

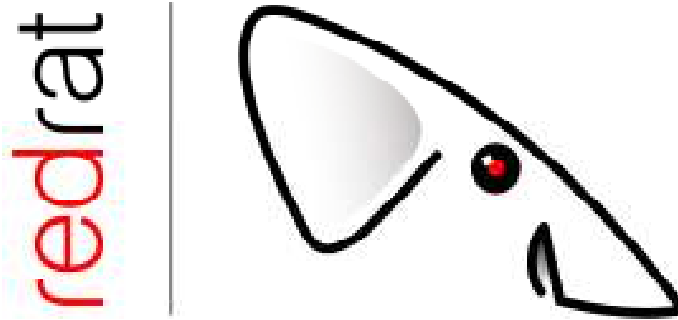
# TestManager – User Guide

Chris Dodge – RedRat Ltd

[www.redrat.co.uk](http://www.redrat.co.uk)

22 January 2008

For TestManager Version 2.13



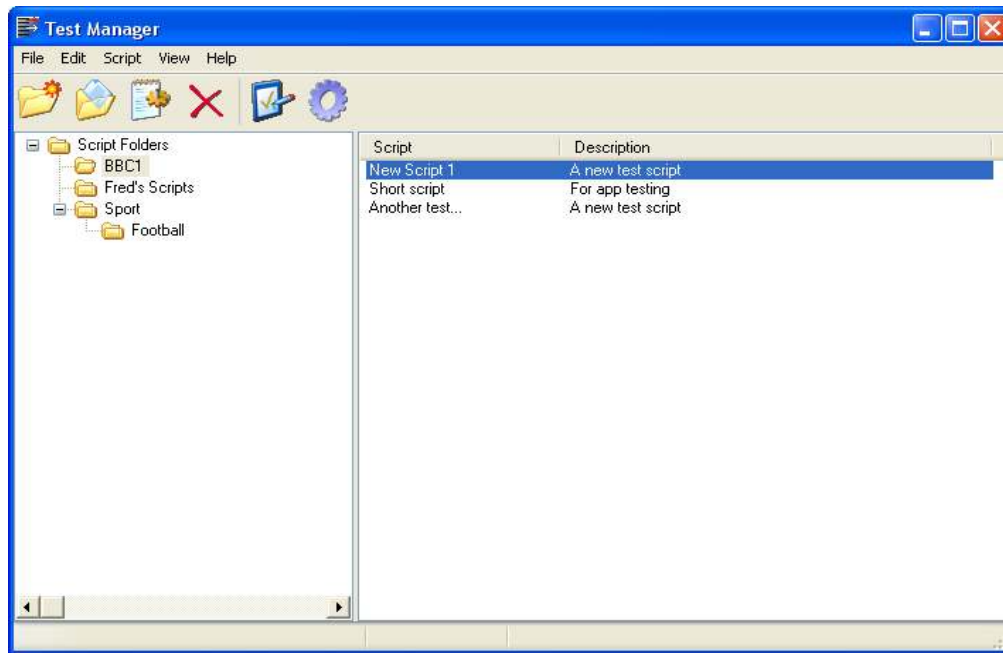
# Contents

1.	Introduction.....	3
2.	TestManager Setup Overview.....	4
3.	TestManager Roles.....	5
4.	Interactive STB Control.....	5
4.1	On-Screen Remotes.....	5
4.1.1	Select Zone.....	5
4.1.2	Select STBs.....	5
4.1.3	Use All STBs.....	5
4.2	Graphical Representation of the STB Layout.....	6
4.2.1	Single STB Mode.....	6
4.2.2	Multi- STB Mode.....	6
4.2.3	Select All STBs.....	6
4.2.4	Selecting a Zone.....	6
5.	Zones.....	7
6.	IR Signal Datasets.....	7
7.	Macros.....	8
8.	Named Operations.....	9
9.	Control of STBs with a Remote Control Handset.....	10
10.	Scripts.....	11
10.1	The Script Language.....	12
10.1.1	Select.....	12
10.1.2	Send.....	12
10.1.3	Wait.....	12
10.1.4	Loops.....	12
10.1.5	Calling Scripts from Scripts.....	13
10.1.6	Comments.....	13
10.1.7	Script Example.....	14
10.2	Script Management.....	14
10.2.1	Script Editing.....	14
10.2.2	Script Validation.....	15
10.2.3	Script Execution.....	15
10.2.4	Script Capture.....	16
11.	Log Output.....	18
12.	Initiating Script Execution from Other Applications.....	19
12.1	TMSendCommand.exe.....	19
12.1.1	-script <script name>.....	19
12.1.2	-verbose.....	19
12.1.3	-help.....	19
12.2	Getting Started with TMSendCommand.exe.....	19
12.3	TMSendCommandW.exe.....	20
12.4	Successful Script Execution via TMSendCommand(W).....	20

# 1. Introduction

TestManager is an application to support automated testing of consumer electronics equipment and interactive or digital television services through the output of infrared remote control command sequences by computer. Building on the ability to generate remote control signals via the USB *RedRat3* and/or the networked *irNetBox*, repetitive or complex, manual testing operations can be replaced by TestManager scripts.

The concept of using scripts to automate sequences of remote control operations is quite intuitive, but the power of the TestManager application lies in its ability to execute the same scripts on banks of different equipment. The key to this is the correct configuration of the test hardware and software, which is discussed in detail in the *TestManager Administration Guide*.



**Figure 1. The TestManager main screen.**

Figure 1 shows the main window for the management of scripts using an “explorer” like interface, which should be familiar to most Microsoft Windows users. It presents a view of scripts organized within folders and stored in the TestManager database. This guide discusses the detail of creating, editing, validating and executing scripts, however there are several concepts and program features that need to be introduced first.

**Please note:** This document discusses how to control set-top boxes (STBs) with TestManager, however any type of audio-visual equipment that uses IR remote control can be used in the place of STBs.

## 2. TestManager Setup Overview

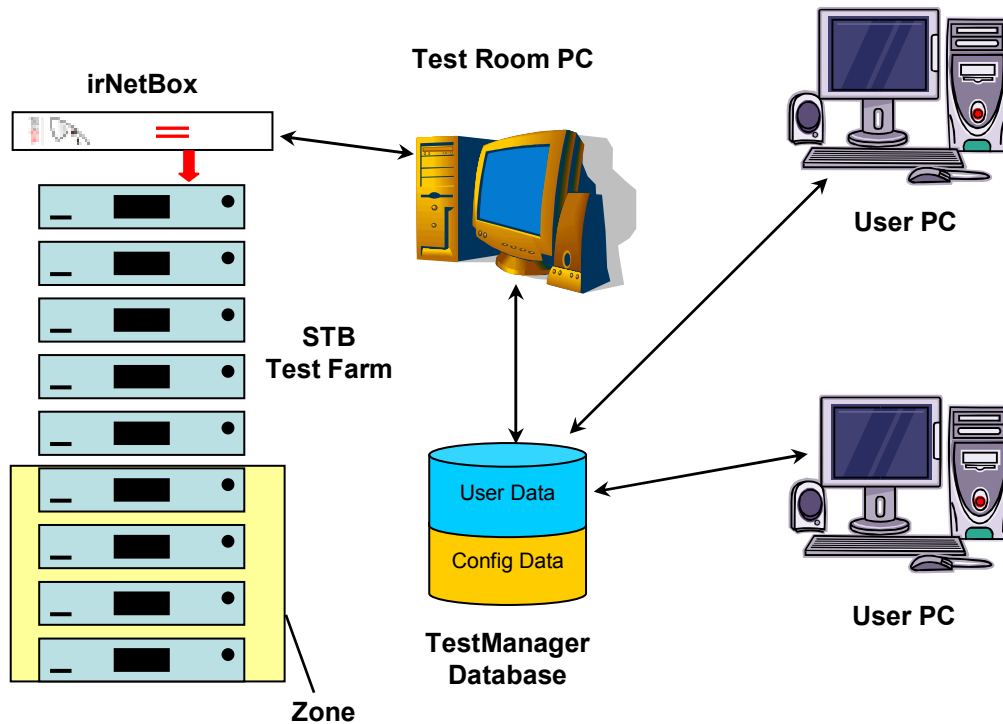


Figure 2. TestManager architecture

The TestManager application uses a typical client-server architecture; the server being the database and the client application running on one or more PCs. In addition, the test setup requires one or more pieces of RedRat hardware (irNetBox or RedRat3) configured to control the equipment under test as shown in Figure 2. The elements of the setup are:

**Test Farm:** The set-top boxes or other consumer electronics equipment under test.

**irNetBox (or RedRat3):** RedRat hardware that sends infrared remote control signals to control the equipment under test.

**TestManager Database:** Holds all configuration and user data for the TestManager application. This is typically installed on the test room PC, but it can be installed on any suitable computer. It uses the MSDE database engine from Microsoft, which is basically a desktop version of SQL Server.

**Test Room PC:** The computer in the test room or near the equipment under test. In each installation, only one PC is able to actually execute the tests so as to prevent multiple concurrent tests being executed simultaneously.

**User PC:** Any other PC on the network that has the TestManager application installed and that can access the TestManager database. Test scripts can be setup and verified on user PCs, but not executed.

**Zones:** A subset of the full test farm for the case that tests are to be run on just certain pieces of the available equipment.

### 3. TestManager Roles

There are two roles when using TestManager: **Administrator** and **User**. The administrator is responsible for installing the irNetBoxes, connecting them to the STBs under test and then configuring TestManager so that it knows the mapping from irNetBox output to STB. Generally, this need only be done once, however the addition or removal of any STBs will require changes to the configuration. Regular backups of the TestManager database are also the responsibility of the administrator. Please see the *TestManager Administration Guide* for administration details.

Users can create, validate and execute run test scripts, but do not usually have permission to change the test configuration.

### 4. Interactive STB Control

Although *TestManager* has been designed for automated testing and running test scripts, it has some very useful features for direct, interactive control of the set of STBs.

#### 4.1 On-Screen Remotes

Depending on system configuration and requirements, TestManager will have one or more on-screen remote controls available. When first using TestManager in a test facility, an on-screen remote control is a good place to start; during application configuration it can help to validate system setup, and can also be used to quickly test remote control signal output sequences. The on-screen remote is started from the *View* → *On Screen Remote* menu item, an example being shown in Figure 3.

When the on-screen remote is moved around the screen, it will “stick” to the edge of the screen, which also prevents it being moved beyond the screen edge. In some situations this behaviour may not be wanted, for example, when multiple monitors are being used, so can be disabled in the *Options* dialog box (*Edit* → *Options* menu item and *Misc* tab).

Clicking on the buttons will cause output of the associated remote control signal to all set-top boxes known to the system by default. If the IR signal associated with a remote control button is not correctly named (E.g. ‘*Select*’ instead of ‘*OK*’) then this can be changed by someone with *TestManager* administrator permissions. The section **Changing IR Signals Output via Virtual Remote Control** in the Administration Guide provides further information.

The set of STBs or Zone to which the IR signal is sent when a button is pressed can be changed by *right-clicking* on the remote to bring up a context menu supporting the following options:

##### 4.1.1 Select Zone

If zones have been configured (see section 5) then one can be chosen and IR signals will be sent to STBs in that zone only.

##### 4.1.2 Select STBs

A dialog box listing all STBs is displayed, and the STBs to be controlled can be individually selected from the list.

##### 4.1.3 Use All STBs

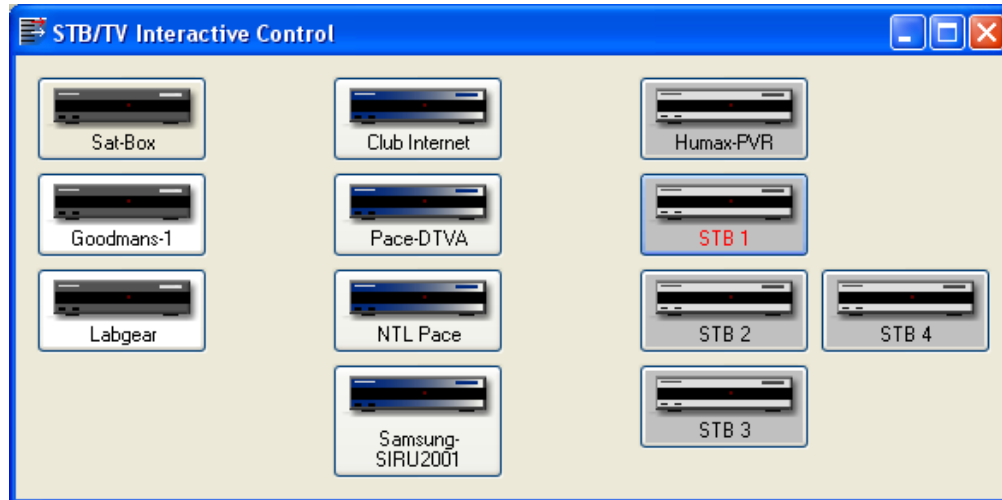
Returns the remote control back to its default behaviour of sending IR signals to all STBs.



**Figure 3. An example on-screen remote control.**

## 4.2 Graphical Representation of the STB Layout

A graphical representation of the test system can be created, and by clicking on the STB icons, a virtual remote control is shown which will give direct control of that STB. This can be an efficient way of interactively working with a large number of STBs.



**Figure 4. The Graphical Representation of STB Configuration**

For information about configuring the panel, please see the *TestManager Administration Guide*. Once configured, it will show your STBs graphically, similar to that shown in Figure 4.

Clicking on any STB will bring up the on-screen remote control associated with that STB and *TestManager* will route IR commands from that remote control to the selected STB only (highlighted in red). There are several STB selection modes, and the user can switch between them by right-clicking to bring up a context menu:

### 4.2.1 Single STB Mode

Allows the control of a single STB only. When another STB is clicked, the on-screen remote control will then be changed to control the newly selected STB.

### 4.2.2 Multi- STB Mode

Successive STB selections will be added to the list of STBs to control, so that a group of STBs can be quickly selected and controlled with a single on-screen remote. There is one limitation, which is that all selected STBs must have been configured to use the same on-screen remote control. If an STB is selected that uses a different on-screen remote, then a warning will be shown and selection of that STB will not be allowed.

### 4.2.3 Select All STBs

All STBs shown in the STB Layout window are selected for IR output using this option. It is then possible to click on any STB to deselect/re-select them, which will also put the window into Multi-STB Mode. The on-screen remote control that is used to control the STBs with this option is defined in the *Options* dialog, in the *Misc* tab (*Edit* → *Options* menu item).

### 4.2.4 Selecting a Zone

All zones that have been defined in the system (see section 5) are shown as sub menu items, and when selected will highlight all STBs in the STB Layout window that are part of the selected

zone. The user will be warned if some of the member STBs of the given zone are not shown in the STB Layout window as it means that not all of the STBs in the zone will have IR signals sent to them. The on-screen remote control that is used to control the STBs with this option is defined in the *Options* dialog, in the *Misc* tab (*Edit* → *Options* menu item).

## 5. Zones

It can be useful in many situations to split a test facility into different zones, for example some operations may fail on a few STBs so they need to be tested in more detail on these few or the test facility may support several testers working at the same time on STB subsets.

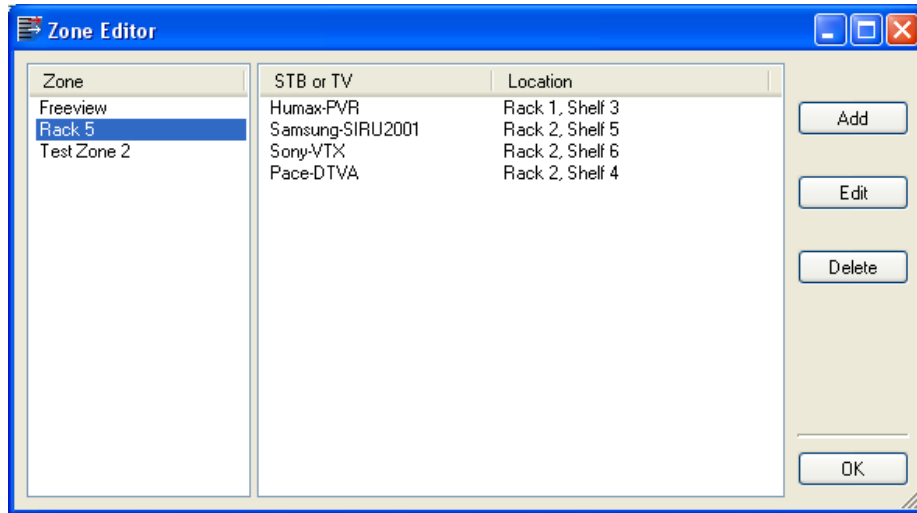


Figure 5. The zone editor.

### The zone editor, shown in

Figure 5, is opened from the *View* → *Zones...* menu item. When a zone is created or edited, a dialog box listing all STBs is shown, allowing the user to add and remove STBs to create the required subset.

**Please Note:** It is possible to delete zones even though they may be used in scripts. Therefore, only delete zones if it is definitely known that the zone is no longer required.

## 6. IR Signal Datasets

The list of IR signal datasets contained within the TestManager database can be viewed from the *View* → *Signal Datasets...* menu item as shown in Figure 6.

Each entry in the main view represents the IR signal data captured from one remote control, and if clicked will open to reveal the individual signals for that remote. It does not support addition or editing of the signal datasets.

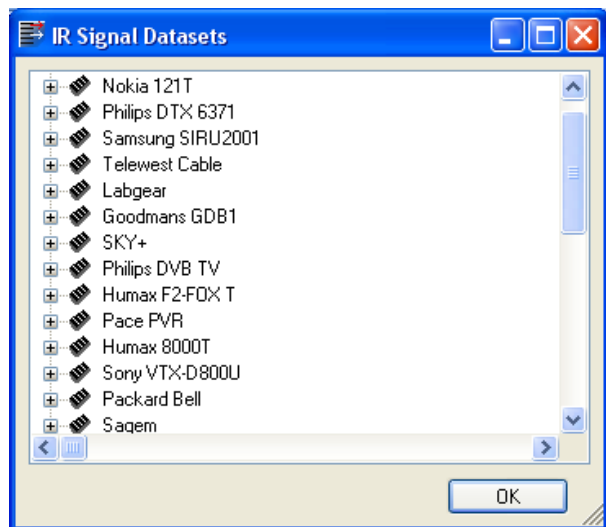


Figure 6. Signal dataset viewer.

## 7. Macros

A macro is a sequence of IR signals that causes some discrete operation to happen on an STB. These are frequently used when changing to a certain channel as most STBs require a two or three digit number to be transmitted, i.e. two or three discrete IR signals.

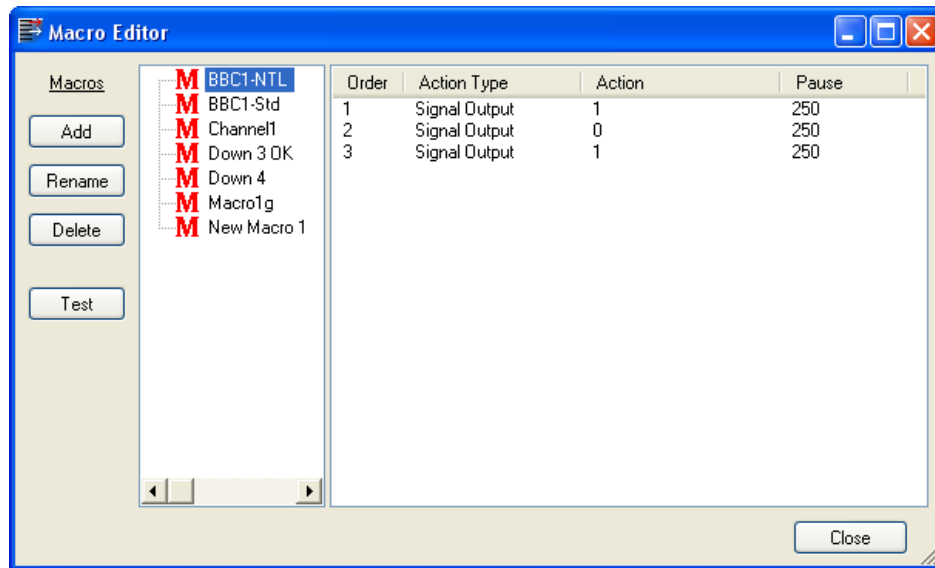


Figure 7. The macro editor.

Figure 7 shows the macro editor which is opened from the *View* → *Macros...* menu item with macros on the left-hand side and the sequence of IR signals on the right hand side. The example macro shown is the IR signal sequence to turn a UK NTL set-top box to BBC1, i.e. sending the signal sequence 1, 0, 1.

To create a macro, take the following steps:

1. Use the *Add* button to create a new macro.
2. Press *Rename* and give it an appropriate name.
3. In the right-hand pan, right-click and select *Add Action*, which will bring up the action dialog box shown in Figure 8.
4. Select the IR signal name from the *IRSignal* drop down menu. The list of signals shown is pre-configured, so please contact your TestManager administrator if you require signals other than those listed.
5. Following the output of IR signals there should be a short pause so that the STB is able to detect the end of one signal and the start of the next. The default value is 250ms, but this can be changed to any required value.

Once complete the macro can be tested using the *Test* button. TestManager needs to know to which STBs to send the macro, so an STB selection dialog will be shown so that one or more can be chosen.

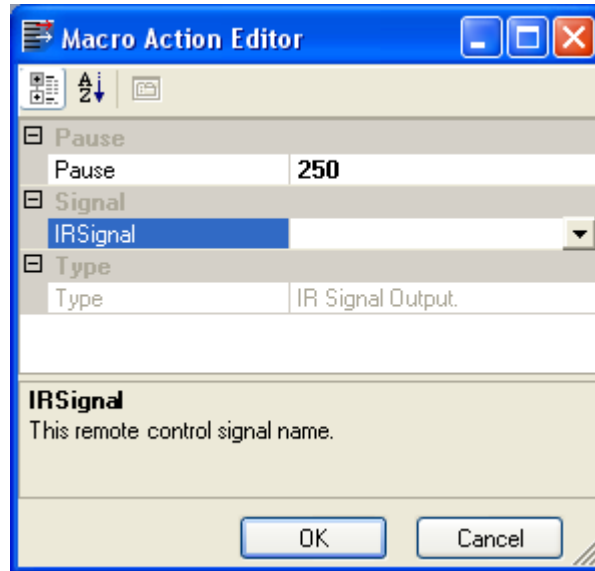


Figure 8. Dialog box for adding a signal to a macro

## 8. Named Operations

The principle behind named operations is that there are operations with clearly understandable name (e.g. change channel to **BBC2**) but which require different sequences of IR signals for different STBs. They build on the concept of macros, but have one additional level of abstraction so require additional configuration steps. Once configured, the output of a named operation to a set of STBs will cause them all to perform the same operation, for example, the named operation *BBC2* will switch all STBs to channel BBC2 regardless of the sequence of IR signals required to accomplish this.

**Error! Reference source not found.** shows the named operation editor with the *BBC2* named operation selected. For each named operation, all STBs are listed in the right hand pane with an associated macro. When the named operation is executed, TestManager iterates through the list of STBs, extracts the macro for that STB then sends it to that STB.

To make the configuration process somewhat easier, it is assumed that each named operation has a default macro (*BBC2-Std* in the example shown), however for STBs requiring a different sequence a different macro can be set (e.g. *BBC2-NTL* for the one of the STBs). To set a different macro for an STB, right-click on the STB and select a macro. The right-click context menu can also be used to set an STB back to the default macro.

Named operations can be tested in a similar way to macros using the *Test* button.

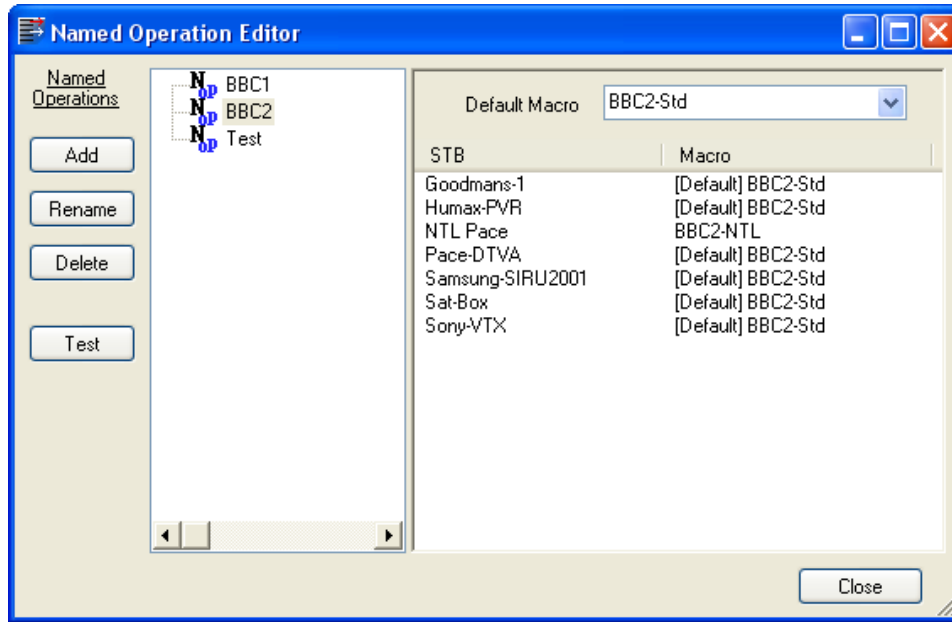


Figure 9. The named operation editor.

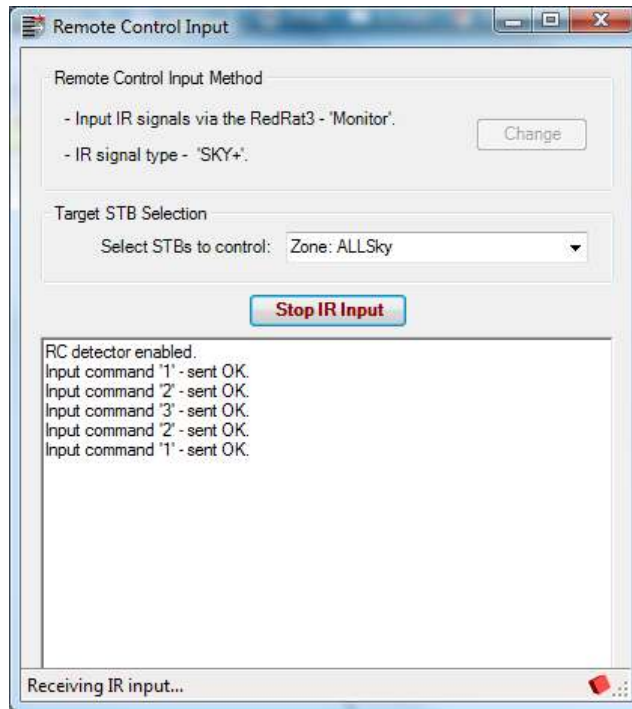
## 9. Control of STBs with a Remote Control Handset

This functionality supports the control of STBs via a remote control handset so that when configured correctly, the selected set of STBs will follow the commands given on the remote. *TestManager* does this by recognizing the IR command sent from the handset, and then mapping that to the correct IR signal for each STB. The following prerequisites are needed to use this method of control effectively:

- A RedRat3 USB device dedicated to IR input must be attached to the computer on which *TestManager* is running.
- A remote control with IR signals that can be reliably recognized by *TestManager*. This can be tested using the Signal DB Utility, and in the UK, Sky remotes are one example of a type that work well.
- When sending IR commands to the RedRat3, the user must make sure that these commands are not directly received by the STBs, so the RedRat3 and remote control must be shielded.

Configuration is done in the options dialog (*Edit* → *Options* → *Control Via Remote tab*). Here the RedRat3 for IR input is selected, and the IR signal dataset that corresponds to the physical handset to be used. If the RedRat3 does not appear in the drop-down list, then it is not yet “known” by the system, so needs to be added using the *RedRat Hardware Configuration* dialog (*Edit* → *Configure* → *RedRat Hardware...* menu item).

To start control via your remote control handset, open the control dialog (as shown in Figure 10) from the *View* → *Control STBs via Remote...* menu item.



**Figure 10. Window used to control STBs with a remote control handset.**

Firstly, select which STBs are to be controlled using one of the options in the *Target STB Selection* drop-down list. The choices here are;

**All STBs** – IR commands will be sent to all STBs known by the system.

**Select from STB List** – Shows the standard STB selection dialog, allowing selection of any combination of STBs.

**Select from STB Layout window** – This pops up the STB Layout window (see section 4.2), and either single or multiple STBs can be selected here.

**Zones** – All zones are listed here, so can be selected. See section 5 for more details on zones.

Once the set of STBs to control has been selected, IR input can be enabled with the “*Start IR Input*” button, which instructs the RedRat3 to start sending IR data to *TestManager* for interpretation and output to the selected STBs.

The Remote Control Input window reports input commands, and whether the corresponding command has been successfully sent to all STBs. More detailed information about exactly which signals are sent, whether the system has recognized the input signal etc. can be viewed in the log window (from the *View* → *Log Output* menu item).

## 10. Scripts

The features listed in the previous sections are all building blocks to support the simple creation of scripts. In this section, the script language will be introduced followed by their use within TestManager.

## 10.1 The Script Language

### 10.1.1 Select

Selects the target STBs for the following script operations where a single STB can be chosen, a zone or the keyword ALL for all devices in the test facility.

```
select [STB | Zone | 'ALL' ]
```

Examples:

- `select humax-pvr`
- `select zone-3`
- `select all`

Any send operations following a select will be sent to the subset of STBs selected until a new select statement is encountered. By default all STBs are selected, i.e. if no select command is given at the beginning of a script, a `select all` assumed.

### 10.1.2 Send

Outputs a signal, macro or named operation to the selected STBs:

```
send [signal | macro | named_operation]
```

Examples:

- `send play`
- `send 'BBC2 Std'`
- `send BBC2`

If a macro or named operation name contains white space, then it has to be surrounded by either single or double quotation marks.

So how does TestManager know whether a name given in a send command corresponds to a signal, macro or named operation? This is done through enforcing the following rules:

1. When a macro is named, it may not have the name of any of the pre-configured signal names.
2. When a named operation is named, it may not take the name of a signal or any macro.

### 10.1.3 Wait

Inserts a delay in the execution of a script of a given number of seconds:

```
wait N
```

Examples:

- `wait 1`
- `wait 360`

### 10.1.4 Loops

Blocks of script instructions can be repeated using loop constructs:

```
loop N  
  <instruction block>
```

```
end loop
```

where **N** is a positive whole number. Loops can be nested within loops, forming constructs as shown in the example below:

```
loop 3
  loop 4
    send down
    wait 2
  end loop
  send OK
end loop
```

**Note:** Indentation is for clarity only.

If an indefinite test is to be setup, then an INFINITE loop can be created, for example:

```
loop INFINITE
  <instruction block>
end loop
```

### 10.1.5 Calling Scripts from Scripts

A second script can be called from a main script, so providing a mechanism for structuring test operations in a modular fashion. For example, a commonly used script section can be extracted and placed in a script file of its own, and then called from other scripts when necessary.

The following example calls two scripts that are found in the same folder as the calling script. Quotation marks need to be used round script names that contain white space.

```
call reset
call 'basic test'
```

If a script to be called is found in a different folder, the full folder path should be given, as shown below:

```
call 'STBs\basic function\reset'
```

Logically, the use of the **call** keyword is identical to inserting that script section within the calling script. As a result, any **send** instructions given in the sub-script will be sent to the STBs most recent selected with the last **select** command, whether that is in the main script or the sub-script.

**Please Note:** It is not recommended that a script calls itself, or that sets of scripts that call each other in an infinite sequence are setup. The system does not currently check for this, and the recursive nature of these operations will eventually lead to a stack overflow exception.

### 10.1.6 Comments

Comments can be inserted anywhere into scripts using a '#'. All text between the # and the end of the line is treated as a comment.

### 10.1.7 Script Example

```
#
# Script example
#
select all
send BBC1
wait 5
sent red
wait 5
loop 3
  loop 4
    send down
    wait 2
  end loop
  send OK
end loop
```

## 10.2 Script Management

As described in the introduction and shown in Figure 1, scripts are stored in the TestManager database and organized within a folder structure. Folders and scripts can be created, edited and deleted from the *Edit* and *Script* menu items, or from the toolbar as shown in Figure 11.

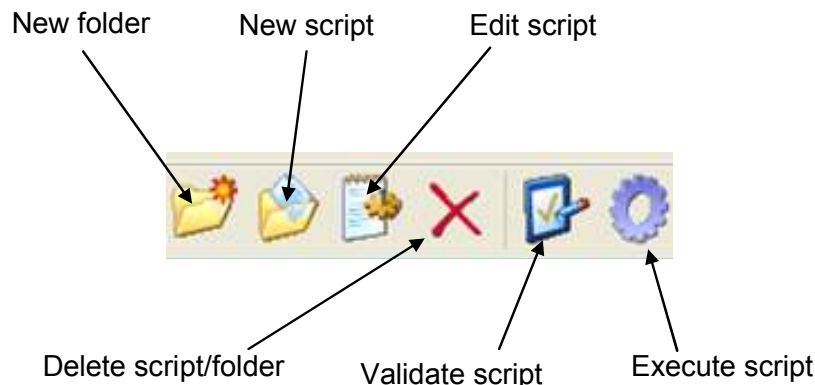


Figure 11. Toolbar button functions.

### 10.2.1 Script Editing

A script is edited by either double-clicking on it in the right-hand pane of the TestManager window or by selecting a script and using the *Edit Script* toolbar icon or *Script* → *Edit Script* menu item.

TestManager supports script editing by writing the script to a temporary file on disk, then opening a text editor into which the script is loaded. By default, *notepad.exe* is used, but other test editors can be configured in the TestManager options dialog (*Edit* → *Options...* in the *Script Editor* tab). The only requirement when using a different text editor application is to ensure that it takes the name of a file as its first parameter.

Once editing is finished, save the file and close the editor. TestManager will then read back the file and store it in the database. **Note:** Don't save the file under a different name (unless you are

making a backup copy on disk) as TestManager reads back from the same file it created so will not read back any changes if they are written to a different file.

## 10.2.2 Script Validation

TestManager can run through the script and check for the following errors:

- That the script syntax is correct, e.g. keywords spelt correctly, loop starts and ends matched up.
- STB and zone names recognized.
- Signal, macro and named operation names recognized.
- For IR signal output instructions (SEND <IR signal>) the particular signal is found in the datasets for the selected STBs.

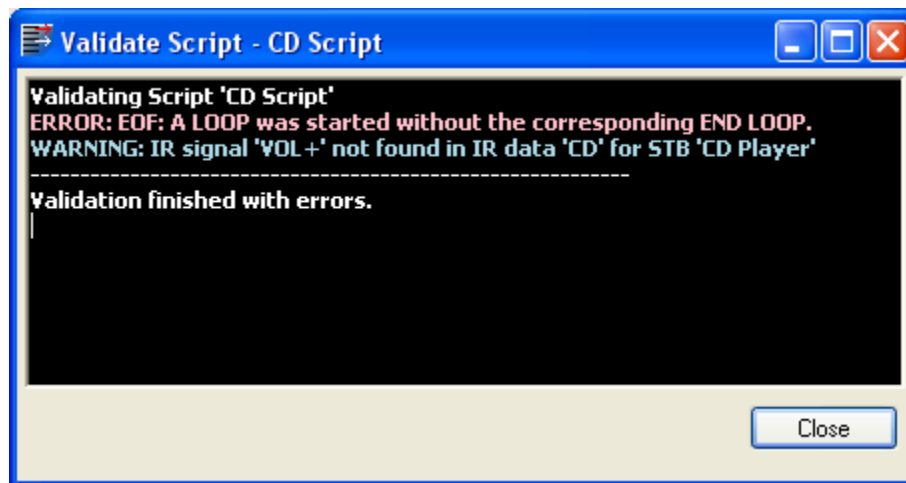


Figure 12. Output from script validation, showing errors.

Figure 12 shows typical output from validation on a script with errors. A script can be validated on any PC on which TestManager is installed so that scripts can be created and validated in preparation for access to the test facility.

Warnings and errors are treated slightly differently (and also printed in different colours). An error prevents the script from being executed, for example incorrect script syntax, whereas a warning will not stop the script being executed but it may not produce exactly the desired effect.

## 10.2.3 Script Execution

In a standard test facility setup, script execution can usually only take place from the PC within the test facility. This is to prevent other TestManager users accidentally initiating script execution from PCs elsewhere on the network, so interfering with the test facility operation. Multiple scripts can be run simultaneously.

Script execution can be started from the *Execute Script* toolbar icon or the *Script* → *Execute Script* menu item or by right clicking on a script. Before the script is executed, it is validated and if no errors are found execution will start. In addition, it will reserve all set-top boxes to be used in the execution of the script so that if another script is started, it cannot use any reserved STBs, hence will not interfere with script execution.

Figure 13 shows the script progress dialog box, the main part showing each instruction as it is executed, including the loops so that script progress can be monitored. The lower part of the window shows the actual IR output log so that every output IR signal can be tracked, including IR signals that can't be output. (See next section for more information on logging.)

The *Stop* button can be used to prematurely terminate the script's operation if necessary and the *Hide Log* button to hide the bottom section of the window.

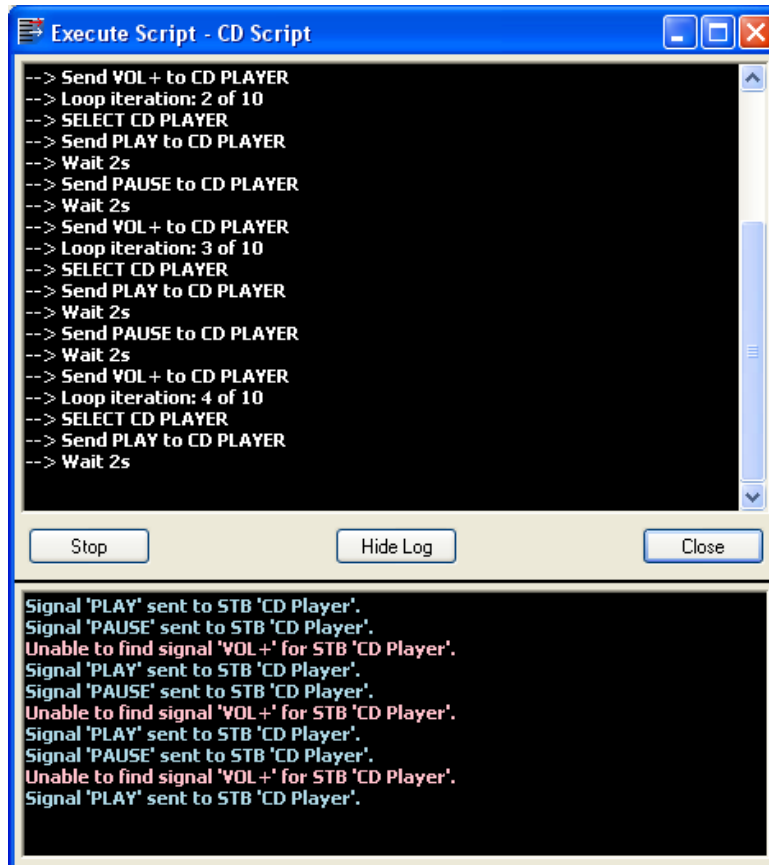


Figure 13. Script execution output .

## 10.2.4 Script Capture

Section 10.2.1 describes how a script can be created manually using a standard text editor. It is also possible to capture a script from an on-screen remote control or an actual remote control using a USB RedRat3 device for IR signal input. Once a script has been selected in the main *TestManager* window, the script capture dialog can be opened from the *Script* → *Capture Script From Remote* menu item.

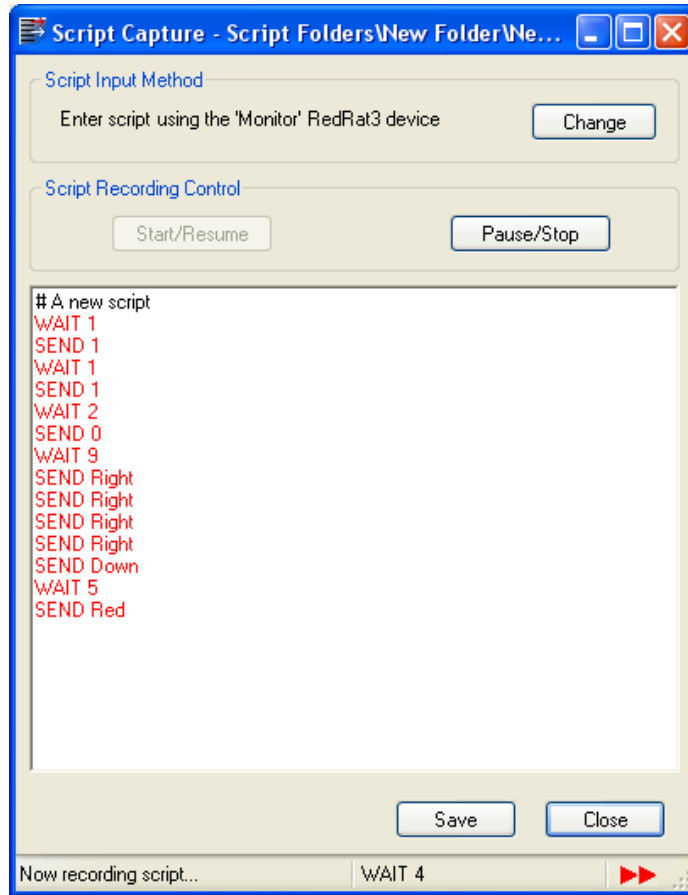


Figure 14. The Script Capture Dialog

#### 10.2.4.1 Configuring the Script Input Method

Before script capture can be started, the capture input method has to be setup in the *TestManager* Options dialog box. To open the options box, either press the *Change* button in the Script Capture box or use the *Edit* → *Options* menu item.

Either an on-screen remote control or a RedRat3 device can be selected as the input method, and if a RedRat3 is chosen, then an IR dataset also has to be selected. This IR dataset is used for input signal recognition so that the appropriate remote control button presses can be inserted into the script.

**Using the On-Screen Remote:** If the on-screen remote is being used for script capture, it can also be used to simultaneously output IR signals so that the result of the buttons presses are seen in real-time. To select which physical STBs/TVs you would like to control, right-click on the on-screen remote and select either a Zone, a set of STBs or all STBs. The “*No IR Output*” menu item can be used to turn off any IR output if it is no longer required.

If it becomes unclear which IR signals are being sent, then the log window can be opened from the *View* → *Log Output* menu item. All output signals will be reported, and also any problems such as certain IR signal data or RedRat hardware not being available.

**A Note on IR Signal Recognition via a RedRat3:** Due to the large variation in IR signal types, not all IR signal datasets are reliably recognized by the general purpose signal recognition algorithm built into *TestManager*. If in doubt, it may be worth experimenting with the Signal

Database Utility which gives direct feedback when IR signals are detected by the RedRat3. Please contact RedRat support for further details or the development of custom signal recognition code if your remote control is not recognized by the general purpose signal recognition algorithm.

If you suspect that IR signals are not being recognized, open the log output window (*View* → *Log Output* menu item) as this will print out each IR signal that is not recognized.

### **10.2.4.2 Recording a Script**

To start recording a script, press the *Start/Resume* button. Depending on the selected input method, either the buttons on the physical remote control or on the virtual remote control can be pressed. The button/signal names are then appended to the script in red.

While a script is being captured, a timer counts the number of seconds between each button press and inserts the given pause into the script. The running timer value is shown in the status bar at the bottom of the window.

### **10.2.4.3 Pausing/Stopping a Recording**

The *Pause/Stop* button can be used at any time to stop script recording. Recording can be stopped/started as often as required.

### **10.2.4.4 Saving a Script**

The *Save* button will write the script back to the database. All newly captured text that is shown in red will turn black to indicate that it has been saved. When the script capture dialog has been closed, the script can be manually edited in the normal manner if required.

## **11. Log Output**

When a script is executed, the information presented shows how the script is progressing, but if an STB is not reacting correctly, it can be difficult to work out exactly why not. The *TestManager* log of IR signals that are output to each STB can help to track the exact sequence of operations. As shown above, the log output can be viewed with script execution, but it can also be sent to a file as well.

File logging is configured from the *Logging* tab in the options dialog box, from the *Edit* → *Options...* menu. If file logging is enabled, then a directory for the log files needs to be set in which *TestManager* will write the log files. The log output from each script executed is written to a file with the same name as the script, and in a directory structure that exactly reflects the *TestManager* script folder structure.

Each entry in a log file is given a time stamp, producing output similar to that shown below:

```
[15:36:35] SCRIPT EXECUTION STARTED
[15:36:35] -----
[15:36:39] Signal 'PLAY' sent to STB 'CD Player'.
[15:36:41] Signal 'PAUSE' sent to STB 'CD Player'.
[15:36:43] Unable to find signal 'VOL+' for STB 'CD Player'.
[15:36:43] Signal 'PLAY' sent to STB 'CD Player'.
[15:36:46] Signal 'PAUSE' sent to STB 'CD Player'.
[15:36:48] Unable to find signal 'VOL+' for STB 'CD Player'.
[15:36:48] Signal 'PLAY' sent to STB 'CD Player'.
[15:36:50] Signal 'PAUSE' sent to STB 'CD Player'.
[15:36:52] Unable to find signal 'VOL+' for STB 'CD Player'.
[15:36:52] Signal 'PLAY' sent to STB 'CD Player'.
[15:36:55] Signal 'PAUSE' sent to STB 'CD Player'.
[15:36:57] Unable to find signal 'VOL+' for STB 'CD Player'.
[15:36:57] Signal 'PLAY' sent to STB 'CD Player'.
```

```
[15:36:59] Signal 'PAUSE' sent to STB 'CD Player'.  
[15:37:01] Unable to find signal 'VOL+' for STB 'CD Player'.  
[15:37:02] Signal 'PLAY' sent to STB 'CD Player'.
```

If a script is re-executed, then the log file is overwritten, so deleting the output from any previous runs.

## 12. Initiating Script Execution from Other Applications

If infrared remote control based tested is only one element of a more complete test environment, then it may be useful to initiate *TestManager* scripts from other, third party applications used in the test system. Two utilities are provided as part of the *TestManager* setup to support this – *TMSendCommand.exe* and *TMSendCommandW.exe*.

A running *TestManager* instance is required; following which one of the above exes can be called by a third party application on the same PC with appropriate parameters. It then communicates with *TestManager*, and initiates the execution of a script.

### 12.1 *TMSendCommand.exe*

This is a console based application, the advantage being that it can provide output from its operations and so give feedback as to whether it has successfully passed the commands to a *TestManager* instance. It is therefore recommended that this application is used for initial experimentation.

Application parameters are:

#### 12.1.1 **-script <script name>**

Instructs *TestManager* to execute the script of the given name. Scripts are identified by the full path name in the *TestManager* script explorer window, for example, the command

```
TMSendCommand.exe -script Sport\Football\Interactive Page1
```

will execute the script called *Interactive Page1* given in the *Sport\Football* folder.

(N.B. The *Script Folders* root folder name can be ignored in the full script path name.)

#### 12.1.2 **-verbose**

*TMSendCommand.exe* will print out each step of its operation, and whether it has successfully sent the command to *TestManager*. *TMSendCommand* does receive any feedback from *TestManager* itself, so to check that *TestManager* has received a valid script name, and that it can execute the script, enable the log output window from the menu item *View → Log Output* in *TestManager*.

#### 12.1.3 **-help**

This will print out the various *TMSendCommand* options available.

## 12.2 *Getting Started with TMSendCommand.exe*

The following procedure is recommended for getting started and to help with the elimination of errors:

1. Start *TestManager*.
2. Enable the log output window from the *View → Log Output* menu item.

3. Open a DOS/Command window, and change directory to the where the TestManager, TMSendCommand.exe etc. have been installed. This is typically

```
C:\Program Files\RedRat\TestManager
```

4. Select a script to execute, add it as a parameter with the `-script` option to the `TMSendCommand` command line, also including the `-verbose` option. For example (on a single line):

```
TMSendCommand.exe -script Sport\Football\Interactive Page1  
-verbose
```

If all goes well, you will get `TMSendCommand` output similar to the following:

```
INFO: Found TestManager application.  
INFO: Sent EXECUTE_SCRIPT to TestManager: Sport\Football\Script2
```

The `TestManager` log window will show:

```
Received TMSendCommand message.  
EXECUTE_SCRIPT instruction: Sport\Football\Script2  
About to execute script: Sport\Football\Script2  
Script 'Script2' about to be started.
```

### 12.3 `TMSendCommandW.exe`

The `TMSendCommand.exe` application is a console application, meaning that it is run from a console window (sometimes called DOS or command windows). If it is executed by another means, for example clicking on it in Windows Explorer, then a console window will be started. This also applies when running it from third party applications.

To solve this, a second version of the `TMSendCommand.exe` is available, called `TMSendCommandW.exe` which is actually a Windows application, but does not create any visible window. When called, this executes the same instructions, but does not cause a console window to be created.

As `TMSendCommandW.exe` does not use console output, and does not create any windows, it does not provide any feedback to the success or otherwise of its execution. Therefore it is recommended that the console version is used to experiment and get the command line parameters correct before using `TMSendCommandW.exe`.

### 12.4 *Successful Script Execution via TMSendCommand(W)*

When `TestManager` receives the script execution command with a valid script name, it attempts to open the script execution window, following which it will validate and execute the script. Just as with interactive script execution from the `TestManager` interface, there are a number of reasons why a script may not be executed or complete successfully:

- The script is already executing or is open in an editor. In which case, the script execution window will not open.
- The script does not validate. This will be reported in the script execution window.
- The script uses STBs/TVs that are being used by another script. This will also be reported in the script execution window.

If successful, script execution will continue to completion, following which the execution window will remain open for a further ten seconds before closing. Once closed, the script can then be re-executed if required.