is set to "Export\_Device.txt".

Select the file to import and press **Open**.

You are now given options as to how to handle imported devices that have the same names as existing devices.

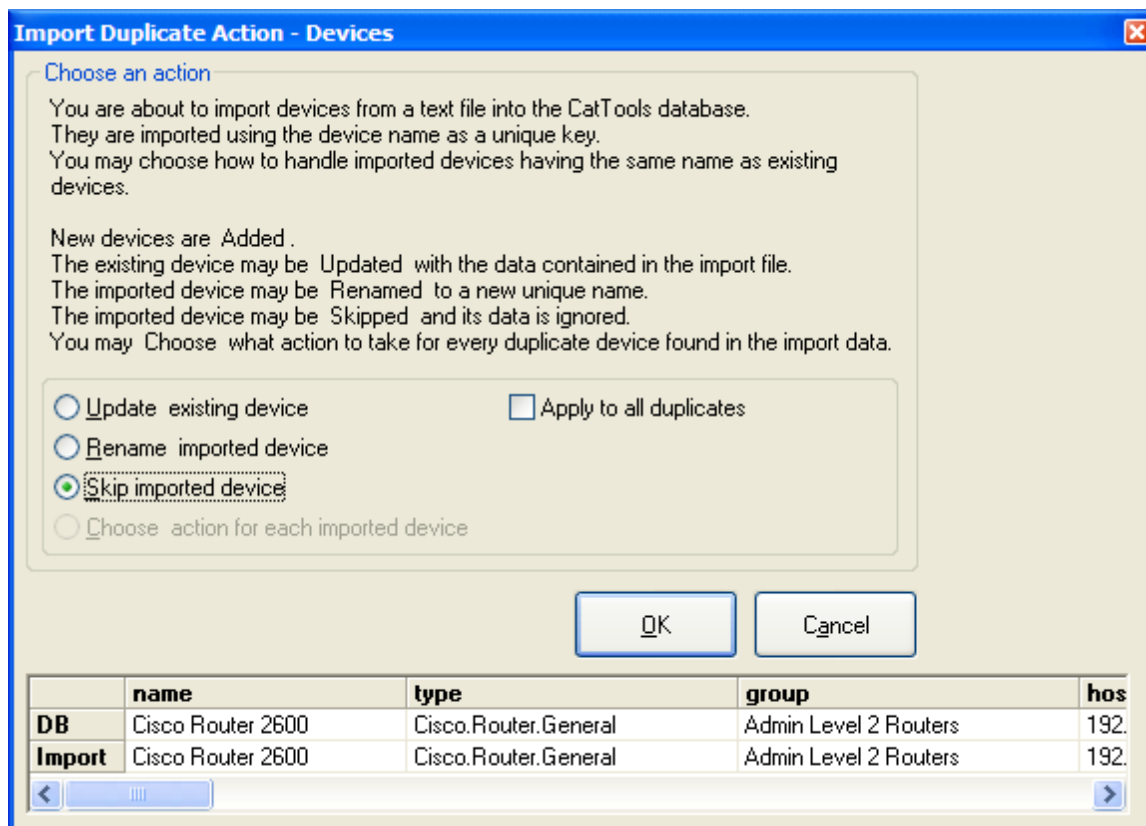


The import process always Adds a new device.

When a duplicate is found, you may elect to:

- **Update existing device** with the imported device's data. This overwrites the current device data with any fields that are found in the import data. Any fields that are not found in the import data are not changed in any way.
- **Rename imported device.** A new device is created with **\_imported\_0** appended to the current device name. The last character is a sequential digit that is incremented when more than 1 devices with the same name are encountered.
- **Skip imported device.** The imported device data is ignored.

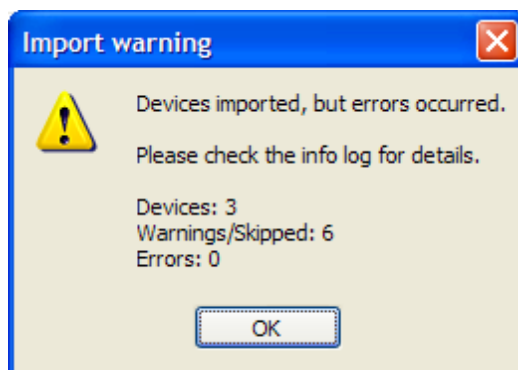
Choose action for each imported device. Any time a duplicate name is found the following dialogue is displayed:



You can view the differences between the 2 devices and choose what action to take for that particular device.

Should you wish to you can also choose to **Apply to all duplicates** subsequent. Pressing the **Cancel** button terminates the import process immediately.

When the import operation is complete, a message will appear.



This will tell you how many devices were imported from the file. If any warnings or error are shown you should check the Info Log for more details.

### Device import file format.

The file must start with a header row that defines the fields. The rows following the header are treated as data.

For example:

Name	Type	Group	HostAddress	Filename	Model	ConnectVia	Telnet	TelnetPort
Cisco Router 2600	Cisco.Router.General	Test Group	127.0.0.1	Cisco_Router_2600	Cisco 2600	Direct connect	Telnet	23

The fields can be in any order, but must follow the same naming convention as the CatTools [Export devices to tab delimited file](#) function.

At a minimum, the file must contain the following 3 fields:

- **Type** (The CatTools device type)
- **Name** (A unique name for the device)
- **HostAddress** (The IP address or hostname for the device)

Below are a list of the device fields.

- **Type**
- **Group**
- **Name**
- **HostAddress**
- **Filename** (Unique filename for device, no path or file extension required. If not specified, CatTools will derive from the **Name** field)
- **Model**
- **ConnectVia**
- **Telnet**
- **Session**
- **VTYPass**
- **ConsolePass**
- **EnablePass**
- **PrivilegeLevel**
- **AAAUsername**
- **AAAPassword**
- **SSH Username**
- **SSH Password**
- **SNMPRead**
- **SNMPWrite**
- **RequireVTY Login**
- **LoginUsesAAA**
- **EnableUsesAAA**
- **VTYPrompt**
- **ConsolePrompt**
- **EnablePrompt**
- **AAAUserPrompt**
- **AAAPassPrompt**
- **Address1**
- **Address2**
- **Address3**
- **ContactName**
- **ContactPhone**
- **ContactEmail**
- **ContactOther**
- **AlertEmail**
- **SerialNumber**
- **AssetTag**
- **Identification**
- **SerialOther**
- **ActivitySpecific1**

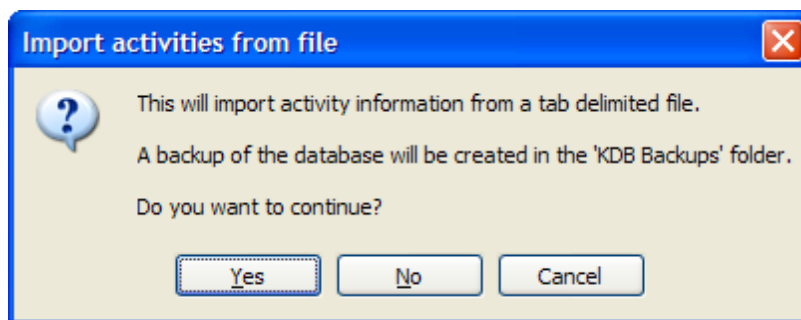
### • ActivitySpecific2

The best way to create a template to start with is to add a few example devices to CatTools first. Then use the **File | Export | Export devices to tab delimited text file** menu item. The file created will contain all the required fields you need to use.

Plain text values found in fields that are normally encrypted within the database (e.g. HostAddress, VTYPass, EnablePass SNMPRead, etc) will automatically be encrypted as part of the import process.

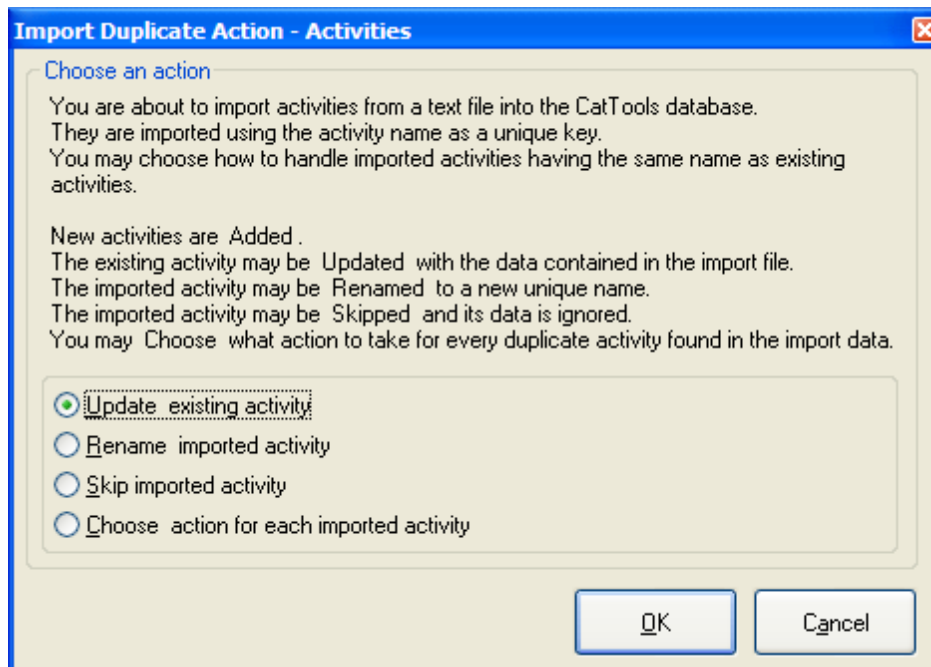
#### 5.1.1.2.2 Import schedules from tab delimited file

This allows you to import activities that have previously been exported from CatTools in tab delimited format.



If you choose to continue you see a normal file selection dialogue from which you choose the file to be imported.

You are then prompted to decide how to handle any activities with duplicate names that are found in the import file.



The Import Duplicate Action dialogue works in the identical manner as for importing [devices](#).

After the import has completed you are shown a prompt telling you whether the import was successful and how many activities were imported.

#### 5.1.1.3 Change encryption password

CatTools employs industry standard RC4 encryption to ensure that the contents of your device fields are protected from prying eyes.

As an additional security measure, you can have the program ask you for a password when it is run. This limits access to the program to those who you trust with the password.

The password you provide is used to encrypt the device fields. This means that if you forget the password, you can not use the program, and you will be unable to access the device data.

When encrypted the Application and Manager will prompt you for the password on startup. The Service will open the database and run regardless of whether a password is set or not.

Resetting the password back to a blank value will stop the program from prompting you for a password.

Even with no password set, the device database is still securely encrypted.

##### **Changing the password:**

Clicking on this menu option will pop up a normal change password box, with three fields:

1. Existing password:
2. New password:
3. Confirm new password:

Enter the existing password (blank initially), then enter the new password twice.

To reset the password back to nothing, use a blank value for the new and confirm password fields.

**DO NOT FORGET YOUR PASSWORD!**

#### 5.1.1.4 Backup current database

Clicking this menu option will back up your database files to the "\\KDB Backups" sub folder.

This is a standard folder created under the "CatTools" folder at installation.

The .kdb files will not be over-written or appended, but each new save will create a new file, and the file name will be incremented using a numeric progression:

Backup-1, Backup-2, Backup-3, etc ... ..

Up to 9 backup incarnations are created. Each time a backup is done, Backup 1 moves to Backup 2 which moves to Backup 3 etc. Backup 9 is dropped off the end and deleted.

#### 5.1.1.5 Restore database from backup

This facility allows you to get a copy of a backup database to the main CatTools folder and make it the current database.

The database is copied into the main CatTools folder and named `Restored1_KiwiDB-CatTools.kdb` where `Restored1` contains a sequence number for uniquely naming restored databases.

#### 5.1.1.6 Squeeze current database

The CatTools database may contain unused space caused by normal device and activity maintenance traffic. It is highly desirable to reclaim this unused space at regular intervals.

The Squeeze database facility enables you to reclaim any unused space in the database file. It also refreshes all indexes to the data.

To run this there must be no other CatTools users running against this database.

#### 5.1.1.7 Open other database

CatTools may open any available compatible CatTools database in any folder it has permissions to fully access. The Open other database dialogue allows you to search for and select the database you require.

Care should be taken when opening databases from previous versions of CatTools. Incompatible files will not be opened. Compatible databases that are not at the current version may be upgraded if required to the version of the CatTools application that is opening it.

#### 5.1.1.8 Create new database

This facility allows you to create a new CatTools database with no data in it.

The new database is named as in `New_KiwiDB-CatTools.kdb`.

### 5.1.2 Delete

You can delete the info log file to recover disk space.

- [Delete log file](#)

#### 5.1.2.1 Delete log file

The information log file is named "InfoLog.txt" and can be found in the main CatTools installation folder (normally `C:\Program files\CatTools\` where *n* is the version) .

To purge the log file, click on the **Delete log file** menu item.

Note: that you can set a maximum size for this file in the **Options | Setup** menu item, [Logging](#) tab.

### 5.1.3 Enable capture mode

Capture mode can be used to help diagnose device connection or device activity problems.

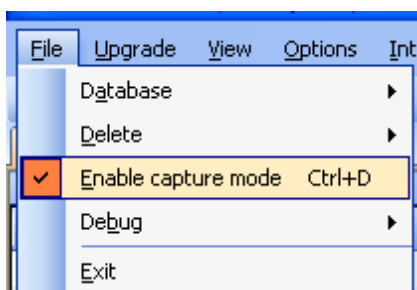
If an activity is failing for an unknown reason, technical support may ask you to '**Enable capture mode**' before attempting to run the activity. This will capture all the interaction between CatTools and the device to a log file. This file can then be sent to technical support to assist in troubleshooting the problem.

The **Enable capture mode** menu option functions like a tick box.

The Default is Off (unticked).

When you click on this menu option then capture mode is immediately enabled. This is shown by the presence of a tick to the left of the menu option next time you look at the **File** menu.





When you click on this option again then capture mode is immediately disabled and the tick mark is removed.

The captured data is stored in the '\\Debug' sub-folder and the file name is dependent on the name of the captured device.

For example, a device called 2950 running on client thread 1 will create a file called: DebugLog-1-2950.txt

Device captures can be included when [creating the diagnostics file for Technical Support](#).

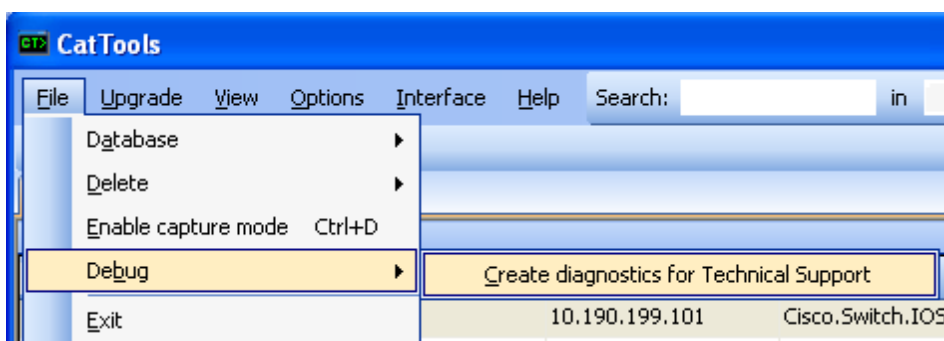
## 5.1.4 Debug

You can create debugging information to send to Technical Support for debugging problems.

### 5.1.4.1 Create diagnostics for Technical Support

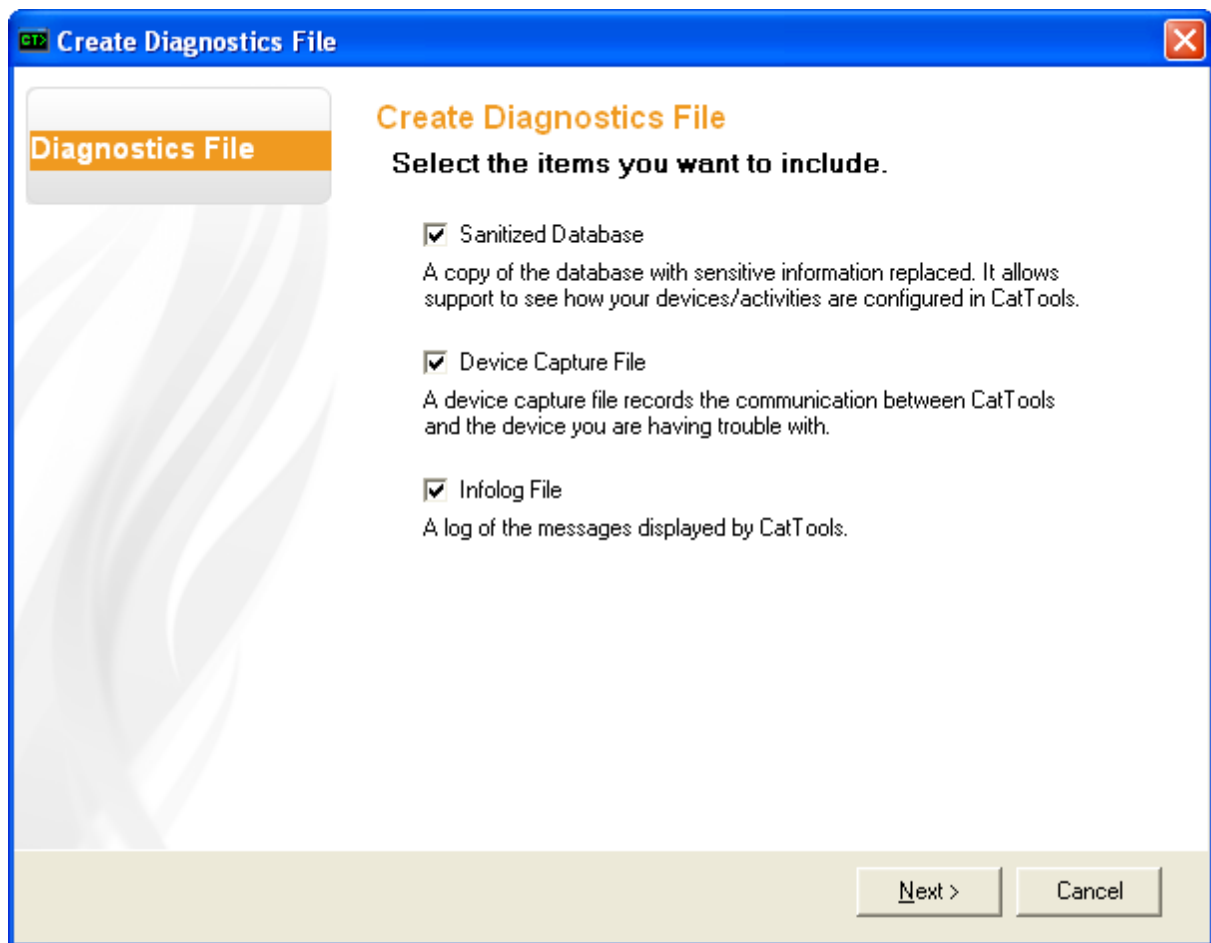
To troubleshooting issues with CatTools, Technical Support may ask you to create a diagnostics (zip) file.

To open the 'Create Diagnostics File' helper screens, click the **File > Debug > Create diagnostics for Technical Support** menu item.



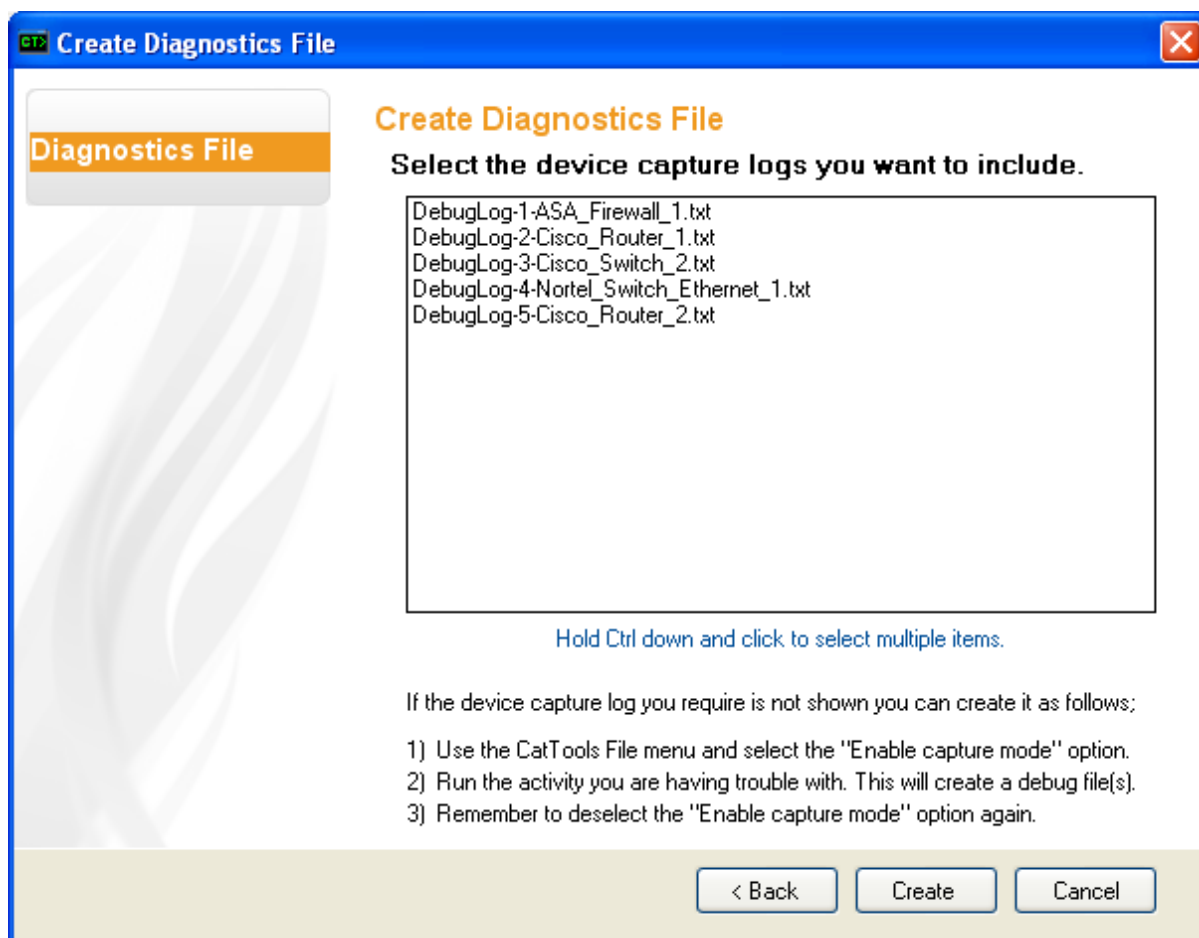
You will then be presented with an options screen that allows you to choose which log/settings files will be included in the diagnostics zip file.

By default all three options are ticked as this will help Technical Support troubleshoot the in general should be the norm.

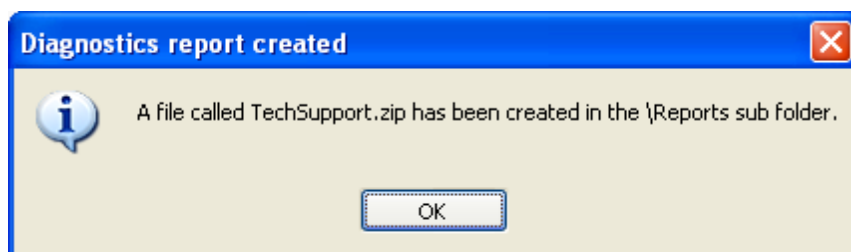


**\*NOTE:** If you are sending a *Device Capture File* to troubleshoot a device issue; then you will first (before creating the diagnostics file) need to [enable capture mode](#) and run the activity to generate the device capture file(s).

If the *Device Capture File* option is selected, a 'Next' button will be available to take you to the DebugLog selection screen so you can choose which device capture files to include in the diagnostics zip file.



When the 'Create' button is clicked, you will receive a confirmation message that the diagnostics zip file has been created.



### 5.1.5 Exit

Close and exit the CatTools program.

**Important note:**

If you are running the Service edition, the CatTools service will continue run as a service in the background.  
The CatTools service can be stopped using the Windows 'Services' administration tool from the Control Panel.

## 5.2 View

The view menu lets you look at various CatTools data files using the notepad editor.

### 5.2.1 View Reports folder

Click on the **View | View Reports folder** menu option to view the Reports folder using Windows Explorer.

The reports from scheduled activities are stored in the \Reports sub folder. Clicking on this menu will open Windows explorer to the Reports sub folder so you can browse the contents.

You can view the reports with MS-Excel, Log Viewer or use the internal viewer in CatTools from the Reports tab.

### 5.2.2 View Captured Data folder

Click on the **View | View Captured Data folder** menu option to view captured data using Windows Explorer.

The data captured from scheduled activities is stored in the \Captured Data sub folder. Clicking on this menu will open Windows explorer to the Reports sub folder so you can browse the contents.

You can view the reports with MS-Excel, Log Viewer or use the internal viewer in CatTools from the Reports tab.

### 5.2.3 View Configs folder

Click on the **View | View Configs folder** menu option to view the configs folder using Windows Explorer.

The device configurations from the [Device.Backup.Running Config](#) configuration activity are stored in the \Configs sub folder. Clicking on this menu will open Windows explorer to the Configs sub folder so you can browse the contents.

You can view the configs with notepad or wordpad etc.

### 5.2.4 View Email log

Click on the **View | View Email log** menu option to view CatTools email activity log file using notepad.exe.

### 5.2.5 View Activity log

Click on the **View | View Activity log** menu option to view CatTools activity summary log file using notepad.exe.

## 5.2.6 View Info log

Click on the **View | View Info log** menu option to view CatTools info log file using notepad.exe.

## 5.3 Options

This enables you to change the setup options.

### 5.3.1 Setup

This menu item displays a form that enables you to set the general properties of CatTools.

The [Email](#) tab allows you to set all the emailing properties.

The [Logging](#) tab enables you to set CatTools to send syslog messages to a syslog server and provides control over information and activity logging functions.

The [Misc](#) tab allows you to set miscellaneous settings.

The [TFTP Server](#) tab contains the settings for controlling the CatTools TFTP server.

The [DNS Resolver](#) tab contains the settings for resolving IP Addresses in ARP reports.

#### 5.3.1.1 Email

The Email tab allows you to set all the emailing properties.

This is further subdivided into:

- [General Options](#)
- [Primary SMTP Server Setup](#)
- [Secondary SMTP Server Setup](#)

##### 5.3.1.1.1 General Options

#### **Send To: (e-mail notification)**

**Errors:** The e-mail address you want CatTools to send error notifications. All the errors that occur during a scheduled activity are collected and sent as a single error alert. The error notification will tell you what error occurred and on which devices.

More than one e-mail address can be specified by using a comma (,) or a semi colon (;) to delimit the addresses.

For example: joe@company.com,mary@company.com,fred@company.com

A check box gives you the option to send error notifications only when in timer mode.

**Reports/Stats:** The e-mail address you want CatTools to send reports and statistics. Most scheduled activities produce a report or statistics table of some kind.

More than one e-mail address can be specified by using a comma (,) or a semi colon (;) to delimit the addresses.

For example: joe@company.com,mary@company.com,fred@company.com

A check box gives you the option to send reports only when in timer mode.

#### **From Address:**

Enter the "From Address" you want to appear in e-mail notifications from CatTools. This makes it easy for you to manage incoming e-mail notifications at your e-mail client, particularly SPAM filters. We suggest that you use a fully qualified real e-mail address. Some SMTP servers only send e-mail on behalf of addresses in their domain. For example, if your mail domain is mycompany.com, use a from e-mail address of "CatTools@mycompany.com" rather than just

"CatTools"

**Logging Options:**

A check box allows you to select whether to keep a log of email activity. The log normally contains informational and error level messages.

A check box allow you to set the logging message level to verbose. This is useful when there is a problem with email that requires you to see the complete email message traffic with your email server. It is not recommended that you set this option on except as needed due to the volume of data recorded.

Two buttons allow you to View the email log, and to Delete it.

**Send Interval:** You can set the email sending interval. This controls the seconds the mailer waits before sending a batch of email messages. A value of 20 means the mailer wakes up every 20 seconds to see if there is any messages to send.

**Messages / Interval:** This option sets the number of email messages the mailer sends to the mail server during one sending session. This allows you to control the amount of email traffic to your email server if you require it.

**Send Retries:** This option set the number of times the mailer will attempt to send an email that may have errors before it flags the message as not deliverable.

**Test Email:**

Click the button to test the current e-mail settings against both mail servers. Email messages to the Errors and Reports/Stats addresses previously entered are sent.

#### 5.3.1.1.2 Primary Mail Server Properties

**Use this server:**

This check option allows you to enable or disable the use of this server when CatTools sends email.

**Server Name / Address:**

Enter the name or IP address of the target SMTP server where you want CatTools to send e-mail notifications.

**SMTP Port:**

Enter the port used by the target SMTP server. It defaults to 25.

**Timeout:**

This is the timeout value for CatTools to wait when trying to connect to the target SMTP server, in seconds. It defaults to 30. If you have a bad connection to your mail server, we recommend that you use a local SMTP relay application. We recommend MailDirect from: <http://www.ocloudsoft.com>.

**Authentication:**

Some SMTP servers require you to authenticate before sending your e-mail, especially remote ISPs or relay servers. Most SMTP servers do not require authentication and so you can leave these values blank. Remember not to confuse SMTP sending with POP3 receiving (which always requires username/password authentication).

**Type:** CatTools can be set to use no authentication, SMTP AUTH, or POP before send.

**Username:** The username for authentication.

**Password:** The password for authentication

**POP Server Name:** The name or address of the POP server to use for authentication.

**POP Port:** The port number of the POP server, normally 110.

**Test Email:**

Click the button to test the current e-mail settings against the primary mail server. Email messages to the Errors and Reports/Stats addresses are sent.

#### 5.3.1.1.3 Secondary Mail Server Properties

CatTools can have an alternate mail server set up that is used should the primary server fail to connect.

If the mailer is unable to connect to the primary server when it attempts to send messages, it immediately tries to connect to the secondary server.

If a connection is lost while the mailer is sending mail to the primary server, the mailer does not immediately retry the connection. Instead it waits for the designated send interval and tries to connect to the primary server again.

If the primary server is not selected for use, the mailer always uses the secondary server.

**Use this server:**

This check option allows you to enable or disable the use of this server when CatTools sends email.

**Server Name / Address:**

Enter the name or IP address of the target SMTP server where you want CatTools to send e-mail notifications.

**SMTP Port:**

Enter the port used by the target SMTP server. It defaults to 25.

**Timeout:**

This is the timeout value for CatTools to wait when trying to connect to the target SMTP server, in seconds. It defaults to 30. If you have a bad connection to your mail server, we suggest that you use a local SMTP relay application. We recommend MailDirect from: <http://www.ocloudsoft.com>.

**Authentication:**

Some SMTP servers require you to authenticate before sending your e-mail, especially remote ISPs or relay servers. Most SMTP servers do not require authentication and so you can leave these values blank. Remember not to confuse SMTP sending with POP3 receiving (which always requires username/password authentication).

**Type:** CatTools can be set to use no authentication, SMTP AUTH, or POP before send.

**Username:** The username for authentication.

**Password:** The password for authentication

**POP Server Name:** The name or address of the POP server to use for authentication.

**POP Port:** The port number of the POP server, normally 110.

**Test Email:**

Click the button to test the current e-mail settings against the secondary mail server. Email messages to the Errors and Reports/Stats addresses are sent.

### 5.3.1.2 Logging

**Syslog:**

**Send logs to Syslog Server:**

Tick box to enable sending of log messages to syslog server. Default is Off.

**Syslog Server:**

Name or IP Address of target syslog server, that you want your logs sent to.

**Protocol:**

Drop-Down menu option; UDP or TCP. Default is UDP.

**Port:**

May be set to any valid port number. UDP default is 514.

**Facility:**

Drop-Down menu option to select Facility setting for outgoing syslog messages. Default is Local0.

**Logging Level:**

Drop-Down menu option to select what level of log messages are forwarded to the target syslog server. Default is "Info and above".

**Test Syslog:**

Click button to send a test message to the target syslog server, using the options selected above. Verifies your configuration and network connection.

**Info log:**

**Log info log messages to local log file:**

Tick the box to enable info log messages to be logged to a local log file. The default is On. The log file created is named "InfoLog.txt" and can be found in the CatTools install folder

**Logging level:**

This drop-Down menu lets you select what level of messages are to be logged to local file. The default is "Debug and above".

**Limit File size:**

This file can become rather large if the logging of debug messages is enabled. You can purge the log file by using the File | Delete | Delete log file menu or by checking this option you can limit the size of the info log to the size in Mb specified in the size field. When the log file gets to this size, it is closed, renamed to infolog\_prev.txt, and a new log file is created. The previous log file is kept in case there is a requirement to go back in the log further than what is contained in the current log. This previous log is deleted before a current log that has reached maximum size is renamed.

The effect of having this option turned on is that there are two log files, a current and a previous.

If the option is set off, the log file grows indefinitely.



**Activity log:****Log Activity summary to local log file:**

Tick the box to enable the CatTools activity summary to be logged to a local log file. The default is On.

The log file created is named "Activitylog.txt" and can be found in the CatTools install folder.

**Limit File size:**

Checking this allows you to limit the size of the activity log to the size in Mb specified in the size field. When the log file gets to this size, it is closed, renamed to activitylog\_prev.txt, and a new log file is created. The previous log file is kept in case there is a requirement to go back in the log further than what is contained in the current log. This previous log is deleted before a current log that has reached maximum size is renamed.

The effect of having this option turned on is that there are two log files, a current and a previous.

If the option is set off, the log file grows indefinitely.

**5.3.1.3 Misc****Telnet client:**

Enter the path and file-name of the local telnet client. Default is "Telnet.exe"

When using SecureCRT as your external telnet client, you will need to add the /TELNET switch to the command line.

For example: `C:\Program Files\SecureCRT\SecureCRT.EXE /TELNET`

**SSH client:**

Enter the path and file-name of the preferred local SSH client. By default, there is no SSH client specified.

NOTE: for PuTTY.exe users. You can now specify PuTTY for both the Telnet and the SSH client. When CatTools detects PuTTY has been specified, the following command line switches are used to connect:

- [protocol Telnet or SSH. Derived from value set in 'Method' field of device setup form. If any SSH value is set, then the protocol used will be 'SSH']
- **P** [port specified in 'Port' field of device setup form]

Example: `c:\Putty.exe -SSH -P 22 HostAddress`

**Clear log window before starting each activity:**

Tick this box to enable clearing of the Info log window (See "Info log" Tab) before each activity starts. The default is On.

**Clear error count before starting each activity:**

Tick this box to enable clearing of the error count display (See Status Line of main window) before each activity starts. The default is On.

**Automatically enable timer mode:**

Tick this box to have the timer automatically set on the specified number of seconds after the service starts.

**Hide password change report passwords:**

Tick this box to hide the passwords reported by the [Device.Update.Password](#) activity report.

**Show current database name above grids**

Tick this box to show the full path of the current CatTools database

**Terminate orphaned activity running longer than *n* minutes**

Set the number of minutes to allow an activity to wait to terminate after which it will be forcibly terminated. The default is 0 minutes, meaning no limit is set. The maximum should be set less than 2880 minutes.

This should only be used in a situation where you have a problem with an activity never terminating. This may be caused by factors such as a logic loop or communications that are stalled for whatever reason. The timing only starts when say all bar one device sessions have completed, and the last session appears hung up.

When this is activated and the time limit is detected, CatTools will attempt to terminate the activity by normal means. If it is unable to terminate the activity by normal means, the CatTools\_Client.exe application, which runs the activities, is forcibly terminated. CatTools then waits for about 30 seconds to enable the system to clean up from the termination, and restarts the scheduler.

**HTML report maximum row limit**

Set the number of output rows for HTML reports. The default is 5000 rows. The minimum is 1 row and the absolute maximum is 50000 rows.

The maximum rows is limited to 50000 as it is unlikely you would want to scroll through an HTML file of more than 50000 rows. If all the report data is required, you should use the corresponding text format file (\*.txt).

**When exiting the Manager should:**

This option is available when CatTools is installed as a Service.

Since most changes need to be done when the timer is offline this feature provides the user with a reminder to start the timer again when the Manager is closed down. In this way you can ensure that your activities will run even if you have forgotten to restart the timer yourself.

*Options include:*

Ignore the timer state: No action will be taken.

Set the timer to ON: The timer will automatically be set to ON when you exit the Manager.

Set the timer to OFF: The timer will automatically be set to OFF when you exit the Manager.

PROMPT if the timer is off: A message box will ask you what you want to do when you close down the Manager if the timer is not running.

**5.3.1.4 TFTP Server**

The TFTP Server tab allows you to set TFTP Server properties.

This is further subdivided into:

- [General Options](#)
- [Security Options](#)
- [Scripting Options](#)

**5.3.1.4.1 General Options****Automatically start TFTP server on program startup:**

Start the server automatically when CatTools starts.

**TFTP Port:**

Enter the port number the server listens on, normally 69.

**TFTP Bindto:**

Enter the IP address of a network interface card to bind to or leave it blank for the only or default card.

**Upload folder:**

This is the name of the folder that contains files to be uploaded to devices from CatTools.

**Download folder:**

This is the name of the folder that contains files downloaded from devices by CatTools.

**Allow overwrite of existing files:**

Tick this box to allow files currently in the folder to be overwritten by files of the same name.

**Allow automatic creation of sub folders:**

Tick this box to allow subfolders to be created automatically during a download

**Remove completed sessions from display after:**

Remove the messages created from a TFTP session from the display tab after the specified number of seconds.

**Maximum file size that can be created:**

Set the maximum file size in MB that the server can create. The default is 20MB. TFTP clients may also have a limit to file size they can send. There appears to be a known limitation in some TFTP clients to a file size of about 32MB (which corresponds to using the standard protocol block size of 512 bytes).

However if your network and devices are capable of handling larger block sizes, then you may be able to create much larger files.

There are no options in CatTools to specify a block size to use; block sizes are negotiated between the TFTP Client and Server.

#### 5.3.1.4.2 Security Options

**Global Read**

If this is unchecked then nothing can be read from the TFTP server.

**Global Write**

If this is unchecked then nothing can be written to the TFTP server.

**Use Access Lists**

If this is checked then access list will be used to ascertain the read and write permissions for the IP address in question.

Note that access lists are subordinate to Global Read and Global Write.

**Access List Usage**

There are five columns in the access list table;

**Inc|Ex IP Range** – This determines whether the IP range specified is to have the Read and Write options applied to it (Include), or is to be exempt (excluded) from the Read and Write options which will then be applied to all addresses outside of the range.

Exclude is useful for allowing only a limited range of IP's to be read or written to because everything outside of the excluded range will have the read/write settings applied to them.

**Start IP / End IP** – Define the IP address range to be used.

**Read/Write** – When ticked Reading/Writing is allowed, unchecked Reading/Writing is not allowed.

The rows of the access list are examined from the top down until a match is found.

### **Access List Examples**

Using the access list below, we will look at some examples.

Incl Ex	IP Range	Start IP	End IP	Read	Write
Exclude		192.168.1.1	192.168.1.100		
Include		192.168.1.60	192.168.1.60		

The first line in the access list is an exclude so all addresses outside of the range 192.168.1.1 – 192.168.1.100 will have the read and write properties applied to them. So in this case anything outside of the excluded range will not be able to read or write to or from the TFTP server.

So attempts to read or write from address 192.168.1.105 would fail.  
Attempts to read or write from address 192.168.1.98 would succeed.

The second line in the access list is an include, so the read and write settings will be applied to addresses included in this range, which in this case is a single IP address.

So attempts to read or write from 192.168.1.60 would fail.

Let us look at the steps that would be traversed if we tried to write to the TFTP server from address 192.168.1.60

- 1) First 'Global Write' is examined, if it is checked then the process continues.
- 2) 'Use access lists' is examined, if it is checked then the access list will be checked.
- 3) The first row in the access list is examined. 192.168.1.60 is within the excluded range and consequently the read or write settings do not apply to it so the process continues to the second row of the access lists.
- 4) The address 192.168.1.60 is included in the range specified in this row so the write settings are applied, which in this case is to not allow writing to the TFTP server.

#### 5.3.1.4.3 Scripting Options

This pane allows you to specify vbscripts that will be run before and/or after a TFTP session.

Use the tick boxes to determine whether a script will be run and enter the full path to the script in the appropriate text box.

### **Timing Issues**

It's important to note that you only have 1 second to get everything done in the pre-session external script and return a value otherwise the TFTP session times out, and retries.

### **Scripting Information**

Your script must start with ***Function Main(session)*** where session is the TFTP session being passed in.

If the script is successful it should return a value of True which will allow the TFTP session to continue.

Returning a value of False will cause the TFTP session to abort.

For security reasons if the script can not be loaded or if the script fails then the TFTP session will be aborted.

You can access the following properties of the session;

Session.CreatTime - The time the session was started by the host. (Actually represented as the number of seconds from midnight 1/1/1970 UTC)  
 Session.FileName - The name of the file to be written.  
 Session.FileSize - The size of the file on the server for a 'Get' operation. (Will be zero for a 'Put' operation.)  
 Session.Host - The IP address of the remote client represented as a long. (Converted from an 8 digit hex value.)  
 Session.HostS - The IP address of the remote client represented as a string. i.e. 127.0.0.1  
 Session.Port - The TFTP Port on the remote client.  
 Session.TransferSize - The number of bytes transferred so far.  
 Note in the pre-session script this will be zero and in the post session script this should be the whole file size.  
 Session.Type - Upload for a Put. Download for a Get.

### Example Scripts:

The script below checks the host address of the remote client issuing a 'Put'. If it matches a certain address then the filename is changed.

#### Rename of the file based on host address;

```
Function Main(session)
  Main= False
  With session
    If .HostS = "192.168.1.60" Then
      .FileName = "NewFileName.txt"
    End If
  End With
  Main= True
End Function
```

#### Reject a file based on the time of day;

```
Function Main(session)
  Dim CreateHour
  Main= False
  With session
    'convert the create time from the unix format to the current time
    CreateHour = DateAdd("s", session.CreateTime, DateSerial(1970, 1, 1))
    'convert to 24 hour format
    CreateHour = FormatDateTime(CreateHour, vbShortTime)
    'seperate out the hour value only
    CreateHour = DatePart("h", currentTime)
    Select Case CreateHour
      'reject files created two hours either side of midnight
      Case 1,22,23,24
        'accept files created at any other time
        Main=False
      Case Else
        Main=True
    End Select
  End With
End Function
```

\* **Please Note;** At the moment there is a bug where if a script is triggered in the pre-session

event that renames the filename e.g.. FileName = "NewTestFileName.txt" The file is put into the TFTP folder with the filename correctly changed to NewTestFileName.txt but the existing 'Test.txt' file in the TFTP folder is deleted.

#### 5.3.1.5 DNS Resolver

DNS Resolver resolves the IP addresses in ARP reports based on the server settings below.

**Note:** The **Resolve IP addresses to Hostnames** check box must be checked.

- 1) Select either **Resolve using known DNS server(s)** or **Resolve by auto detection of DNS server(s)**.
- 2) By default there is a list of internal IP address ranges listed (10.x.x.x , 192.168.x.x , 169.254.x.x , 172.16.x.x , 172.17.x.x , 172.18.x.x , 172.19.x.x , 172.20.x.x , 172.21.x.x , 172.22.x.x , 172.23.x.x , 172.24.x.x , 172.25.x.x , 172.26.x.x , 172.27.x.x , 172.28.x.x , 172.29.x.x , 172.30.x.x , 172.31.x.x)
- 3) To add more internal IP address ranges, click **Add**. You can remove any added ranges by selecting the range and then clicking the **Remove** button.
- 4) Select the NetBIOS/DNS Server to resolve.
- 5) Enter the primary and alternate DNS server IP addresses for both internal and external DNS servers
- 6) Specify the days to flush DNS cache.
- 7) Specify DNS Query time out.
- 8) **Save** the changes.

#### 5.3.2 Device Wizard

This will open the Device Wizard which will help guide you through the addition of new devices to the CatTools database.

(Alternatively you can add new devices from the [Devices](#) pane.)

#### 5.3.3 Activities Wizard

This will open the Activities Wizard which will help guide you through the addition of new activities to the CatTools database.

(Alternatively you can add new activities from the [Activities](#) pane.)

### 5.4 Interface

This enables you to change the [Panels](#) and [Themes](#) interface options.

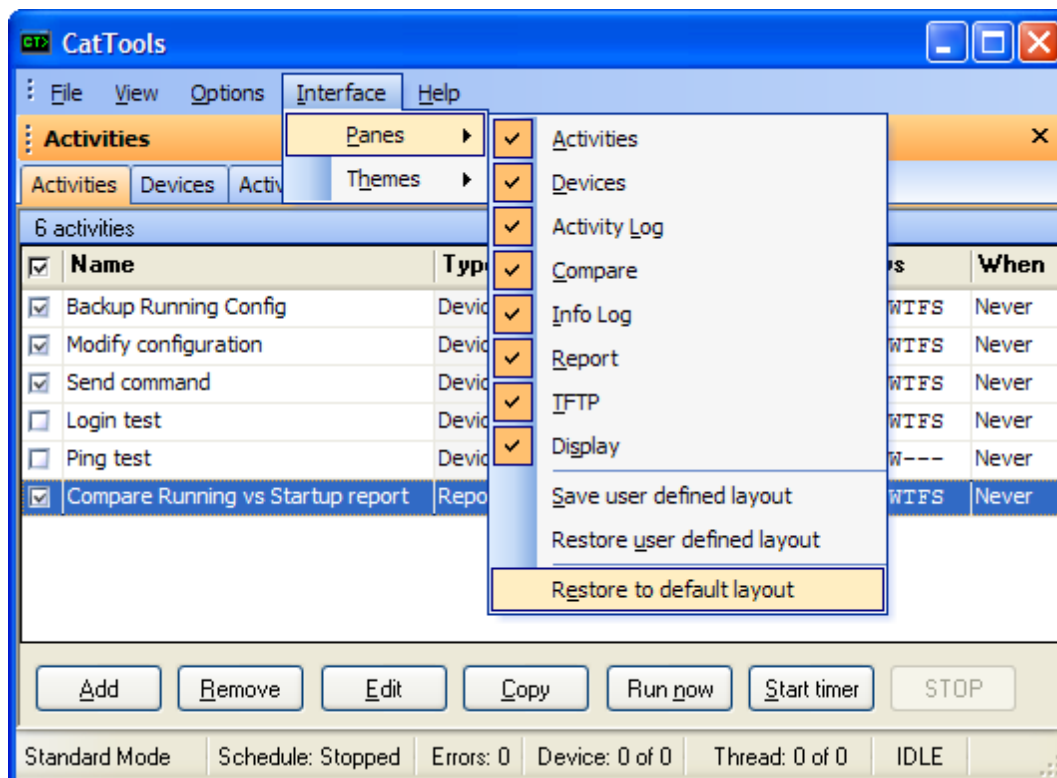
#### 5.4.1 Panels

There are a number of options under the **Interface | Panels** menu item.

- Panels can be turned on and off by selecting or deselecting them from the panels list.
- You can save your current pane layout by clicking the **Save user defined layout** menu

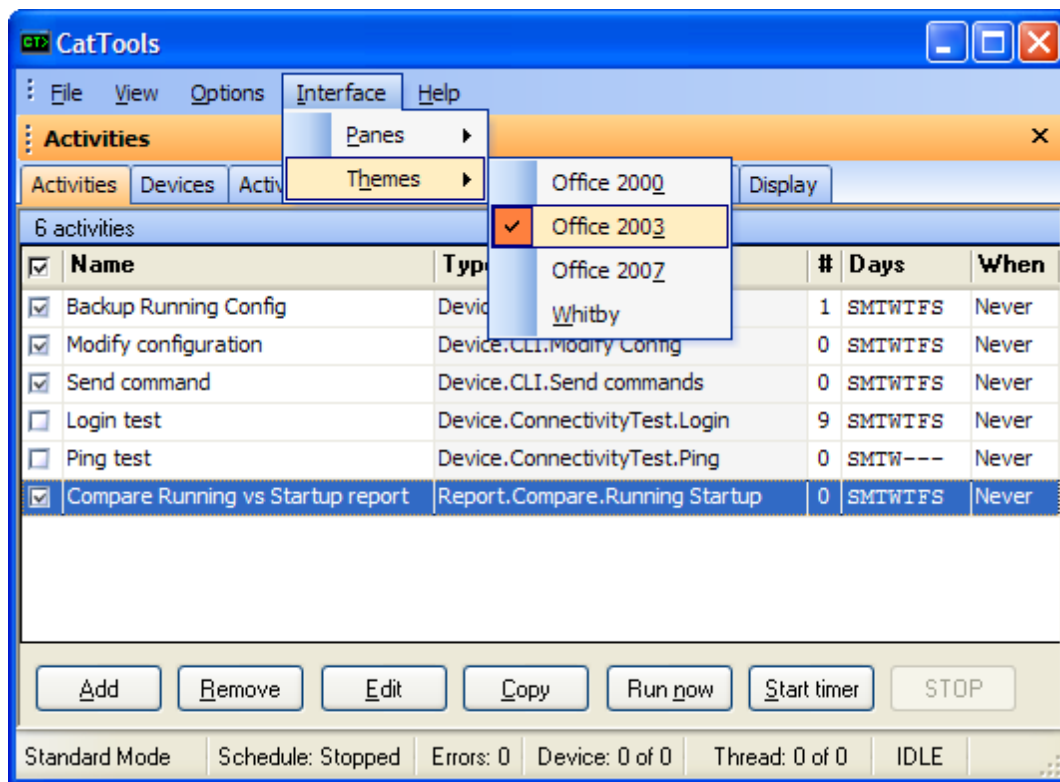
item. (Note: CatTools automatically saves your current pane layout when you exit the application)

- You can restore your previously saved pane layout by clicking the **Restore user defined layout** menu item.
- You can restore the CatTools default pane layout by selecting the **Restore to default layout** menu item.



## 5.4.2 Themes

You can change the theme of CatTools by clicking the **Interface | Themes** menu item and selecting an option from the list of themes.



## 5.5 Help

The help menu enables you to view help, get or set registration details, and visit the web site.

### 5.5.1 Contents

Displays the Contents of the Help file.

### 5.5.2 Online FAQ

Opens a browser window and views the CatTools Online Frequently Asked Questions page.

### 5.5.3 Purchase CatTools

Opens a browser window to the online purchase page. Various off-line methods or payment are also available.

### 5.5.4 Enter registration details

1. Enter Registration details page will lead you to the Kiwi CatTools License application.
2. Activation keys can be found by logging in to the customer portal <http://www.solarwinds.com/customerportal>
3. Complete the Activation wizard to upgrade your license.

### 5.5.5 About

Opens the "About" window, showing the program version number, registration details, and support information.



When contacting technical support about a problem, please make note of your version and build number so we can assist you better.

## 6 Panes

Version 3.1.1 of CatTools saw the introduction of the docking panes interface. This replaced the tabbed layout of earlier versions.

The panes interface is essentially the same as the tabbed layout but it is customisable allowing the user to hide and show the panes that matter to them and to have two or more panes viewable at the same time.

- [Overview](#)
- [Devices pane](#)
- [Activities pane](#)
- [Activity Log pane](#)
- [Compare pane](#)
- [Info Log pane](#)
- [Report pane](#)
- [TFTP pane](#)
- [Display pane](#)
- [Mail pane](#)

### 6.1 Overview

#### **Docking panes**

There are a number of different docking options.

For example, you can dock panes at the top, bottom, left or right of the screen, or dock them to the top, bottom, left or right of other panes.

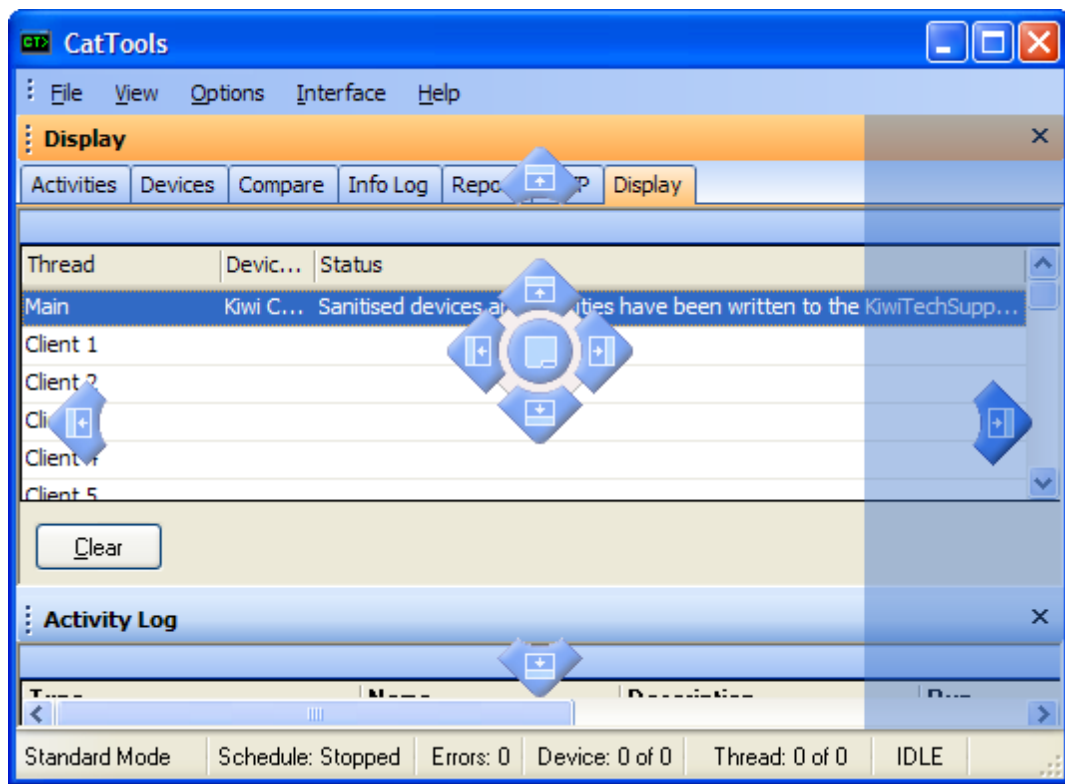
To dock a new pane, do the following:

1. Click on the pane you wish to dock and while holding the mouse button down drag it into a pane. The docking pane locator will appear on screen.
2. While holding down the mouse button, move the cursor over one of the docking pane locator "stickers". The area where the pane will be docked will be highlighted (as below).
3. Release the mouse to dock the pane in that location.

The example below shows how to dock the Display pane to the right of the screen by using the right-most locator sticker.

The pane is docked to the right of **all** the existing panes.

If you wanted to dock the Display pane to the right of just the top-most pane (leaving the Activity Log pane to span the width of the window), then you would drop the pane on the right-hand sticker within the centre cluster.



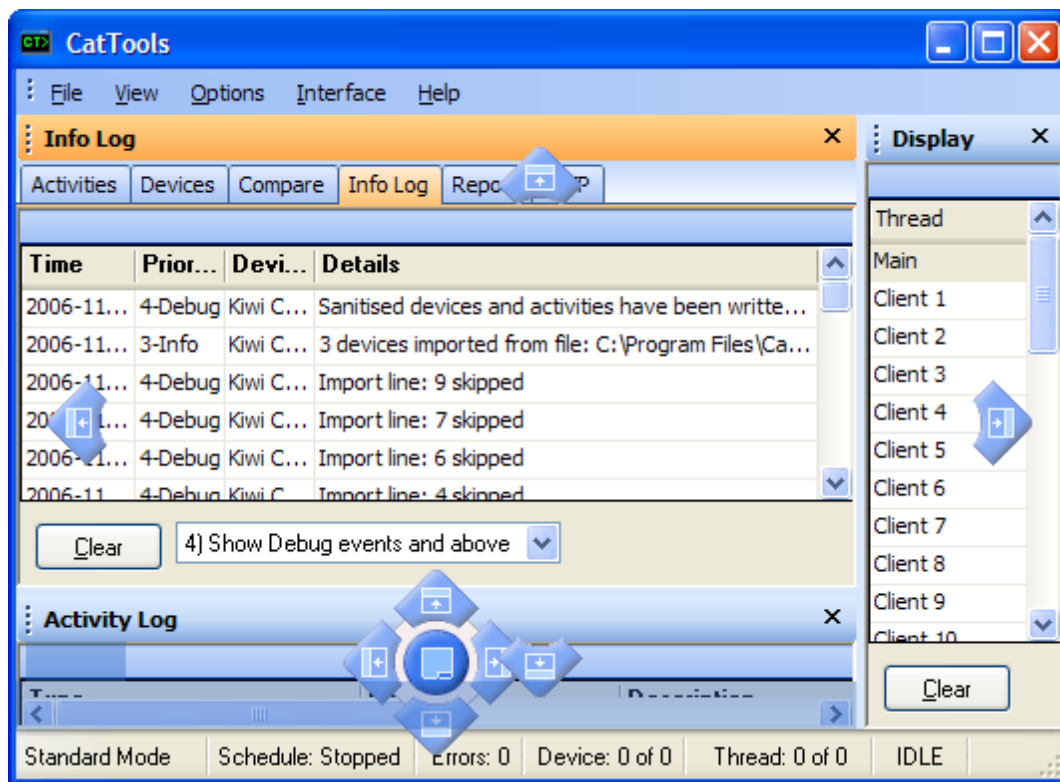
Band1t1

### **Moving/grouping panes**

To move a pane from one group of panes to another, do the following:

1. Click on the pane you wish to move and while holding the mouse button down drag it into the pane you want to group it to. A docking pane locator will appear on screen within the desired pane.
2. While holding down the mouse button, move the cursor over the centre-most locator sticker for that pane. The whole pane will be highlighted (as below).
3. Release the mouse to drop the pane into that pane group.

The example below shows how to move the Info Log pane from the top pane group and group it with the Activity Log in the lower pane.



### Rearranging and saving your layout

You can rearrange your pane layout easily by moving the mouse cursor over the edge of a pane until the "splitter" cursor appears.

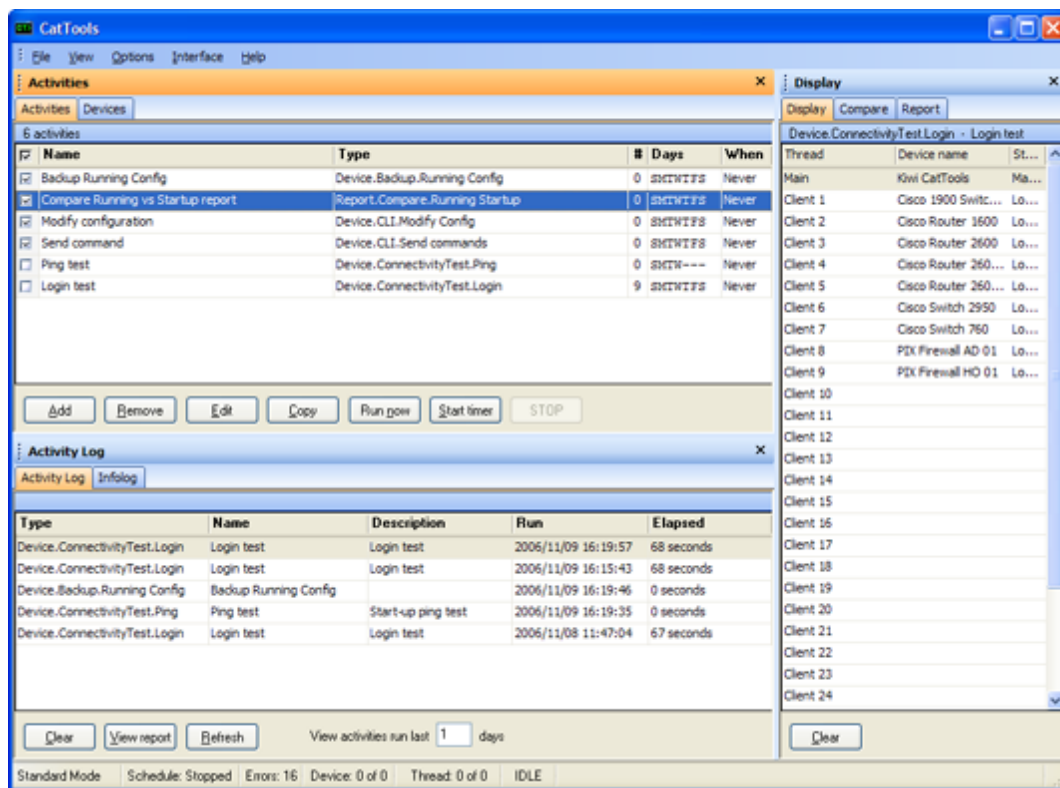
Then, by holding the mouse button down, move the cursor to where you want the edge of the pane to go to and release.

Once you have decided on your layout, you can save your changes by clicking on the **Interface | Panes | Save user defined layout** menu item.

See [Interface](#) for further details.

CatTools will also automatically save your changes for you when you exit the application.

Example of a resized window with multiple pane groups.



### Resetting pane to default

To reset the panes back to the CatTools default, click on the **Interface | Panes | Restore to default layout** menu item.

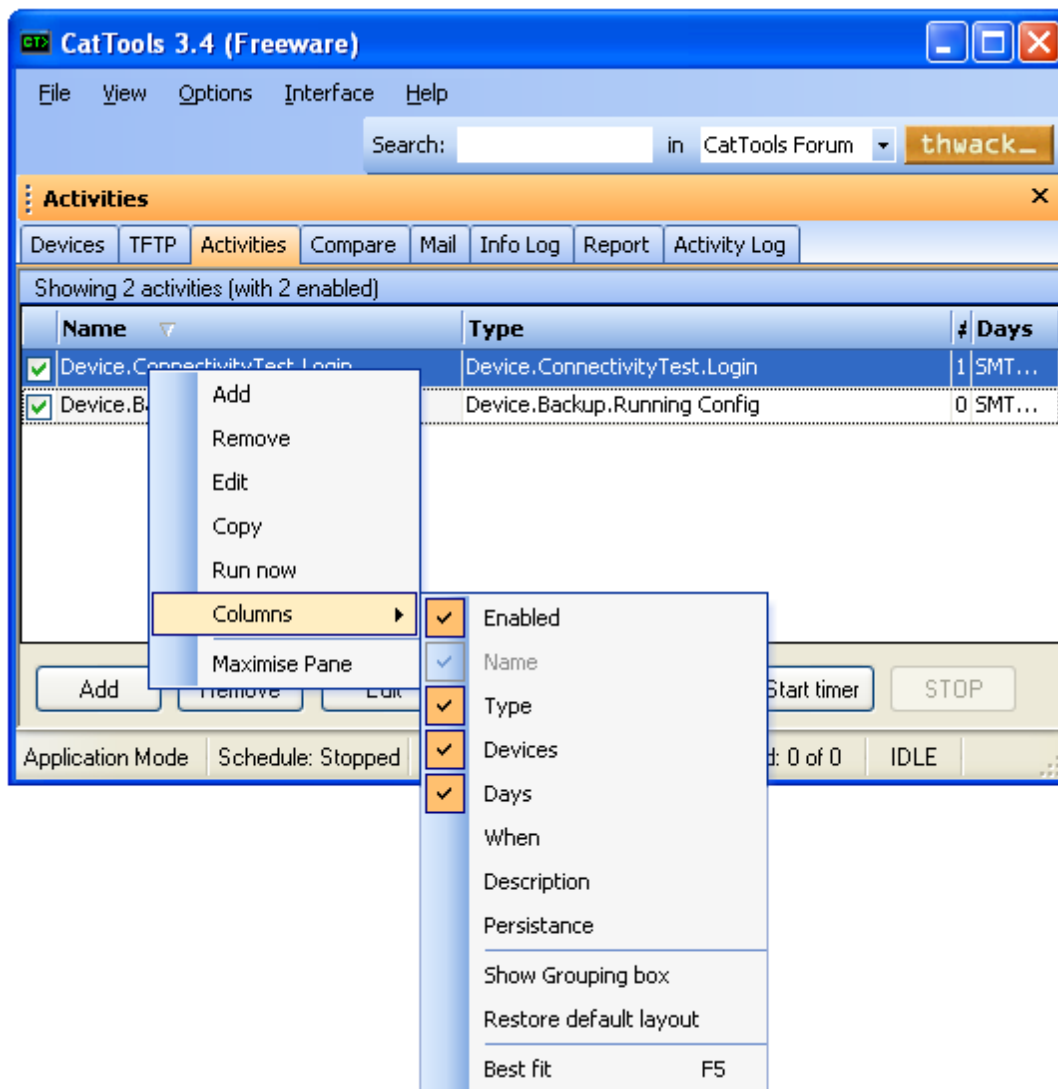
See [Interface](#) for further details.

### Right-Click short menu.

The functions of the buttons at the bottom of the tabs can also be accessed via the shortcut menu by selecting a row from within a pane and clicking the right mouse button.

The shortcut menu also includes other items which are specific to the pane in which you are currently in. For example, on the Activities pane, there is an item called "Columns" which contains sub items to show and hide data columns relating to activities; Show and Hide the grouping box which allows you to group your activities by data columns, and restore the pane back to its default layout.

You can maximise the current pane by right clicking and selecting the Maximise Pane menu option. To return a maximised pane to its normal size right click and select restore panes.



## 6.2 Devices

This pane displays the devices you have defined.

### 6.2.1 Add

Click to Add a new device.

The "Device Information" window has five (or six) tabs:

- [Device info](#) - describes the device Make/Model/etc
- [Passwords](#) - device login/enable and other passwords
- [Prompts](#) - describes the command line prompts used by a device if they differ from the default.

- [Contact info](#) - describes contact information of personnel responsible for a specific device.
- [Extra info](#) - Additional device specific information and fields.
- [Variations](#) - [Only visible on supported devices] Specify user-defined device variations.

**Device Information**

Device info | Passwords | Prompts | Contact info | Extra info | Variations

Vendor: [All Vendors]   
 Filter by vendor or show 'All Vendors'.

Device Type: Cisco.Firewall.ASA

Group:\* Default   
 Logically group your devices, if desired.

Name:\* ASA Firewall 2   
 A unique name for this device.

Host Address:\* 127.0.0.1

File Name:\* ASA\_Firewall\_2   
 Base file name for this device.(unique)

Model: ASA5505   
 Select model from list or type your own.

Connect via:\* Direct connect   
 Name of another device to connect to first, if necessary.

Method:\* Telnet

Port:\* 23

Ping device | Telnet/SSH | OK | Cancel

**Buttons:** (these appear at the foot of all device tabs)

**"Ping device"** - Sends ICMP Ping packets to the highlighted device.

**"Telnet/SSH"** - Uses the defined Telnet or SSH client program to start a manual session with the highlighted device. The Telnet & SSH client programs can be defined in the [CatTools Setup dialogue](#).

**"OK"** - Accept the current changes and return to the main display.

**"Cancel"** - Cancel any changes and return to the main display.

#### 6.2.1.1 Device info

**Vendor:** Use the drop down list to filter the 'Device Type' list box by vendor or choose all vendors to show all Device Types.

**Device Type:** Use the drop down list to select the CatTools device type.

**Group:** Enter or select the name of a group you want this device to belong to. This is a free form field where you can simply create a new group by typing new text. Use group names to group your devices into logical or physical categories.

**Name:** Give the device a unique name.

**Host Address:** Enter the IP address of the new device (in standard aaa.bbb.ccc.ddd format, or use a hostname)

**File Name:** This is the base file name CatTools will use for this device's data and reports. This field reflects the device name, but transforms any characters that are not usable in a file name.

**Model:** Use the drop down list to select the device Model. This is a free form field so you can enter any text you like. This field is only for documentation and has no effect on the operation of CatTools - the Device Type determines how CatTools handles a session with a device.

**Connect via:** Use the drop down list to select a device to connect via. The default is "direct connection". You only need to specify another device when direct access to the desired device is not possible. CatTools allows you to hop from one device to another using Telnet or SSH to reach your final destination. For example, if your device is behind an access list, but a Linux box has access to that device, you can connect via the Linux box first, then launch a telnet or SSH session to the destination device from there.

Note: When using a Cisco router as a jump point, it is recommended that you disable "logging synchronous" in the Line VTY section of the config. This can cause problems when trying to telnet out from the router.

By default most Cisco routers would have been configured with 5 lines (line vty 0 4). As CatTools is multi threaded and can support (depending on your edition) up to 30 client threads (connections); if you have created an activity for more than 5 devices which all connect via the router, you may end up with timeout errors or connection failures as a result of all the available router lines being used.

To prevent these errors occurring you could:

- a) Increase the number of VTY lines available on your main connect via router.
- b) Set the 'client threads' to use '5 threads'.
- c) Use a Linux box as a connect via device instead of a router.

**Method:** Use the drop down list to select connection method. The default is "telnet". Note that when using SSH there may well be a specific variant of it that is required to connect with the specific device. For instance, Netscreen devices supporting SSH2 require the variant SSH2-nopty to connect successfully with CatTools. More information on Session connections is available [here](#).

**Port:** Enter the port number the selected connection method is to use. The telnet default is 23, etc, and typically this does not need to be changed. SSH connections should use port 22.

### 6.2.1.2 Passwords

**VTY Password:** Enter the Virtual Terminal or initial login password here

**Enable Password:** Enter the enable mode or privileged password here

**Privilege Level:** You may set the enable mode privilege level if required. This is typically not required and so is best left blank. On a Cisco router, there are 16 privilege levels (0 to 15). Standard enable mode puts you in level 15.

This is appended to the enable command when it is sent to the device, so if it is present and not required it may cause an error when the device tries to authenticate enable mode.

**Console Password:** Enter the console password if using a direct COM port connection.

**Username:** Enter the Username (e.g. AAA, TACACS, RADIUS or Local username/password, etc.)

**Password:** Enter the Password (e.g. AAA, TACACS, RADIUS or Local username/password, etc.)

**SSH Username:** Enter the SSH Username if required.

**SSH Password:** Enter the SSH Password if required.

**SNMP Read:** Enter the SNMP Read community name for your device. The SNMP Read community name default is normally "public", however it is recommended you change your SNMP Read community name to some else. This field is required for running the SNMP.System. Summary activity otherwise  
*SNMP timeout or unknown error* will occur.

**SNMP Write:** Enter the SNMP Write community name for your device. The SNMP Write community name default is normally "private", however it is recommended you change your SNMP Write community name to some else.

**Initial login requires password:** tick box if initial login to this device requires a password (typically the VTY or initial console password).

**Initial login requires username/password:** tick box if initial login to this device requires both a username and password.

**Enable mode requires username/password:** tick box if login to enable mode on this device requires both a username and password. The value from the Username field is used along with the Enable password.

### 6.2.1.3 Prompts

Entries are only required here where a device has non-standard prompts configured.

**VTY Prompt:** Enter the non-standard VTY login prompt for this device.

The VTY prompt refers to the text that the device prompts you with when asking for the initial login password.

For example:

"Password: "

**Enable Prompt:** Enter the non-standard Enable prompt for this device.

The Enable prompt refers to the text that the device prompts you with when asking for the Enable password.

For example:

"Password: " or,

"Super user login: "



**Console Prompt:** Enter the non-standard Console prompt for this device.

The Console prompt refers to the text that the device prompts you with when asking for the initial console login password.

For example:

"Password: "

**Username Prompt:** Enter the non-standard Username prompt for this device.

The Username prompt refers to the text that the device prompts you with when asking for the initial login user name, normally used along with a password for AAA/TACACS/RADIUS or Local authentication.

For example:

"Username: " or,

"Please enter login name: "

**Password Prompt:** Enter the non-standard Password prompt for this device.

The Password prompt refers to the text that the device prompts you with when asking for the initial login password, normally used along with a username for AAA/TACACS/RADIUS or Local authentication.

For example:

"Password: " or

"Please enter login password: "

#### 6.2.1.4 Contact info

**Address 1:** Location of the device (can be seen on the Devices list if the "Location" column has been enabled).

**Address 2:** Location of the device.

**Address 3:** Location of the device.

**Contact Name:** The name of the person responsible for this device.

**Contact Phone:** How to contact the person responsible for this device.

**Contact E-mail:** How to contact the person responsible for this device.

**Contact Other:** Any additional contact info.

**Alert e-mail:** Who to notify by e-mail of any alarms or alerts for this device.

#### 6.2.1.5 Extra info

**Serial number:** The serial number of this device.

**Asset Tag:** Asset tag information.

**Identification:** Identification info for this device.

**Other info:** Any other serial number information.

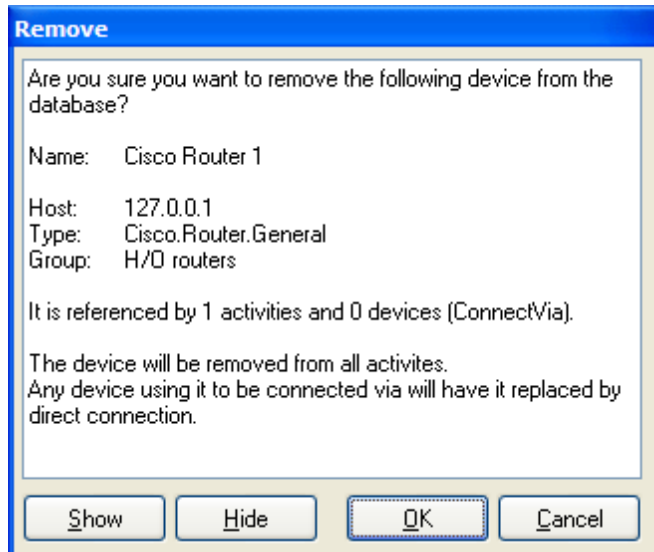
**Activity Specific1:** Information specific to a particular activity.

**Activity Specific2:** Information specific to a particular activity.

These fields are all free form and you can enter any text you wish.

## 6.2.2 Remove

Click to Remove a selected device. The following dialogue is shown:



This shows whether the device is used in any activities or devices. Use the **Show** button to see a list of entities the device is connected to.

If you choose to remove the device any relationships it has with activities or devices are also removed.

## 6.2.3 Edit

Click to Edit a selected device.

Opens the "Device Information" window, with the same tags and options as with the "Add" device button, but with the existing configuration for the highlighted device shown.

## 6.2.4 Copy

Click to Copy a selected device.

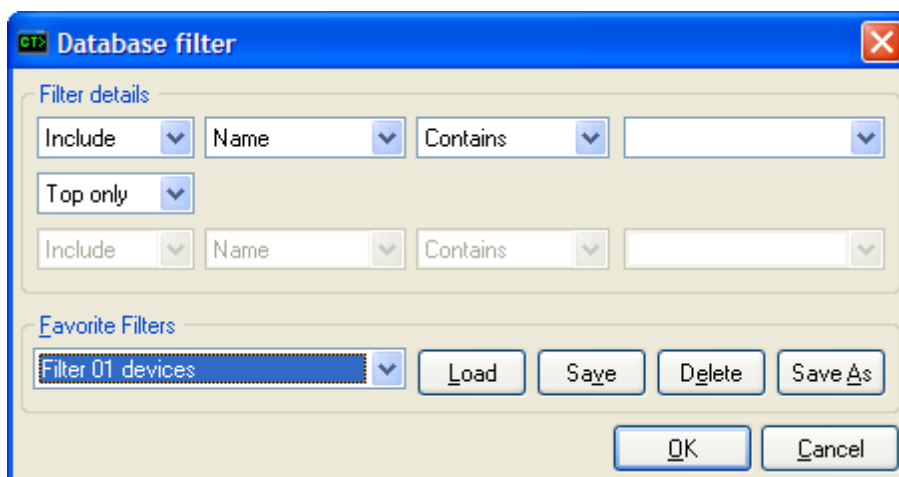
Opens the "Device Information" window, with the same tags and options as with the "Add" device buttons, but with the existing configuration from the highlighted device shown.

You must at least change the name of the device to save this configuration as a new device.

## 6.2.5 Filter

Click to filter the on-screen display of devices under the Devices tab.

This button pops up the "Database Filter" window :



In the "Filter details" box are several fields containing drop down lists.

In the first line of fields, the left-most field has two choices, "Include" or "Exclude" Use this field to either include or exclude specific device types, names, or other properties as shown in the following fields.

The next field has four choices:

1. **Group**
2. **Name**
3. **Host name**
4. **Type**

By selecting these options, different sets of devices may be included or excluded from the device display.

The third field specifies which part of the Group or Name etc the filter will operate on, so it has three choices:

1. **Contains** - any part of the name may contain the specified string
2. **Starts With** - the start of the name matches the specified string
3. **Ends With** - the end of the name matches the specified string

The right-most field is for entering the specific string that you want the filter to match against the device name, or group, etc.

The second line in this box contains only a single field, which has three options accessed by a drop down list.

The first two options are simple Boolean operators used to combine filters, either by a logical AND operation or a logical OR operation.

The third option in this field is the "**Top Only**" option. This will display only the top-most device that satisfies the criteria entered in the first line of filter options.

The third line of fields in this box are identical to the first line, being used to create a second filter that can be combined with the first.

Using the "**Favourite filters**" box, you can save your filters for later use. These options are fairly self-explanatory:

The "**Save As**" button saves a new filter.

The "**Save**" button saves a changed filter to disk.

The "**Load**" button loads the specified filter from disk.

The "**Delete**" button deletes the specified filter.

## 6.2.6 Show All

The Show All button removes all filtering, so that all devices in your database are displayed.

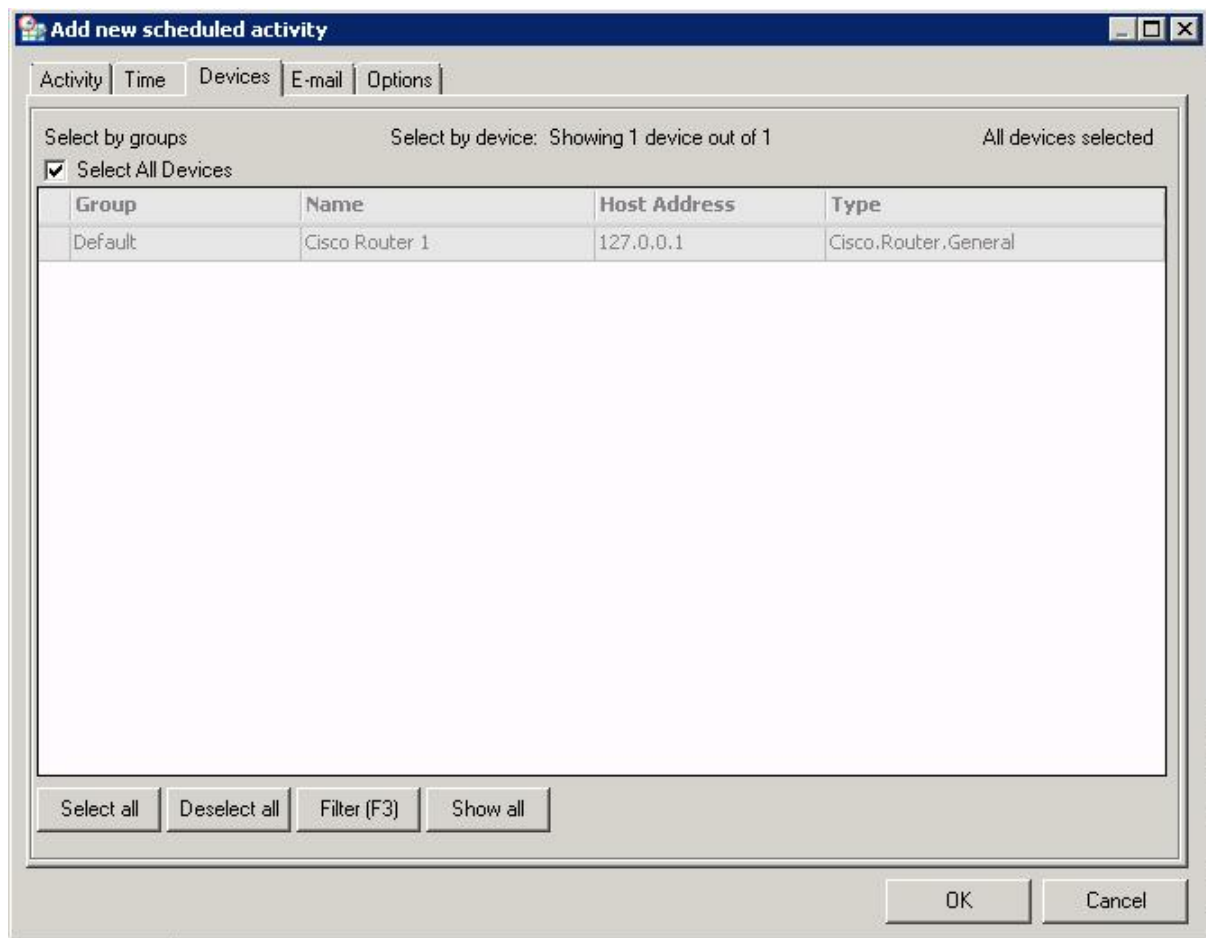
## 6.2.7 Select All Devices

The **Select All Devices** option will run an activity against all devices available without having to select them one by one. Checking this box allows you to automatically add new devices to existing activities when new devices are added.

For example:

You create a backup activity for your current 15 devices, but 4 weeks later you buy new equipment and add these new devices into CatTools.

If the backup activity was setup with this **Select All Devices** option, these new devices will be automatically picked up by the activity.



## 6.3 Activities

This pane displays all the activities you have defined.

Right clicking will bring up a context menu allowing customisation and access to core functions.

### 6.3.1 Add

Click to Add a new activity.

This will bring up the [Add a new scheduled activity](#) window.

### 6.3.2 Edit

Click to Edit a selected activity.

This will bring up the [Edit scheduled activity details](#) window.

### 6.3.3 Copy

Click to Copy a selected activity.

This will bring up the [Edit scheduled activity details](#) window.

### 6.3.4 Run now

Click to immediately run the Activity that is selected/ highlighted in the grid.

### 6.3.5 Start timer

Click to Start Timer and run scheduled activities.

When the timer is running all of the checked/ticked Activities will have their schedules run at what ever interval you have set for them.

### 6.3.6 Remove

Click to remove / delete the currently selected activity.

## 6.4 Activity log

This pane displays a log of all activities that have run since CatTools was started the last time.

These entries are also saved to the activitylog.txt file in the main CatTools folder.

### 6.4.1 Clear

Clears the display of any activity information.

This does not delete the activity log file; it just clears the display.

### 6.4.2 View report

View the report associated with the highlighted activity. This will automatically switch the display to the Report pane and open the report for viewing.

## 6.5 Compare

This pane enables you to create a comparison report of two files of your choice.

The report is the same format as that created by the [Device.Backup.Running Config](#) activity when changes are detected.

### 6.5.1 Save

The Save function enables you to save the comparison data displayed to a disk file.

### 6.5.2 Clear

The Clear function clears the display of any compare information.

### 6.5.3 Select

The Select function displays a dialogue that allows you to select the two files to compare.

## 6.6 Info log

This pane displays a grid of all messages displayed by CatTools while it is running, in the order that they occur.

The display may be filtered by using the View filter.

The messages are also logged to the **infolog.txt** file in the main CatTools folder.

### 6.6.1 View Filter Drop-Down

This allows you to change the filter options on the Info Log display which controls which level of messages are displayed.

There are four filters available:

1. **Show Error events only**
2. **Show Warning events and above**
3. **Show Info events and above**
4. **Show Debug events and above**

"**Show Info events and above**" is the default setting.

### 6.6.2 Clear

The Clear function clears the Info Log display.

## 6.7 Report

The Report pane enables you to view any text reports that are created by CatTools.

You may also view any tab delimited text file in the grid.

Fields in the viewed file may be modified and saved as required.

### 6.7.1 Open

Click on this menu option to open a Report file.

A file browser window will appear. Select the file you want to open and Click the '**Open**' button.

The in-built report viewer will open tab delimited files and display them in the grid for viewing, sorting, editing and saving.

### 6.7.2 Save

Saves the currently displayed Report to a file.

### 6.7.3 Clear

Clears the Report display.

### 6.7.4 Delete

The Delete function enables you to delete the report file currently being viewed.

### 6.7.5 Refresh

The Refresh function refreshes the view of the report file currently being displayed. This is useful when a report file has been updated while the view is open.

## 6.8 TFTP

This pane enables you to view messages originating from the CatTools TFTP server.

You can also stop or start the TFTP server from this tab.

### 6.8.1 Start

This button starts the TFTP server if it is not already started.

### 6.8.2 Stop

This button stops the TFTP server if it is running.

### 6.8.3 Clear

This button clears the display of TFTP server messages.

## 6.9 Display

The Display pane shows the current message displayed by each of the various tasks and Clients associated with CatTools.

The Thread column identifies the originating task, which may be the main CatTools task or one of the Clients.

The Device column identifies the device currently linked to the task.

The Status column shows the latest message from the task.

### 6.9.1 Clear

This button clears the Display pane of any messages.

## 6.10 Mail

The Mail pane shows messages currently in the mail queue waiting to be sent.

Summarised in the title bar at the top of the window is the total number of messages currently in the queue as well as the current status of the Mailer.

The Mailer can be stopped and started using the buttons at the bottom of the window or through the popup menu which is shown when the pane is right clicked. You would stop the mailer if you need to change an email message for some reason before it is sent, or if you want to perhaps delete a message from the queue.



Entries in the list can have their To: From: and Subject: fields edited in-line should the need be necessary. The retry state may also be reset on failed messages by selecting this option from the right click popup menu.

The same right click popup menu also contains options for customising the display allowing the user to turn on and off various columns.

Entries may be deleted by selecting them and then choosing the Delete option from the right click pop up menu.

If the State of a message is set to not sendable due to some kind of error, you can reset the state by selecting the message and choosing the Reset State function. This makes the message available to be sent by the mailer again.

## 7 Notification

CatTools has options for notifying you when an event occurs.

- [Email](#)

### 7.1 Email

Sending an email message is one method CatTools uses to notify you of selected events.

The mailer engine of CatTools provides flexible facilities for emailing reports and error notifications created by activities.

This engine normally runs asynchronously to the main CatTools task. When CatTools assembles a notification for emailing, it passes this to the mailer. Rather than sending the message immediately while CatTools waits, the mailer creates the message and puts it on a queue. CatTools can carry on processing once the message is queued successfully. At set intervals, the mailer connects to the mail server and sends it the messages that are on the queue. Meanwhile the CatTools main task continues to function without interruption.

The queuing mechanism enables messages to remain in safety until they can be sent. If the mail server is unavailable for some reason, messages are saved on the queue until the server becomes available.

The mailer can also send messages via a secondary mail server if the primary server is unavailable.

Mail server authentication is supported. A username and password can be used with standard SMTP AUTH type authentication. The mailer can also send the username and password to a linked POP server before sending mail to the SMTP server.

Further details are available on the [Mail Pane page](#).

## 8 Custom scripting

CatTools release v3.3.1 and later provides a custom scripting facility for you to add your own custom devices and activities to the user interface, should your device not be supported by one of the pre-defined device types, or the supplied activities not meet your specific requirements.

Detailed information on custom scripting; how to create the custom files, location of the custom scripting templates, code examples, etc. can be found within the following help file pages:

- [Creating a custom device](#) within the [Devices](#) chapter for custom device scripting.
- [Creating a custom activity](#) within the [Activities](#) chapter for custom activity and reports scripting.

When adding custom devices or activities/reports to CatTools, you will need to edit the script files using a [script editor](#).

## 8.1 Variables

Below are a list of the built in variables available to the Client scripts.

Use the "cl" prefix to access a variable or function from the client.

For example: If cl.QuitNow = 1 then exit sub

### Variables:

#### **QuitNow** As Long

If set to a value other than 0, indicates that the script must stop processing and exit immediately.

When the STOP button is pressed on the Activities page, the value of QuitNow will be set to 1 for each running thread.

Ensure that this variable is checked within any long duration loops so that the client will respond to the stop events in a timely manner.

#### **DeviceName** As String

Name of the current device that has been passed from the main script via the marshal threads function.

Use this value to perform further database lookups based on this device name.

#### **ScheduleNumber** as integer

Contains the current schedule number. Each activity is assigned a unique identifying number. This number is used as the unique key for database lookups of activity information.

#### **ScriptFileClient** As String

File name of the current script file that is currently being run

#### **UniqueFileID** As String

Unique file ID prefix for this script (set from the Main script). Use this unique value when writing results files that are to be read by the main script. This avoids other threads from overwriting the same results file.

#### **ProcessID** As Long

The ID number of this thread (1 to MaxThreads). This value is not unique since the thread can

be reused for running a different script and connecting to a different device. This variable can be used when signalling the main script as a way of identifying the number of this thread.

**AppPath** As String

The folder in which the main application (CatTools) is installed. This folder is always terminated with a "\" character. Use this variable as the root path when saving data to results files. (The ClientTemp folder for example is located under the root path.)

**DeviceHostnameID**

Hostname as recovered from the device after login. For example: sales1600. When a connection to a device is made, the hostname is extracted from the prompt. A prompt of "sales1600>" gets extracted and converted to "sales1600".

**DeviceVTYPrompt**

This value holds the prompt that is expected to be returned after an action is taken in vty mode. For example, after a show interface command has completed, the DeviceVTYPrompt can be expected. Example: sales1600>

**DeviceEnablePrompt**

This value holds the prompt that is expected to be returned after an action is taken in enable mode. For example, after a show interface command has completed, the DeviceEnablePrompt can be expected. Example: sales1600#

**RecDataTimeOut** as long

Holds the number of seconds to wait for a response from the device. For example, time to wait for login prompt to be received after sending a command.

**RxBuffer** As String

Holds the data returned from the active connection. For example: SendData Blah, then WaitForMultData. After a return value has been matched or the timeout has occurred, the received data will be held in RxBuffer.

**StripVT100ESC** as Long

If set to non zero, vt100 type escape sequences are removed from the buffer stream at the point of reception.

This is useful for devices with menu based telnet interfaces. This variable can be toggled at any time to enable or disable the removal of the escape sequences. For example, set this variable = 1, then call the WaitForMultData function.

**DebugMode** as long

If set to 1, the send and receive data from the active connection is written to a debug text file. The filename is made up from the formula: AppPath + "Capture" + UniqueFileID + ".txt". This variable is set to 0 for normal operation.

**Device values.**

After calling the function `DBDevicesGetAllFields(DeviceName)`, the following fields are loaded with the information from the database.

**CurDevType** As String

Device type. For example: `Cisco.Router.General` or `Cisco.Firewall.PIX`.

To ensure a good match, always test for lowercase values. If `Lcase(CurDevType) = "cisco.router.general"` then...

**CurDevGroup** As String

Group to which the device belongs. This is a free form field and can be any value that the user specifies.

**CurDevName** As String

Device name. This is a free form field and can be any value that the user specifies. The device name must be unique.

**CurDevHostAddress** As String

IP address or hostname of the device.

**CurDevFilename** As String

A base filename for use when saving information to file for this device. The filename is made from the device name. All illegal filename characters are replaced with an underscore.

**CurDevModel** As String

Device model. For example: 1601 or 3300. This is a free form field and can be any value that the user specifies.

**CurDevConnectVia** As String

The name of the device to connect to first before attempting a telnet or SSH session. For example, connect to device A , then attempt a telnet connection to the current device.

**CurDevTelnet** As String

Method used for connection. Dependant on the particular device. Example: Telnet, SSH1, COMM.

**CurDevTelnetPort** As String

Port to use for connection attempt. Example: 23 for telnet, 22 for SSH, 1 for COMM1.

**CurDevSession** As String

Session number to use once connected to device. Example: Telnet to device, then issue session 15 command to connect to LANE card.

**CurDevVTYPass** As String

VTY password for the device.

**CurDevConsolePass** As String

Console password for the device. The console password is only required when the connection is via the console (COMM port connection).

**CurDevEnablePass** As String

Enable password for the device. This is usually the enable-secret password on Cisco devices.

**CurDevPrivilegeLevel** As String

The privilege level to use when entering enable mode. Example: Enable 7.

**CurDevAAAUsername** As String

Username to use when device is set to use AAA (TACACS/RADIUS/Username)

**CurDevAAAPassword** As String

Password to use when device is set to use AAA (TACACS/RADIUS/Username)

**CurDevVTYPrompt** As String

The VTY prompt to expect from the device. Normally set to "" if the default prompt is expected. Example: Login:

**CurDevConsolePrompt** As String

The Console prompt to expect from the device. Normally set to "" if the default prompt is expected. Example: Login:

**CurDevEnablePrompt** As String

The Enable mode prompt to expect from the device. Normally set to "" if the default prompt is expected. Example: Password:

**CurDevAAAUserPrompt** As String

The Username prompt to expect from the device. Normally set to "" if the default prompt is expected. Example: Username:

**CurDevAAPassPrompt** As String

The Password prompt to expect from the device. Normally set to "" if the default prompt is expected. Example: Password:

**CurDevRequireVTYLogin** As String

Check box value. 0 = no VTY password expected, 1 = VTY password expected.

**CurDevLoginUsesAAA** As String

Check box value. 0 = no Username/Password expected, 1 = Uses AAA. Username/Password required.

**CurDevEnableUsesAAA** As String

Check box value. 0 = no Username/Password expected for enable mode, 1 = Uses AAA for enable. Username/Password required when entering enable mode.

**CurDevAddress1** As String

Free form address information

**CurDevAddress2** As String

Free form address information

**CurDevAddress3** As String

Free form address information

**CurDevContactName** As String

Name of contact for device

**CurDevContactPhone** As String

Contact phone number for device contact

**CurDevContactEmail** As String

Contact e-mail address for device contact

**CurDevContactOther** As String

Other contact info for device contact

**CurDevAlertEmail** As String

E-mail to send alert info to for this device. If left blank, the e-mail address defined in the

properties will be used.

**CurDevSerialNumber** As String

Device serial number. Free form field.

**CurDevAssetTag** As String

Device asset tag information. Free form field.

**CurDevIdentification** As String

Other device identification. Free form field.

**CurDevSerialOther** As String

Other device identification. Free form field.

**CurDevActivitySpecific1** As String

Activity specific information for this device.

**CurDevActivitySpecific2** As String

Activity specific information for this device.

**Temp01 to Temp10** as Variant

General purpose global variables for use by the client scripts. Can be used to allow transfer of data from one script to another. These values are not initialised on startup and may contain data from previous script instances.

## 8.2 Functions

Below are a list of the built in functions available to the Client scripts.

Use the "cl" prefix to access a function from the client.

For example: cl.Log 1, "Expected response not received from command ..."

### Functions:

Function **ConnectHost()** As Boolean

Attempts to connect to the current host specified in CurDevHostAddress. If the connection is successful, a value of true is returned. The connection will be attempted 3 times before failing. A warning message is logged if a connection attempt fails. After 3 failed attempts, an error message is logged.

Function **ConnectHostSingleAttempt()** As String

Attempts to connect to the current host specified in CurDevHostAddress. If the connection is successful, a value of "OK" is returned, otherwise the error description will be returned. This function is called 3 times by ConnectHost.

Function **ConvertToFilename(Filename as variant)** As String

Replaces any invalid characters in the filename with an underscore. The following characters are considered illegal in a filename: "&+?:;,()=|V<>\*^%@" . Quotes and spaces are also replaced. This function can be used to convert a device name into a suitable filename.

Function **CreateFile(Filename)** As Boolean

Attempts to create the specified file even if the path does not exist. If the file already exists, no action will be taken. If the file does not exist, the path and a file are created. The file will be 0 bytes in length.

Function **CurrentDateStamp(Format as long)** as String

Returns the current date formatted based on the value of "Format".

Format values:

0 = ISO Format (YYYY/MM/DD)

1 = US Format (MM/DD/YYYY)

2 = UK Format (DD/MM/YYYY)

Function **CurrentDateTime(Format as long)** as String

Returns the current date and time formatted based on the value of "Format".

Format values:

0 = ISO Format (YYYYMMDDHHNNSS)

1 = US Format (MMDDYYYYHHNNSS)

2 = UK Format (DDMMYYYYHHNNSS)

Function **CurrentDateTimeStamp(Format as long)** as String

Returns the current date and time without separators, formatted based on the value of "Format".

Format values:

0 = ISO Format (YYYY/MM/DD HH:MM:SS)

1 = US Format (MM/DD/YYYY HH:MM:SS)

2 = UK Format (DD/MM/YYYY HH:MM:SS)

Function **DBCheckScheduleOption(ScheduleNumber, Option Number)** As Long

Returns the activity specific option check box value. Each activity has 10 optional data values that can be set for the activity. Pass the activity number and an option value (from 1 to 10) and the return value of the specific option check box will be returned. 0 = unchecked, 1 = checked. Refer to the activity ini file for details of each specific option.

Sub **DBDevicesClearCurDevFields()**

Clears all the current device variables and sets them to ""

Sub **DBDevicesGetAllFields(DeviceName)**

Locates the specified device in the database and loads all the fields into the CurDevXXX variables. To check if the function worked, check len(CurDevName) > 0.



Function **DBDevicesGetField(DeviceName, FieldName)** as Variant  
Returns the data contained in the field "FieldName" of device "DeviceName" from the Database file.  
Returns blank if retrieval failed.

Function **DBScheduleGetField(ScheduleNumber, FieldName)** as String  
Returns the data contained in the field "FieldName" of activity "ScheduleNumber" from the Database.  
Returns blank if retrieval failed.

Function **DeleteFile(Filename)** As Boolean  
Deletes the specified file and returns true or false based on if the file exists after the delete attempt. True = File deleted. False = File still exists and was not deleted successfully.

Sub **DisconnectHost()**  
Closes the active connection to the current device

Function **FileExists(Filename)** As Boolean  
Tests to see if the specified file exists. Returns true if the file is found, false if it is not.

Sub **FlushRxBuffer()**  
Flushes the data found in the receive buffer and waits until there is no further data being received from the active socket.

Sub **InfoUpdate(Message)**  
Sends an info update message back to the main program.

Sub **LogToFile(Filename, Data, Append)**  
Opens a file for writing. Writes the data stored in Data to the file.  
If Append is set to 1, the data is appended to the end of the file. If Append is set to **1**, the file is overwritten with the new data.

Function **Ping(Hostname As String, Timeout As Long, PingCount As Long, ProcessID As Long)** As String  
Attempts to ping the host name and returns the results in the format:  
!!!! TAB 100% TAB 10 TAB 100 TAB 80 or ..... TAB 0% TAB TAB TAB  
Hostname is the IP address or hostname of the device to ping  
Timeout is in seconds. The default is 5 seconds.  
PingCount is the number of Ping echo requests to send. Default is 5 pings.  
ProcessID is the current processID (cl.ProcessID). This is used to identify the sent ping packets so that the responses can be tied back to the correct process that sent them.  
The min, max and average response times will be calculated from any ping echoes received.  
The return data is tab delimited for easy parsing and reporting.

Function **ReadFromFile(Filename)** As String  
Reads the contents of the specified filename and returns the data as a string value. A "" value is returned if the file doesn't exist or contains no data.

Function **ReplaceClientFilenameVariables(Filename)** As String

Replaces the %Variable% item with the associated value.

%AppPath% = cl.AppPath

%GroupName%" = cl.CurDevGroup

%DeviceName% = cl.CurDevName

%DateISO% = cl.CurrentDateStamp(0) ("YYYY-MM-DD")

#### Sub **SaveResults(Data, Append)**

Saves results data into separate files. The file name is created from the device name and UniqueFileID set in the Main script.

Filename format:

Filename = cl.AppPath + "\ClientTemp\" + UniqueFileID + ConvertToHex(DeviceName) + ".txt"

DeviceName is converted to hex format to remove any invalid chrs present.

#### Sub **SendData(MessageText)**

Sends the string in MessageText to the current connection.

#### Function **WaitForMultData(Mult, Choices, Timeout)** As Long

Monitors the RxBuffer for data contained in the Mult() array. If a match is found, the array index value is returned. A return value of 0 indicates that no match was found before the timeout occurred. The timeout is specified in seconds and the variable cl.RecDataTimeOut is usually passed as this value. Choices indicates the number of elements in the Mult array. Example: Mult(1) = "Login:" Mult(2)= "Password:", Choices=2.

#### Function **ReplaceControlChrs(Message)** As String

Replaces any control characters (less than ASCII value of 32) with a text string of the numeric value. For example: A tab character (ASCII 9) becomes <009>. This is used when logging debug data to the display or file for later analysis.

### 8.2.1 cl.Log

#### Sub **Log(Priority, Message)**

Sends a log event message back to the main program

Use one of the following priority codes to classify the message

Priority values:

4 = Debug

3 = Info

2 = Warning

1 = Error

## 8.3 Script editors

In order to modify custom script files, you will need to load them into a text file or script editor.

While Notepad.exe (provided with Windows) is a capable text file editor, you may find the script file much easier to read and modify within a syntax highlighting (color-coding) script editor.

If you have Visual Basic or Visual Studio installed on your system, then open the .txt files with either of these programs to view the files with syntax highlighting.

If you do not have access to a syntax highlighting editor, then you may want to try downloading one from the Internet.

Once installed, the settings you need to change in order to open a custom script .txt or .custom file with syntax highlighting are\*:

- 1) Within the menu item: Settings --> Highlighters settings  
Clear all language tickbox options (click the 'None' button) and select just the **MS VBScript** language option.  
Add to the 'File Masks' box \*.txt and \*.custom file types. This will add these file types to the VBScript file filter list.
- 2) Open one of your script files and select the menu item: View --> Change Syntax  
Select from the available syntaxes **MS VBScript**, then click the 'OK' button to apply the syntax color highlighting. Commented code (i.e. code that's been added to help assist you with creating your custom scripts) will now be highlighted in green.

(\* settings apply to PSPad version 4.5.3 (Build 2298), being the current release @ 14-Jan-2008)

## 9 Troubleshooting

If you are having problems look here to see if there is an answer. Alternatively you can check our [CatTools Knowledge Base](#) articles.

If you still require help, then please contact our CatTools Technical Support Team using the [Technical Support](#) form on our [web site](#).

### 9.1 Device specific

There are a number of devices that require specific settings or configuration setups within CatTools in order for CatTools to handle their behaviour.

Please see the [Device specific information](#) section for a list and of these devices.

### 9.2 "Error: 70 Permission denied" message in Info Log

#### Error: 70 Permission denied (starting client threads)

If the you see the error: *"Problem starting CatTools\_Client instance #1. Error:70: Permission denied"* in the InfoLog, the user running CatTools doesn't have enough permission to launch the CatTools\_Client DCOM component.

We recommend you try the following...

- Run the program "dcomcnfg.exe" from the Start | Run menu
- Locate the section called Component Services | Computers | My Computer | DCOM Config
- Select the item called CatTools\_Client.clsMain (or CatTools\_Client.KiwiSNMP depending on your version)
- Right click and choose Properties
- Click the Security tab
- Under the Launch and Activation permissions, click Customize
- Click the Edit button
- Adjust the permissions of the user running CatTools to allow local launch and local activation.

**Note:** The service runs under the LocalSystem account.

After making the permissions changes, reboot the system and see if CatTools is able to start the client threads correctly.

(The *EventLog* would show this as a DCOM error with Event ID: 10016.)

## 9.3 Reporting problems

When reporting a problem with CatTools, please ensure you provide as many details as possible.

If, for instance, you are getting an error logging on to a device, tell us what the actual error is that you are getting. An error message should be present in the infolog that describes the error.

It is very helpful to know what type of device the problem is happening to. CatTools supports a number of different devices that can exhibit very different problem behaviours.

If you are sending us your infolog.txt file it is very helpful to tell us the approximate time the problem occurred. It is easy for us to miss something when we are scanning a 50MB text file for an error message.

Please do not send us raw screen shots of messages or setup issues unless there is no other way of getting the information across. Mail servers often block this data, and they produce very big emails. Use the CatTools facilities on the File menu to provide us with the text information.

If sending data present in the infolog, please send the file itself as an attachment, zipped if it is large. Trying to copy and paste entries from the infolog into an email can produce some pretty garbled emails. We often need to see more of the infolog than is copied to the email as well to provide us with context.

Doing this allows us to answer your queries in the quickest time.

## 9.4 Remote Desktop Systems

CatTools may experience problems starting the Client threads when running on machines that use remote desktop software.

### **RealVNC and TightVNC**

CatTools was tested on TightVNC 1.3.9 and RealVNC 4.3.2 and (with the exception of Windows Vista) had no issues.

Some older versions of VNC can cause issues with starting the client threads. It appears that older versions of these applications interrupt the windowing system and cause the client to only start one thread.

The first recommendation is to upgrade your current version of VNC to see if this fixes the problem.

If you can't upgrade your VNC software, then you may want to try changing the registry settings.

You will need to use RegEdit to set the "use\_Deferral" setting to 0.

```
Key:  
HKEY_CURRENT_USER\Software\ORL\VNCHooks\Application_Prefs\CatTools_Client.  
exe  
Set Dword: use_Deferral to 0
```

```
Key: HKEY_CURRENT_USER\Software\ORL\VNCHooks\Application_Prefs\CatTools.exe  
Set Dword: use_Deferral to 0
```

### PC Anywhere

CatTools was tested on the trial version of PCAnywhere 12.1 and (with the exception of Windows Vista) had no issues.

## 9.5 Remote Authentication

Some difficulties have been experienced when running CatTools on a network where remote authentication is used. The symptoms are random failures of CatTools to logon to devices during the running of a scheduled activity. The next time the activity runs, CatTools logs on to the same devices with no problems.

The problem can become more apparent on heavily loaded networks or where the remote authentication system is under load.

You may be able to set the device to wait longer for a remote authentication server response before timing out. For instance, with some Cisco devices you can issue the following command:

```
set tacacs timeout 20
```

to set the timeout period to 20 seconds where the default setting may be something like 5 seconds.

## 9.6 Anti Virus Tools

Some anti virus tools can provide barriers to accessing the network, either incoming or outgoing.

If you suddenly experience CatTools failing to connect to any devices, or being unable to connect to the mail server to send alerts, check that you do not have an anti virus program getting in the way.

McAfee Enterprise V8.0 can be set up to include program and port blocking. This can prevent CatTools from connecting to an email server for sending reports.

## 9.7 XP Firewall

If you are running CatTools on an XP system you need to be aware that after Service Pack 2 the XP Firewall is turned on by default. You may need to configure an exception for CatTools or make another adjustment if you continue to run with the XP Firewall on.

## 9.8 The service is running but nothing is scheduled

There may be occasions when the CatTools appears to be running, the scheduler is on, but no activities are run.

A look at the infolog file would probably show that an activity appears to be hung up and has not terminated. This may be caused by a logic loop or communications hanging up where CatTools cannot detect a timeout. Normally this is not something that happens every time, but may happen due to a glitch.

There is a setting on the [Misc](#) tab in the [CatTools Setup](#) that might be useful in allowing CatTools to continue running productively after such an occurrence. This allows you to forcibly [terminate](#) an orphaned activity after waiting for a set number of minutes. CatTools attempts to recover and resume scheduling activities afterwards.

You should normally leave the setting at 0 unless there is a problem so that you do not forcibly terminate an activity that takes a long time to run, but runs successfully to conclusion.

Another cause of this problem may be due to a permissions issue on the DCOM component. To verify this as a cause, check your Info Log for the following message:

"Problem starting CatTools\_Client instance #1. [Error:70: Permission denied](#)"

## 10 API

The CatTools API is included as a way by which you can directly read from and write to the CatTools database. The API can basically perform the same function as the 'Add', 'Remove' and 'Edit' buttons in the CatTools UI.

*Note:* The CatTools API is not available with the Freeware edition of CatTools.

If you create an application or code that uses the CatTools API, then you can only run this on the system where CatTools has been installed, as it needs to read the CatTools license details in order to work.

The API is limited to database connectivity and accessing of the device and device related data. There are no activity related public classes available within the API to modify activities.

The API is accessed using various [Classes](#) which are explained in this section. In addition you will also find sample code to help you get started.

### 10.1 Environments

Due to the capabilities of the CatTools API (i.e. it can directly modify device data in the CatTools database), it is assumed that someone using the API would have a reasonable knowledge of Visual Basic programming skills and therefore would probably be familiar with the Visual Basic development environment, for example Visual Studio; or alternatively knows how to utilize the VBA programming environment (Visual Basic editor) within Microsoft Office applications.

This is why the API sample code is written in VB/VBA form.

Within these development environments, you simply add a reference\* to the CatTools API called "CatTools API" to expose the classes available. Once the reference has been added, you should be able to simply copy and paste the sample code from the CatTools API help pages into your VB project and it will work.

However, if you are using a text editor (such as Notepad), you won't be able to add a reference to the CatTools API. For text editors, you need to change the code slightly to use the 'CreateObject' function to reference the CatToolsAPI available classes.

For example (within a text editor):

```
Dim DB
Set DB = CreateObject("CatToolsAPI.Database")
```

This would be used to replace the code in the sample code:

```
Dim DB As CatToolsAPI.Database
Set DB = New CatToolsAPI.Database
```

You can then save the text file as a VBScript (.VBS) file and run it.

In order for the API code to work, the CatTools API library file (Cattools.dll) must be registered on the system correctly, which you can do using the following from the START (button) > RUN dialog:

```
regsvr32 "C:\Windows\system32\cattools.dll"
```

\* Adding a reference (within a VB editor):

- For the Visual Basic editor within Microsoft Office applications, click the Tools > References... menu item and select the "CatTools API" from the available references.
- For the VB6 IDE, click the Project > References... menu item and select the "CatTools API" from the available references.

## 10.2 Classes

The following API classes are contained within the CatTools API.

### Classes

<a href="#">Database</a>	Used to connect to the database and verifies license requirements.
<a href="#">Devices</a>	An enumerator for a collection of Device objects as well as the method by which to get and put devices into the database.
<a href="#">Device</a>	Encapsulates each Device stored in the CatTools database.
<a href="#">DeviceTypes</a>	A collection of DeviceType objects
<a href="#">DeviceType</a>	A CatTools DeviceType. eg "Cisco.Router.General"
<a href="#">Groups</a>	A collection of Group objects
<a href="#">Group</a>	A CatTools device Group. eg "Default"

### 10.2.1 Database

The Database class is used to connect to the CatTools database. In doing so it also verifies your license.

The database that the API connects to is the same as the one stored in your KiwiCatTools-Settings.ini file.

**Properties**

Version	The version of the API dll file. Read Only.
Path	The database being accessed. Read Only. This is evaluated from your ini file.
EncryptionPassword	The encryption password needed to decrypt the database.
Err	The id of the last error encountered. Read Only.
ErrDescription	The last error encountered. Read Only.
Connected	Is the database connected ok? Read Only.

**Methods**

OpenConnection	Opens a connection to the database as defined in your ini file.
CloseConnection	Closes the connection to the database.
Devices	A collection of the devices contained in the database.
ErrClear	Clears any error messages.
DeviceTypes	A collection of the device types contained in the database.
Groups	A collection of the groups contained in the database.

**10.2.1.1 Sample Code**

Sample VB code showing you how to connect to your database using the CatTools API.

```
Dim DB As CatToolsAPI.Database
Set DB = New CatToolsAPI.Database

DB.EncryptionPassword = "MyPassword"

If DB.OpenConnection Then
    MsgBox "Connected ok"
Else
    MsgBox "DB connection failed: " & DB.ErrDescription
End If

DB.CloseConnection
Set DB = Nothing
```

**10.2.2 Devices**

The Devices class is used to enumerate CatTools [Device objects](#) and to enact changes to a Device. ie add, edit and delete.

The following Methods and Properties are available :

**Properties**

Count	Returns the number of devices in your database.
-------	---

**Methods**

AddNew	Returns a Device. This Device can then have its settings modified before being saved to the database. Takes as a property the name of the device-type that you would like to create.
DeleteByDevice	Deletes a device from the database. Takes a device object as a parameter.
DeleteByName	Deletes a device from the database using the devices name as an



GetByName	identifier.
Items	Returns a Device object. Takes the devices name as a parameter.
Refresh	The collection of Device objects.
SaveDevice	Resynchronise with the database.
	Saves a Device object to the database. Takes the Device to be saved as a parameter.

### 10.2.2.1 Sample Code

Sample VB code showing you how to iterate devices using the CatTools API.

```

Dim DB As CatToolsAPI.Database
Dim Devices As CatToolsAPI.Devices
Dim Device As CatToolsAPI.Device

Set DB = New CatToolsAPI.Database

DB.OpenConnection
If DB.Connected Then
    Set Devices = DB.Devices
    Debug.Print Devices.Count & " devices in the database"
    For Each Device In Devices
        If Device.Group = "Default" then
            Debug.Print Device.Name & " " & Device.HostAddress
        End If
        DoEvents
    Next
Else
    Debug.Print "Connection failed: " & DB.ErrDescription
End If

DB.CloseConnection

Set Device = Nothing
Set Devices = Nothing
Set DB = Nothing

```

### 10.2.3 Device

The Device class encapsulates a CatTools device.

The following Properties are available :

#### Properties

AAAPassword	Password: The password to use if your device requires a username/ password login.
AAAPassword Prompt	Prompt: What does the password part of a username/password request look like from your device.
AAAUsername	Password: The username to use if your device requires a username/ password login.
AAAUserPrompt	Prompt: What does the username request look like from your device.
ActivitySpecific1	Info: Not used
ActivitySpecific2	Info: Not used
Address1	Contact: Location of the device.
Address2	Contact: Location of the device.
Address3	Contact: Location of the device.
AlertEmail	Contact: Who to notify by e-mail of any alarms or alerts for this device.

AssetTag	Info: Asset tag information.
ConnectionMethod	Device: The method to use when connecting to this device. eg Telnet or SSH2.
ConnectVia	Device: The Name of the device to connect via.
ConnectViaID	Device: The ID of the device to connect via. -1 indicates a direct connection.
ConsolePassword	Password: The console (com port connection) password.
ConsolePrompt	Custom prompt. What does the console password request look like from your device.
ContactEmail	Contact: How to contact the person responsible for this device.
ContactName	Contact: The name of the person responsible for this device.
ContactOther	Contact: Any additional contact information.
ContactPhone	Contact: How to contact the person responsible for this device.
EnablePassword	Password: The password to get into enable mode.
EnablePrompt	Prompt: What does the enable password request look like from your device.
EnableUsesAAA	Password: Does entering enable mode require username/password validation?
FileName	Device: The base file name for the device.
Group	Device: The name of the Group the device belongs to. eq "Default"
GroupID	Device: The id corresponding to the Group field. -1 indicates a new Group is to be created.
HostAddress	Device: The IP address or host address for this device.
ID	Device: The id that identifies this device in the database.
Identification	Info: Identification info for this device.
LoginUsesAAA	Does connecting to the device require username/password validation?
Model	Device: A value describing the model of the device.
Name	Device: A unique name that describes this device.
Port	Device: The port to use when connecting to this device.
PrivilegeLevel	Password: Sets the enable mode privilege level.
RequireVTYLogin	Password: Does connecting to the device require password only validation?
SerialNumber	Info: The serial number for the device.
SerialOther	Info: Any other serial number information.
SNMPRead	Password: SNMP Read community name.
SNMPWrite	Password: SNMP Write community name.
SSHPassword	Password: The SSH password required for SSH connections to this device.
SSHUsername	Password: The SSH username required for SSH connections to this device.
TypeID	Device: The id of the device-type eq 10
TypeOfDevice	Device: The name of the device-type eq "Cisco.Router.General"
VTYPassword	Password: The password to use if your device verifies using a password only to log in.
VTYPrompt	Prompt: What does a password request look like from your device.

### 10.2.3.1 Sample Code

Sample VB code showing you how to use the Device object of the CatTools API to add, edit or delete an entry in the database.

#### Add

```
Dim DB As CatToolsAPI.Database
Dim Device As CatToolsAPI.Device

Set DB = New CatToolsAPI.Database
If DB.OpenConnection Then
    Set Device = DB.Devices.AddNew("Cisco.Router.General")
    If Not Device Is Nothing Then
        Device.Name = "My Device 01"
        Device.HostAddress = "192.168.1.1"
```

```

        Device.Group = "My Test Group"
        Device.RequireVTYLogin = True
        Device.VTYPass = "My Password"
        Device.ConnectionMethod = "Telnet"
        If DB.Devices.SaveDevice(Device) Then
            Debug.Print Device.Name & " saved to the database OK."
        Else
            Debug.Print "Adding " & Device.Name & " failed: " & DB.ErrDescription
        End If
    Else
        Debug.Print "Creation of a new device failed: " & DB.ErrDescription
    End If
Else
    Debug.Print "DB connection failed: " & DB.ErrDescription
End If

Set Device = Nothing
Set DB = Nothing

```

### Edit

```

Dim DB As CatToolsAPI.Database
Dim Devices As CatToolsAPI.Devices
Dim Device As CatToolsAPI.Device

Set DB = New CatToolsAPI.Database
If DB.OpenConnection Then
    Set Devices = DB.Devices
    Set Device = Devices.GetByName("My Device 01")
    If Not Device Is Nothing Then
        Debug.Print Device.Name & " Loaded OK"
        Device.AAAPassword = "My New Password "
        If Devices.SaveDevice(Device) Then
            Debug.Print Device.Name & " saved OK"
        Else
            Debug.Print Device.Name & " NOT saved: " & DB.ErrDescription
        End If
    Else
        Debug.Print "Device not found"
    End If
Else
    Debug.Print "DB connection failed: " & DB.ErrDescription
End If

Set Device = Nothing
Set Devices = Nothing
Set DB = Nothing

```

### Delete

```

Dim DB As CatToolsAPI.Database

Set DB = New CatToolsAPI.Database
If DB.OpenConnection Then
    If DB.Devices.DeleteByName("My Device 01") Then
        Debug.Print "Deleted OK"
    Else
        Debug.Print "Delete failed: " & DB.ErrDescription
    End If
End If

```

```

    End If
Else
    Debug.Print "DB connection failed: " & DB.ErrDescription
End If

```

```

Set DB = Nothing

```

## 10.2.4 DeviceTypes

The DeviceTypes class is used to enumerate CatTools [DeviceType objects](#).

The following Methods and Properties are available :

### Properties

Count	Returns the number of device types in your database.
-------	--

### Methods

Items	The collection of Device objects.
Refresh	Resynchronise with the database.
Sort	Sort the collection of DeviceType objects in ascending or descending order.

### 10.2.4.1 Sample Code

Sample VB code showing you how to list the Device Types found in your database using the CatTools API.

```

Dim DB As CatToolsAPI.Database
Dim DeviceTypes As CatToolsAPI.DeviceTypes
Dim DeviceType As CatToolsAPI.DeviceType

Set DB = New CatToolsAPI.Database

DB.EncryptionPassword = ""

DB.OpenConnection
If DB.Connected Then
    Set DeviceTypes = DB.DeviceTypes
    If Not DeviceTypes Is Nothing Then
        Debug.Print DeviceTypes.Count & " device types in the database"
        For Each DeviceType In DeviceTypes
            Debug.Print "->" & DeviceType.Name
            DoEvents
        Next
    Else
        MsgBox "Error:" & DB.ErrDescription
    End If
Else
    MsgBox "Connection failed: " & DB.ErrDescription
End If

DB.CloseConnection

Set DeviceType = Nothing
Set DeviceTypes = Nothing
Set DB = Nothing

```

## 10.2.5 DeviceType

The DeviceType class encapsulates a CatTools device type.

The following Properties are available :

### Properties

ID	The ID used to identify this Device Type in the database.
IniFile	The ini file used to define this DeviceType
Name	The friendly name of the DeviceType that you see on screen.

### 10.2.5.1 Sample Code

It is not possible to add/edit or delete Device Types.

You can however use the DeviceTypes class to enumerate DeviceType objects and populate them into a drop list for your user to select from.

See the [Device-Types sample code](#) for more information.

## 10.2.6 Groups

The Groups class is used to enumerate CatTools [Group objects](#).

The following Methods and Properties are available :

### Properties

Count	Returns the number of device types in your database.
-------	--

### Methods

Items	The collection of Group objects.
Refresh	Resynchronise with the database.
Sort	Sort the collection of Group objects in ascending or descending order.

*Note:* You can only enumerate Groups you can not add edit or delete them using this object. To add a Group simply give the text value to your [Device](#) and when it is saved the Group will be created.

### 10.2.6.1 Sample Code

Sample VB code showing you how to list the Groups found in your database using the CatTools API.

```
Dim DB As CatToolsAPI.Database
Dim Groups As CatToolsAPI.Groups
Dim Group As CatToolsAPI.Group

Set DB = New CatToolsAPI.Database

DB.OpenConnection
If DB.Connected Then
    Set Groups = DB.Groups
    If Not Groups Is Nothing Then
        Debug.Print Groups.Count & " groups in the database"
        For Each Group In Groups
            Debug.Print "->" & Group.Name
        Next Group
    End If
End If
```

```

        Next
    Else
        MsgBox "Error:" & DB.ErrDescription
    End If
Else
    MsgBox "Connection failed: " & DB.ErrDescription
End If

DB.CloseConnection

Set Group = Nothing
Set Groups = Nothing
Set DB = Nothing

```

### 10.2.7 Group

The Group class encapsulates a CatTools device Group object. ie These are the values presented in the Group drop list when you edit a device.

The following Properties are available :

#### Properties

ID	The ID used to identify this Group in the database.
Name	The friendly name of the Group that you see on screen.

*Note:* When CatTools starts it removes from the database any unlinked entries.

#### 10.2.7.1 Sample Code

It is not possible to add/edit or delete Groups using the Groups class.

To add a new group simply assign it to a [Devices.Group property](#).

You can however use the Groups class to enumerate Group objects and populate them into a drop list for your user to select from.

See the [Groups sample](#) for more information on how this might be achieved.

## 10.3 Limitations

The API has the following limitations:

1. It is only available within the licensed edition of CatTools (is not available with the Freeware edition).
2. It is a part of CatTools and as such must be installed on the same computer as CatTools. (i.e. you can not take the .dll and use it from another computer to access CatTools on a remote computer.)
3. The API is limited to database connectivity and accessing of the device and device related data. There are currently no activity related public classes available within the API to manage activities.
4. Because the API is directly modifying the device data within the CatTools database (i.e. the underlying device values used within an activity), it would always be recommended that the CatTools scheduler timer has been stopped within the CatTools UI and that no activity is currently running before the API code/script is executed. Failing to do so may result in undesirable results.

5. Starting and stopping of the CatTools scheduler timer has not been exposed in the API.

## 11 Installation

This section outlines some tips on installation procedures for CatTools.

- [Automating the installation of CatTools](#)
- [Problems after installing or upgrading](#)

### 11.1 Automating the installation of CatTools

It is possible to automate the installation and startup of CatTools without the need for any human interaction.

To install and start CatTools as a standard interactive application you will need to create a batch file that contains the following information;

```
"*AppPath\CatTools_X.X.X.exe" /S INSTALL=APP /D=*InstallPath
"*InstallPath\CatTools.exe"
```

So your bat file will look similar to;

```
"C:\CatTools_3.5.0.setup.exe" /S INSTALL=APP /D="C:\Program Files\CatTools3"
"C:\Program Files\CatTools3\CatTools.exe"
```

To install and start CatTools as a Windows service you will need to create a batch file that contains the following information;

```
"*AppPath\CatTools_X.X.X.exe" /S INSTALL=SERVICE /D=*InstallPath
"*InstallPath\CatTools_Manager.exe"
```

So your .bat file will look similar to;

```
"C:\CatTools_3.5.0.setup.exe" /S INSTALL=SERVICE /D="C:\Program
Files\CatTools3"
"C:\Program Files\CatTools3\CatTools_Manager.exe"
```

### 11.2 Problems after installing or upgrading

If you are having problems after installing or upgrading CatTools, then please see our [Knowledge Base article](#).

If you still require help, then please request support via [thwack](#), the SolarWinds online

community site.



# Index

## - A -

add device 158  
add new device 158  
Add schedule 166

## - B -

backup databases 136  
back-up databases 136

## - C -

CatTools 1  
change an existing schedule 166  
change device 163  
change password 136  
copy device 163  
copy existing device 163  
copy existing schedule 167  
copy schedule 167  
create schedule 166

## - D -

delete device 163  
delete schedule 167  
Disclaimer 6  
display all devices 165  
display selected devices 163

## - E -

edit device 163  
edit existing device 163  
edit existing schedule 166  
edit schedule 166  
Email 142  
email settings 142  
e-mail settings 142

## - F -

Features 1  
Feedback - Comments or Bugs 6  
file logging 145  
filter device display 163  
filter group 163  
filter host name 163  
filter name 163  
filter type 163

## - G -

Getting started 4, 6

## - H -

HowTo 4, 6

## - I -

import devices from file 131

## - L -

Logging 142

## - M -

misc 142, 146  
miscellaneous 142, 146

## - N -

new schedule 166

## - R -

remove all filters 165  
remove device 163  
remove schedule 167  
run schedule immediately 167

## - S -

selecting devices    53  
show all devices    165  
Software License Agreement    6  
start timer    167

## - T -

Terminate orphaned activity    146  
TFTP    142