TestManager – User Guide

RedRat Ltd

March 2020

For TestManager Version 4.90

Contents

1. Introduction

TestManager is an application to support automated and interactive control of consumer electronics equipment using RedRat devices. Automated control takes places via scripts which send sequences of remote control commands to consumer electronics equipment. Interactive control uses on-screen virtual remotes or physical handsets to route commands to selected CE devices.

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 section 4 onwards.

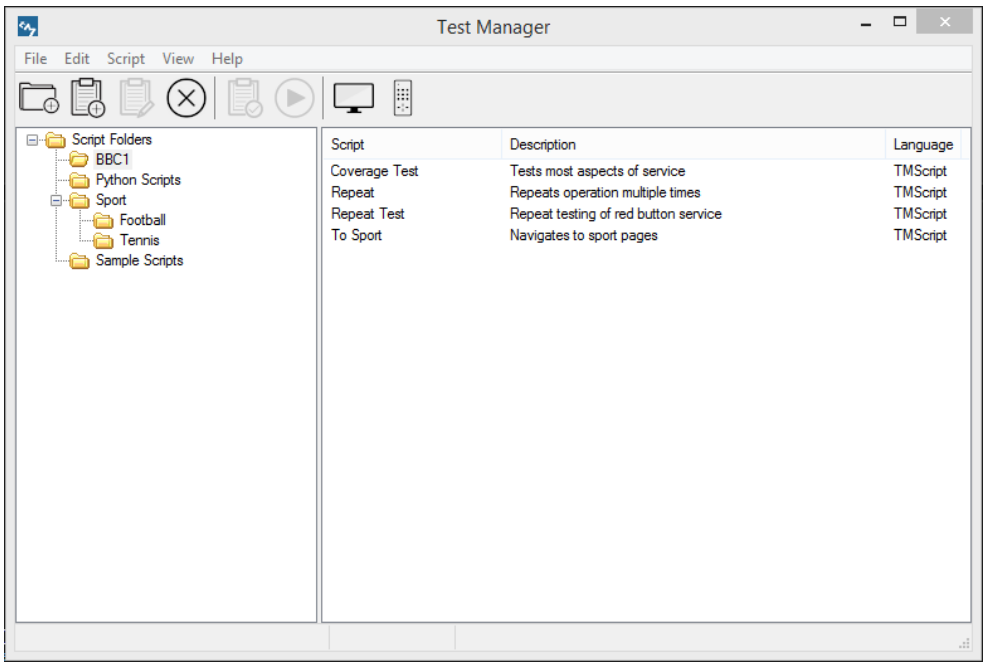


Figure 1. The TestManager main screen.

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.

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

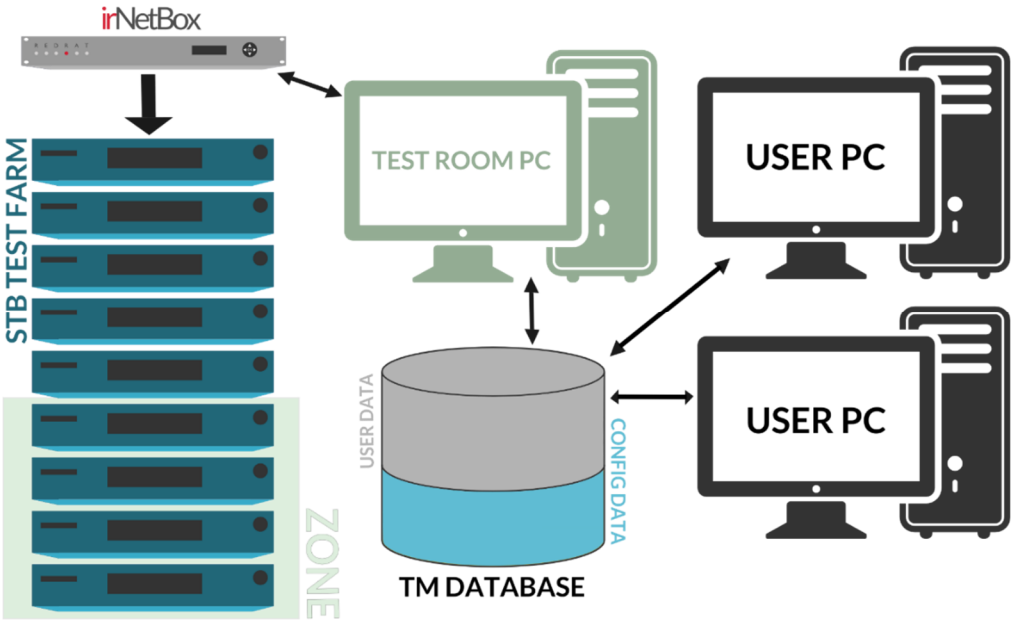


Figure 2. TestManager architecture (multi-user mode)

The TestManager application (multi-user mode) 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 RedRat devices 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 other RedRat device): 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.

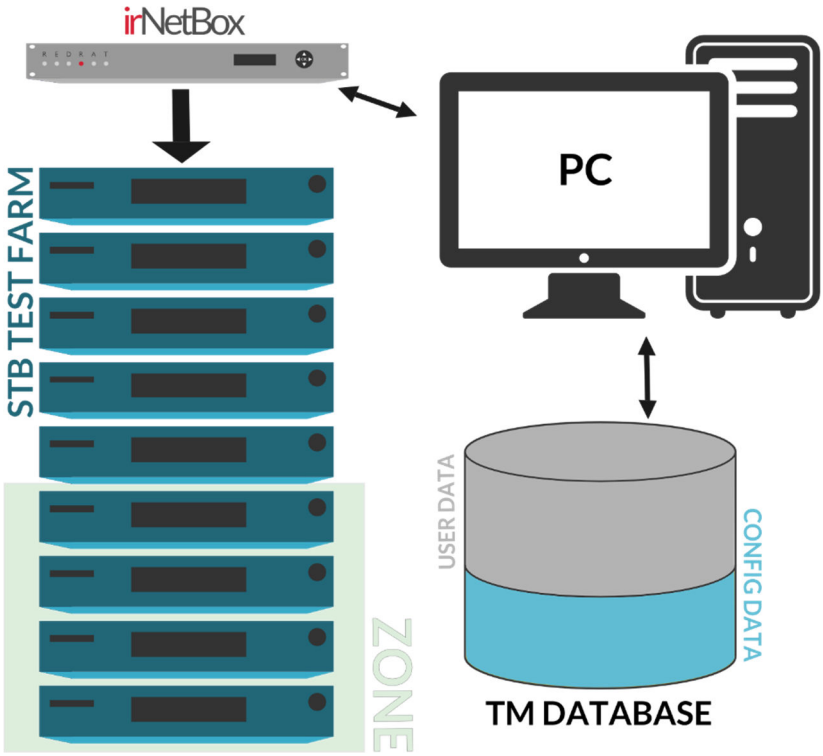


Figure 3. TestManager architecture (single-user mode)

TestManager in single user mode (as shown in Figure 3) has the database stored in a file on the computer on which it runs. This makes installation somewhat easier, but makes the sharing of scripts more difficult.

3. Quick Start Guide

This section is intended to describe the minimum set of steps necessary to get started with the TestManager application. More detailed information is given in later chapters of this guide.

3.1 Initial Configuration

There are a number of settings that need to be configured before you can begin to control your own hardware. These are accessed via *Edit* → *Configure* menu item:

3.1.1 RedRat Hardware

TestManager needs to know what RedRat devices to use, so the *RedRat Hardware Configuration* window is used to add new hardware to TestManager. Upon clicking the *Add* button, TestManager will search for connected RedRat devices, and so those which you intend to use with TestManager can be selected in the *Add RedRat Devices* window. Devices can also be added directly if their IP address is known. (See section 6.1 contains for further information.)

3.1.2 Import IR Data

Assuming that you have already experimented with control of your STBs using RedRat devices, then you will have captured the IR datasets needed using the Signal Database Utility application. The full IR data XML file from the Signal Database Utility should be imported into TestManager via the *File -> Import Signal DB* menu item. (See section 5 for further details.)

3.1.3 STB/TV properties:

The *STB/TV Configuration* window is used to define the set of STBs (or TVs etc.) within TestManager. Clicking the *Add* button will open the edit set-top box window. This window allows you to configure the necessary details so that TestManager can control each STB:

- **Name:** Unique name and identifier of the STB.
- **IR Signal Dataset:** The IR dataset in the imported data (section 3.1.2 to be used with the device.
- **RedRat Device:** Which of devices set in section 3.1.1 is to be used.
- **Output Port:** If a RedRat device with multiple output ports, which is setup to control this STB.

3.2 STB Control

Once configuration is complete, there are two primary methods of controlling your STB's:

3.2.1 Interactive control via the STB layout window.

The STB/TV Configuration Designer allows you to create a graphical representation of the STB setup, for example reflecting their positions in a rack or perhaps how their video output is organized on a multi-viewer.

Use the *Edit → Configure → STB layout* menu item to open the STB Layout designer. Right-click on the background to add one of the STBs known about by TestManager. The properties of the visual STB representation can be changed, such as colour, image, size, position and which on-screen remote control to use for control.

Use the *View -> STB Layout* menu item to open the STB Layout window for STB control. Click on an STB, and the associated on-screen remote control should be shown, which can be directly used for STB control. See section 8 for more detailed information.

3.2.2 Autonomous control via scripting

To automate control of your hardware, new scripts can be added to the TestManager main window. These can be validated and executed from within the main window or via the TestManager script editor. Section 14 gives more detail about scripting.

4. Connection to the TestManager Database

When TestManager is started, it needs to load data from the data store, however the first time that the application is run it does not know where to find the data, so this has to be setup.

4.1 Using SQL Server – Multi-user Mode

In this mode an external database is used, which may be located on another server. Open the *Options Dialog* from the *Edit* menu and then enter the database login information in the *Data Source* tab.

Note that in this mode, multiple TestManager instances could use the same database, so could potentially edit scripts or configuration information. It is recommended that permissions and roles and informally agreed between TestManager users in this scenario.

4.2 Using an Internal Database – Single-user Mode

When TestManager is first run in this mode, it will create a file for storage of your configuration data, signal datasets, scripts etc. The default name is ***TestManagerDB.sdf*** and it is initially located in your application data folder.

To view the file's location, use the *Data Source* tab in the *Options* dialog. It is possible here to:

- Open a different file: Press on the "…" button, navigate to the new file location, select and open it.

- Create a new file: Press on the "…" button, navigate to the location of the new file, enter a file name in the *File name* field and press *OK*.

It is possible to have many different files, although this is not necessarily recommended as single file can store all necessary configuration information.

5. Managing IR Signal Datasets

All IR signal data is held in the TestManager database rather than in an external file. On installation, the database does not contain any IR signal data, so this needs to be added. If you have already used RedRat hardware to capture and output IR data, then the XML datasets can simply be loaded into the database (see section 5.2).

TestManager uses a fixed set of IR signal names as keys to look up the correct IR signals when executing scripts. They need to be consistently named across all IR signal datasets so that when a script looks for a certain signal for a number of STBs (e.g. CHANNELUP) then it can find it in all IR datasets.

For standard STBs the signal names initially configured are:

0	1	2	3
4	5	6	7
8	9	0	POWER
RED	GREEN	YELLOW	BLUE
UP	DOWN	LEFT	RIGHT

OK	MUTE	VOLUP	VOLDOWN
EXIT	CHANNELUP	CHANNELDOWN	TVGUIDE
HOME	INFO	SETUP	MENU
PLAY	PAUSE	STOP	RECORD
FASTFORAWRD	REWIND	STEPFORWARD	STEPBACK

Table 1. Default IR signals names

These signals need to be captured from the remote of the new STB using the Signal DB Utility application and then loaded into the TestManager database.

5.1 Capturing a New IR Dataset

If you have a RedRat3 or RedRatX available, it is easier to use this rather than an irNetBox to capture IR signals (due to better light filtering and perhaps having it in a more convenient location). Before attempting to capture the whole dataset, it is worth experimenting with one or two signals to ensure that the capture and replay works correctly. If it does not appear to work, then please check the RedRat support section on our website or contact RedRat support.

The following steps use the *New Device/Remote Wizard* in the Signal Database Utility to capture a new dataset:

1. Select *New Device/Remote Wizard* from the *Edit* menu in the Signal Database Utility and fill in the fields in the dialog, selecting SET_TOP_BOX as device type.
2. In step 2 of the wizard, adjust the list of signals to be the same as those given in **Table 1** unless the STB does not support this functionality.
3. In step 3 of the wizard, press the *Start Capture* button to initiate input of the signals from the remote.
4. Once complete, save the XML file for use by the TestManager application (and to keep as a backup).

5.2 Inserting the Dataset into the TestManager Database

IR datasets are loaded on top of any data already in the database, however no data will be deleted. Therefore if device A, signal B already exists in the database and the file you are loading also contains device A with signal B, then this will overwrite the database version. In this way it is possible to update signal data in the database if any of the signals in the database don't work effectively.

The signal data XML file can be loaded using the *File* → *Load Signal DB* menu item.

5.3 Adding or Changing the IR Signal Names

TestManager will only recognize certain IR signal names in scripts or when being output from an on-screen remote control, these names being held in the TestManager database. The reason for using a pre-defined set of signal names is twofold:

1. To enforce a standard naming convention across all STB IR signal datasets in an installation.
2. To ensure that signal names, macro names and named operations are all unique.

Table 1 shows the standard IR signal names for STBs in the UK. When TestManager is used with other types of STBs then the set of IR signal names may need to be adjusted, which can be done through the *IR Signal Names* dialog (launched from the *Edit → Configure → IR Signal Names*… menu item) as shown in **Figure 4**.

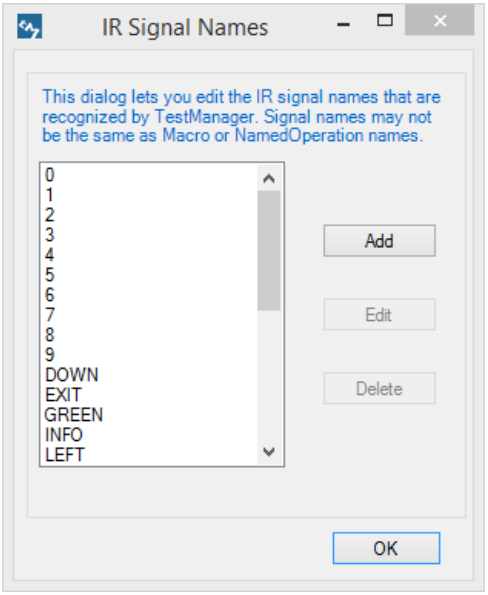


Figure 4. The IR Signal Names dialog.

For example, if you want to control a new STB that has a ***SERVICES*** button on the remote control, then this name would need to be added to the IR signal name list. Once the IR dataset for this remote control has been captured and loaded into *TestManager*, then the ***SERVICES*** signal can be used in scripts, macros and from the on-screen remote control (where supported by that STB).

5.4 Changing Remote-Control Button Commands

The keys on the virtual remote controls each output an IR signal of a given name, however in some environments the IR signal name associated with a key may not be correct. For example, STB remotes use “OK” and “SELECT” interchangeably, but for scripts to be run against all STBs, a choice has to be made as to which of these two is going to be used in scripts.

If the on-screen remote control buttons attempt to output IR signals with the wrong names, then the IR signal names associated with each button can be changed. To do this, right-click on any remote control button and select the *Set Button Action* menu item. This brings up a dialog in which the IR signal name that is associated with that key can changed.

Once changed, the new key mapping is stored in the TestManager database and will then be used whenever this remote control is shown on the screen.

6. Configuring TestManager

This section explains how TestManager is configured to "*know*" about your setup, i.e. what RedRat devices are used to control which STBs. The configuration options described in this section are found under the *Edit* → *Configure* menu item in the TestManager application.

6.1 RedRat Hardware

This step sets up which RedRat hardware is to be used by this TestManager installation as shown in figure 2.

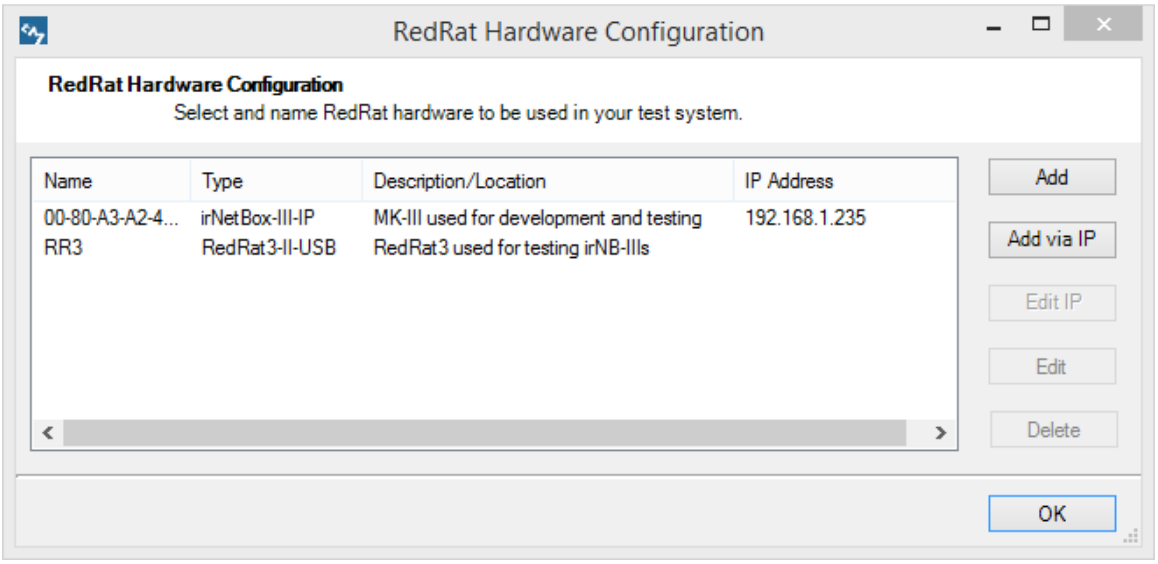


Figure 5. RedRat hardware configuration dialog.

6.1.1 Add

This initiates a search for RedRat hardware, and will discover any RedRat devices attached to this PC or any irNetBoxes on the network. Select only the hardware that is to be used with this TestManager instance. For example, if there are two test farms both using irNetBoxes, then all irNetBoxes may appear in this list, however you should only select the ones in the test farm for which this TestManager installation is being used.

6.1.2 Edit

A dialog box showing hardware specific information is displayed, such as the firmware and hardware versions. A few fields can be edited so that more useful information is displayed:

Name: Some simple name to identify the hardware. This is then used in later configuration steps.

Description: Helps identify the hardware, for example giving its exact location.

Resolve (irNetBox only): When an irNetBox is to be used, it can either be found using a UDP based network search (which takes several seconds) or by communicating directly with the stored IP address. If the irNetBox has a static IP address, then either option can be used, however if it obtains its IP address dynamically (using DHCP) then this field has to be set to *lookup* as the IP address is not guaranteed.

6.1.3 Delete

RedRat hardware no longer in use by this TestManager installation can be removed from the system. This is only allowed if no set-top boxes are currently configured to be controlled by this hardware.

6.2 IrNetBox Output Groups (Optional)

An output group is a set of irNetBox outputs that should operate as a single virtual IR output. This is rarely used but is available for example in situations where you know that you will always want to send the same IR signals to a set of outputs. (Within TestManager, using Zones is a more flexible way of grouping STBs.)

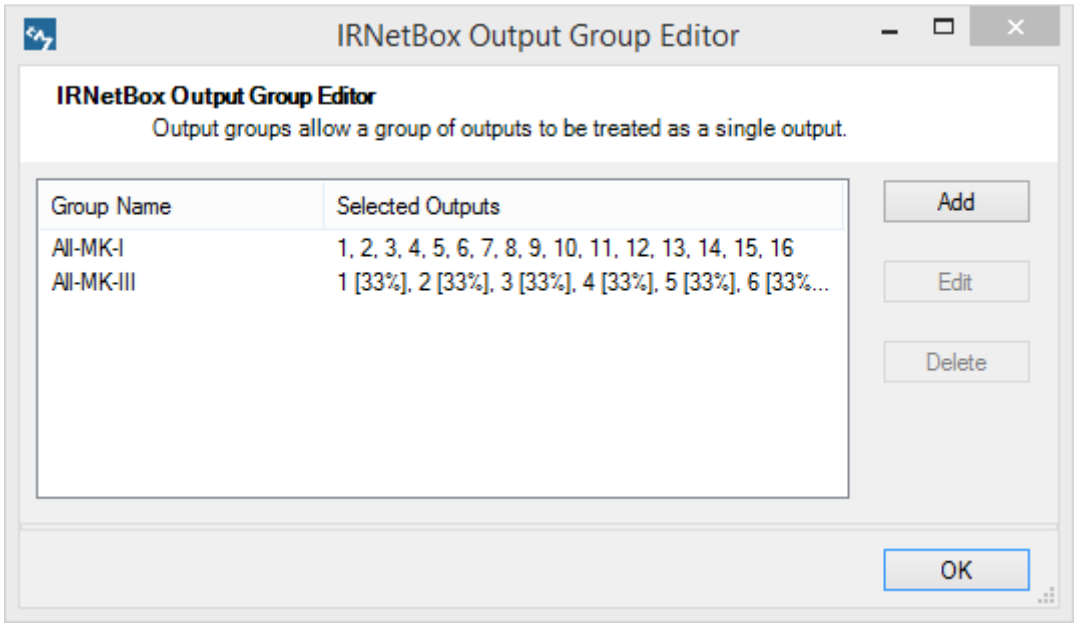


Figure 6. The output group editor.

When selecting outputs, the editor allows selection of different output powers for the different generations of irNetBox:

MK-I: Simple ON or OFF for each output.

MK-II: Power levels of LOW, MEDIUM or HIGH can be chosen by right-clicking on each output.

MK-III and MK_IV: Power levels of 0 (off) to 100 can be chosen, by right-clicking on an output and using the slider.

6.3 STBs/TVs

In this step each STB/TV is assigned a particular IR output and an IR dataset (see section 5), which is usually done while wiring up the irNetBox IR emitters.

6.3.1 Wiring and Testing irNetBox IR Emitters

When using an irNetBox, an IR emitter (also sometimes called "flasher") is stuck to the front of the STB as near as possible to the IR detector. It is very important that the emitter is placed correctly to ensure reliable control of the STB. On some STBs, the exact position of the IR detector can be difficult to find, so the following procedure can help:

1. Try and identify the position of the IR detector on the STB using any visual cues (IR windows etc.). Using an LED or halogen torch can sometimes help view behind any IR transparent windows on the front of the STB.
2. Ensure that you have the IR signal dataset captured for this STB. See section 5 for details on IR dataset management.
3. Start the Signal DB Utility and load the dataset for use with this STB (or all datasets if they are stored in one file).
4. Plug in an IR emitter into a free output on the irNetBox that is going to control this STB.
5. In the Signal DB Utility, select the irNetBox (step 4) from the *RedRats* menu item. Open the irNetBox output selection window from the *View → IRNetBox Output Config* menu item and then enable the output into which the IR emitter was plugged in step 4 (click on the virtual LEDs to turn on/off).
6. To help the correct positioning of the IR emitter on the front of the STB, the Signal DB Utility can repeatedly output IR signals while the IR emitter is moved around to find the optimal location. Select a suitable signal that will cause an action each time it is output (e.g. CHANNELUP), right-click on it and select *Test Signal Output*. In the output dialog select *Repeat Start*.
7. Once the optimal position has been found, stick the emitter in this position.

If it is difficult to find any position that works very reliably, then try using output 1 on the irNetBox with high-power output enabled.

6.3.2 Using the TestManager STB Configuration Dialog

The main editor is shown in Figure 7, in which STBs can be added, edited and deleted. For each STB to be controlled by TestManager, the dialog box shown in Figure 8 is used enter the following information:

Name: Some simple name to identify the STB. This is then used to identify this STB in zones and scripts.

Description: Helps identify the STB, for example giving its exact location.

IR Signal Dataset: This is the name of the IR signal dataset that will be used with this STB. A drop-down list allows the choice of any IR datasets stored in the TestManger database – see section 5 for details of loading IR datasets.

RedRat Device: Selection of one of the RedRat devices configured in section 6.1.

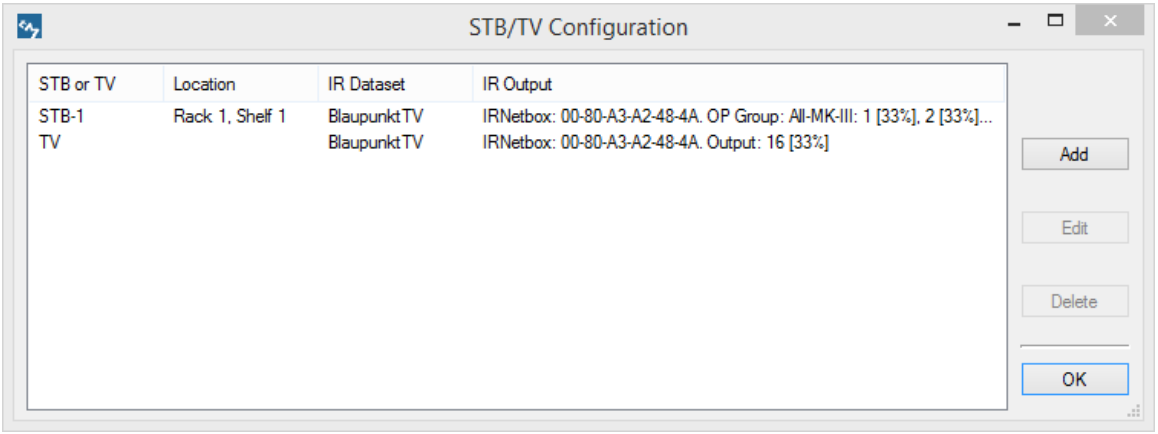


Figure 7. The editor for setting up STBs

Output Port: If the RedRat has multiple output ports, then this lists the outputs available.

Output Group: If an output group has been configured as in section 6.2, then it can be selected here for use with this STB.

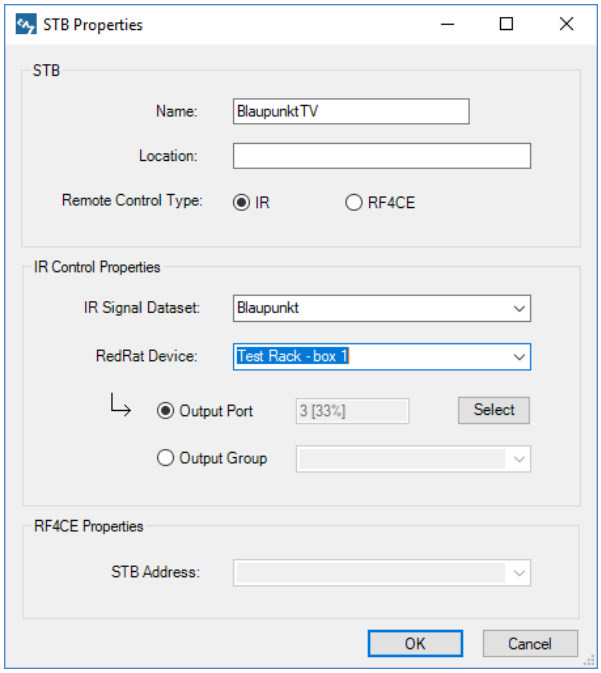


Figure 8. Configuration dialog for one STB.

6.3.3 Graphical Representation of the STB Layout

In some contexts, having an intuitive and rapid method of interactive STB control for all STBs in the system can be useful. The *STB Layout* window is designed for this purpose and clicking on any STB in the layout window will bring up an on-screen remote control to directly control that STB. However, it is not required for the operation of scripts, so this section can be skipped and returned to at a later point in time if required.

Figure 9 shows an example of a *STB Layout* dialog, which has STB icons arranged to reflect their groupings and/or physical positions. Once configured, clicking on any STB icon will bring up an on-screen remote control to that STB, so allowing simple control of any STB within the test system. This section explains how the *STB Layout* window is configured.

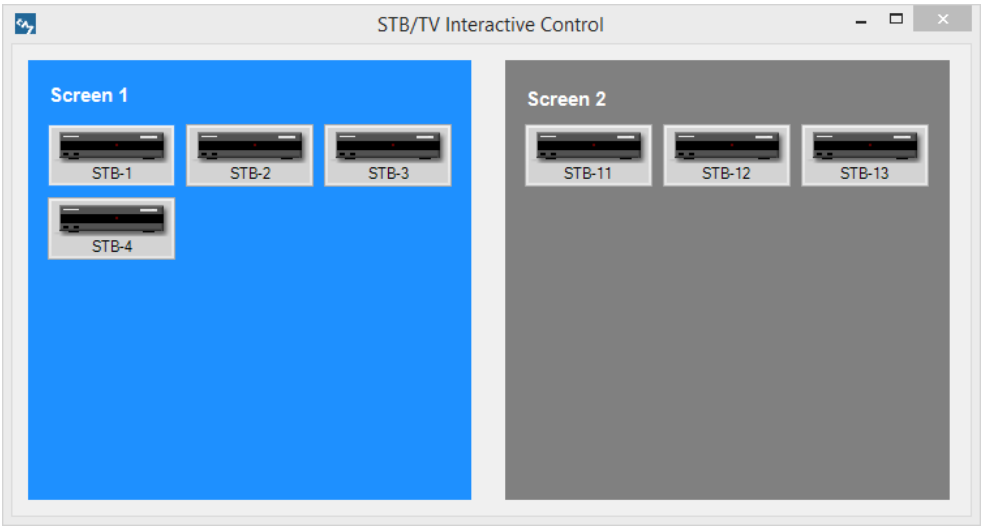


Figure 9. The Graphical Representation of STB Configuration

To setup the STB Layout window, it can be switched into edit/designer mode where STB icons can be added and configured. To start the designer, do one of the following:

- Open the STB Layout window from the *View → STB Layout* menu item. Right-click in the window and select *Edit STB Configuration* from the popup menu. Or…
- Use the *Edit → Configure → STB Layout* menu item.

6.3.3.1 Adding Panels and Labels

Panels and labels can be added by right-clicking on the editing panel.

6.3.3.2 Adding STB Icons to the Diagram

The STB Layout window works with STBs that have already been defined, as described in section 6.3.2. To add an STB to the diagram, right-click and select *Add* from the popup menu, and an STB icon will be added with the name of the first STB in list of STBs known about by *TestManager*. The actual STB associated with this icon and other properties can be configured by selecting the icon and modifying the values in the grid on the right-hand side of the designer.

6.3.3.3 STB Icon Properties

When an STB is selected, some of its properties can be changed directly in the designer, and some in the grid on right hand side of the designer.

- **Size & Position:** Select and drag the icon to the correct position. Also, the corners and sides can be moved to adjust its size.
- **STB/TV Name:** The actual STB that this icon represents. The drop-down list is populated from those configured in STB/TV Configuration window as given in section 6.3.2. There can only be one icon per STB, so once an icon has been assigned to a certain STB, that STB name is removed from this drop-down list.
- **Remote Control:** Select an on-screen remote control to use to control this STB. When this STB icon is clicked by a user, this on-screen remote control will be shown.
- **Background Colour:** A background colour for icon can be selected to help differentiate it or visually reinforce groupings.
- **STB Image:** A drop-down list of available icon images allows you to select a particular image for this STB. It is also possible to select your own image for an STB by clicking on the "+" symbol in the image selection dialog.

When editing is complete, changes are stored and if the STB Layout window is open, it will be updated to reflect the changes made.

7. Backing Up the TestManager Database

The TestManager database can contain a lot of data, scripts and other information that would take a lot of time to re-create, so it is good practice to take backups of the database to store elsewhere; on a different computer at least, but ideally off-site.

There may already be backup mechanisms in place for databases and file systems, but if not then it is possible to make database backups from TestManager. The backup process depends on whether the application is setup to run in single-user or multi-user mode.

7.1 Multi-User Database Backup

This makes a copy of the SQL Server data from the database by instructing the SQL Server itself to execute a backup operation. Figure 10 shows the dialog for creating the backup, which is started from the *File -> Backup TestManager DB* menu item. This will cause SQL Server to create the named backup file in the given directory on the database server.

Please Note: This all takes place on the database server and not on the local machine (unless they are the same).

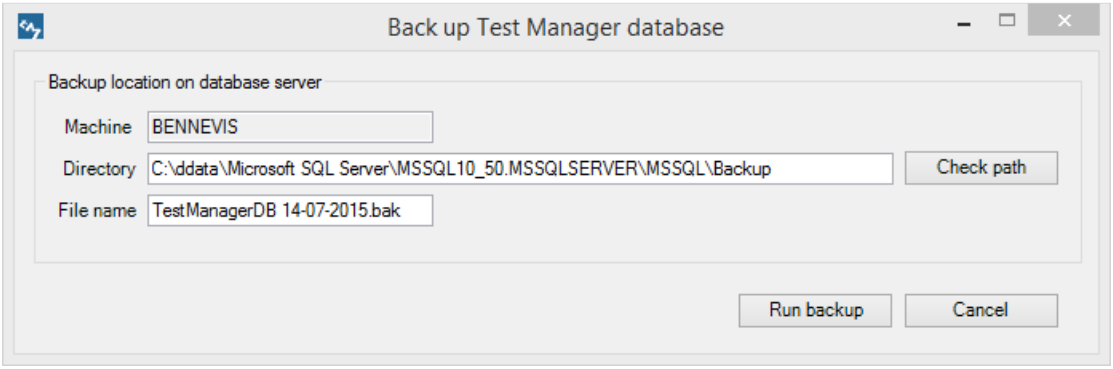


Figure 10. The database backup dialog

As this takes place on the server, to help in the selection of the directory the **Check Path** button can be used to help ensure that the directory actually exists.

7.2 Single-User Data Backup

As data is stored in a file on a local machine, the backup in this mode takes a copy of this file and prompts you for a location to save the file. It is recommended that the backup files are copied off the machine in case of hard-disk failure.

8. 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.

8.1 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 the system's 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 **Error! Reference source not found.**.

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 and 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 selected set-top boxes. 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 – see section 5.4.

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:

Figure 11. On-screen Remote Control

8.1.1 Select Zone

If zones have been configured (see section 9) then one can be chosen and IR signals will be sent to STBs in that zone only.

8.1.2 Select STBs

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

8.2 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.

For information about configuring the panel, please see section 6.3.3. Once configured, it will show your STBs graphically, similar to that shown in Figure 9.

Clicking on any STB will bring up the on-screen remote control associated with that STB and TestManager will the 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:

8.2.1 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. Using the keyboard CTRL key while selecting STBs allows multiple STB selection in a similar manner to Multi-STB Mode described next.

8.2.2 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.

8.2.3 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).

8.2.4 Selecting a Zone

All zones that have been defined in the system (see section 9) 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).

8.2.5 Executing Scripts from the STB Layout Window

If one or more scripts have been made available in the STB Layout window, then they can be executed here to run against the selected STBs. To execute a script, right-click on the STB Layout window and select the required script from the *Execute Script* context menu item. This requires that the script has been setup to use the list of externally selected STBs, details of which are given in section 14.4.

9. 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.

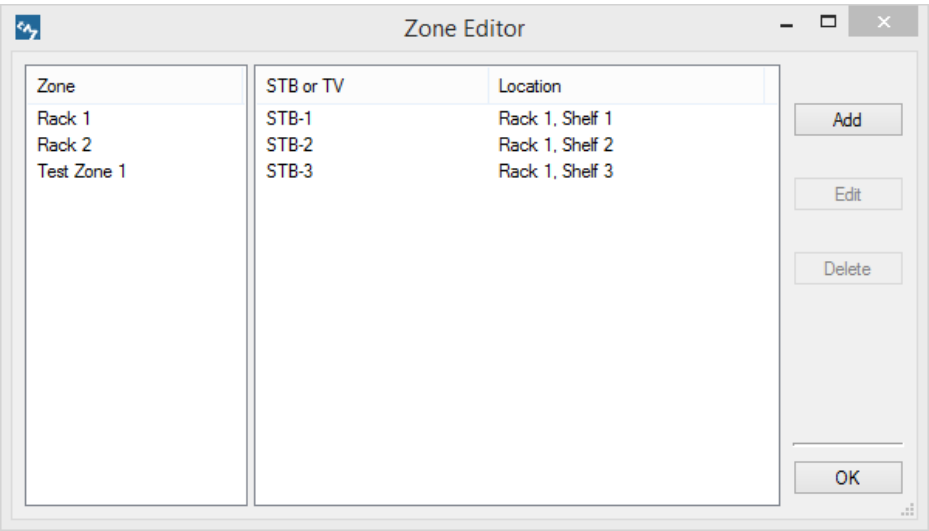


Figure 12. The zone editor.

The zone editor, shown in Figure 12 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.

10. 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 13.

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.

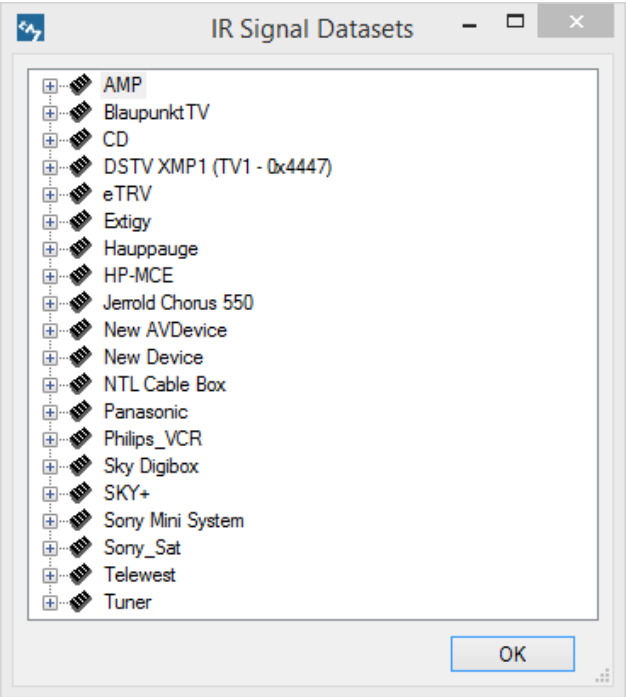


Figure 13. Signal dataset viewer

Selecting a signal will bring up a graphical viewer for the IR signal.

Figure 14. Graphical IR signal viewer.

11. Macros

A macro is a sequence of IR signals that cause a 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.

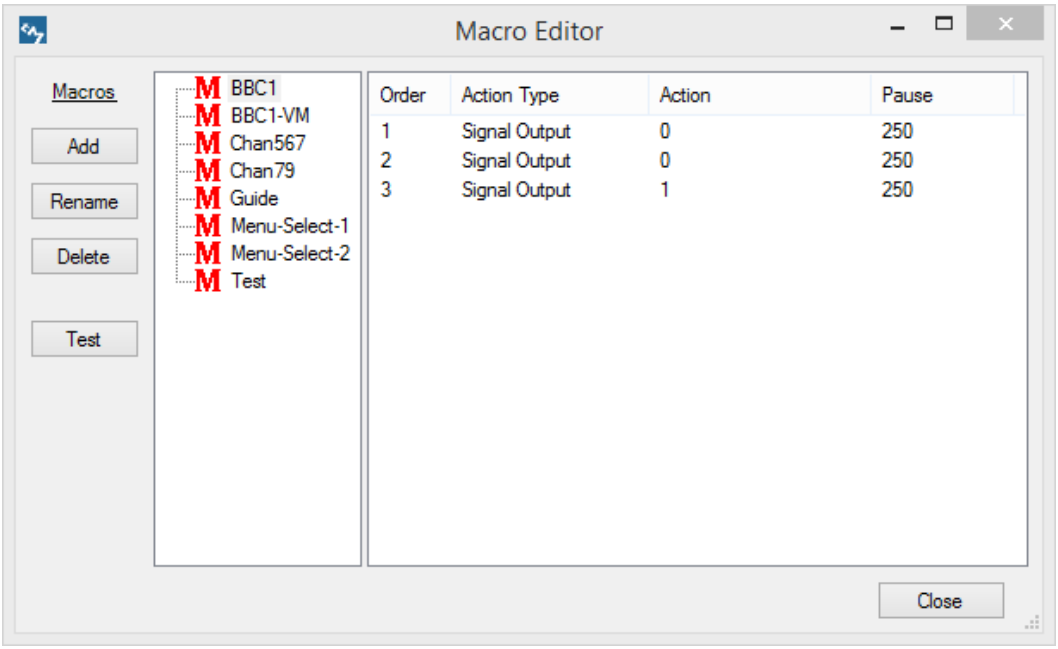


Figure 15. The macro editor.

Figure 15 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 panal, right-click and select Add Action, which will bring up the action dialog box shown in Figure 16.
4. Select the IR signal name from the *IRSignal* drop down menu. The list of signals shown is pre-configured, but be changed as described in section 5.3 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.

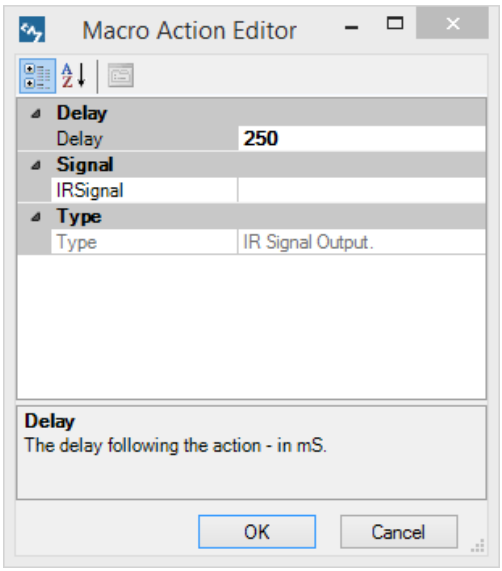


Figure 16. Dialog box for adding a signal to a macro

12. Named Operations

The principle behind named operations is that there are operations with clearly understandable name (e.g. change channel to **BBC2**) but which may 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. Figure 17 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 a 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.

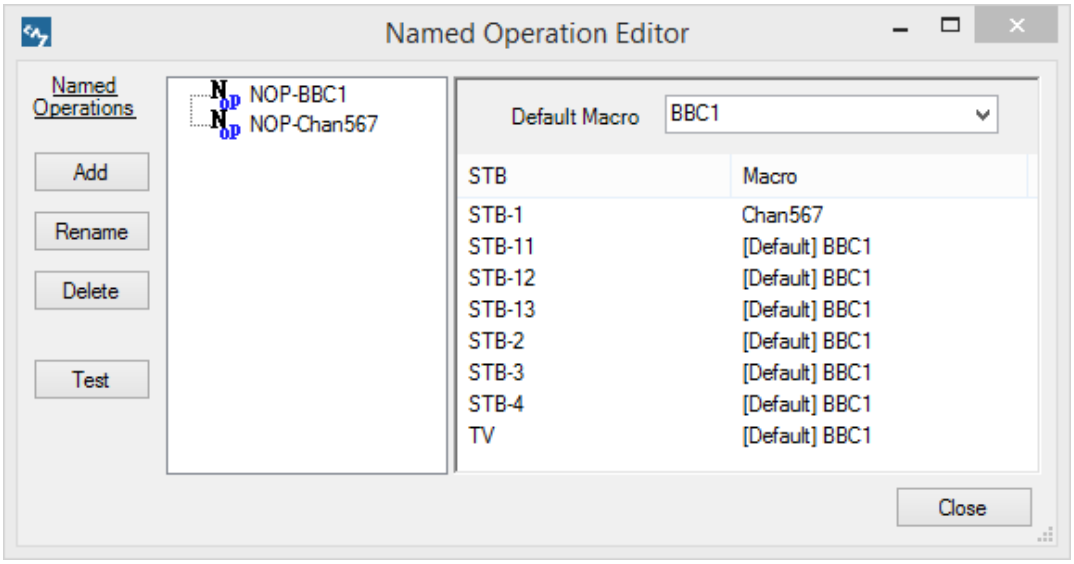


Figure 17. The named operation editor.

13. Control of STBs with a Remote Control Handset

This functionality supports the control of STBs via a physical 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 or RedRat-X 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/X and remote control must be shielded or being used in a different physical location.

Configuration is done in the options dialog (*Edit → Options → Control Via Remote* tab). Here the RedRat3/X for IR input is selected, and the IR signal dataset that corresponds to the physical handset to be used. If the RedRat3/X 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).

It is possible to select multiple IR datasets to use to decode input from remote control handsets if more than one handset is to be used. Using many IR datasets for decoding can potentially reduce the accuracy of decoding, so it is recommended that only one or two be used.

To start control via your remote control handset, open the control dialog (as shown in Figure 18) from the *View → Control STBs via Remote…* menu item.

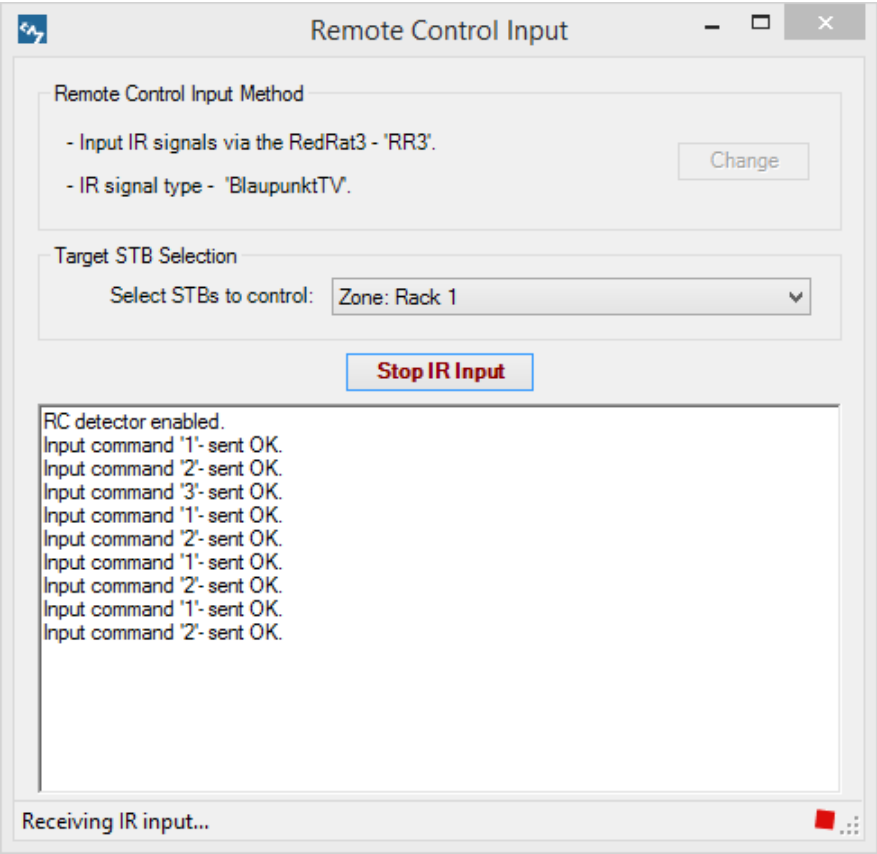


Figure 18. Window used to control STBs with a remote control handset.

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 8.2), and either single or multiple STBs can be selected here. Clicking on a STB will not bring up an on-screen remote control when the STB Layout window is accessed in this way.

Zones – All zones are listed here, so can be selected. See section 9 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. This information is also stored in the log file ***IR-Input.log*** if file logging is enabled.

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).

14. Scripts

The features listed in the previous sections are all building blocks to support the simple creation of scripts. In this section, TestManager scripting will be introduced.

Two script languages are supported; TM Script and Python, the details of which are described in later sections. Firstly, general script management within TestManager will be introduced.

14.1 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 19.

Figure 19. Toolbar button functions.

14.1.1 Script Editing

Scripts can be edited either using the built-in editor or an external editor. A script can be opened in an editor in one of the following ways:

1. Double-clicking on it in the right-hand panel of the *TestManager* window.
2. Selecting a script and using the *Edit Script* toolbar icon.
3. Using the *Script → Edit Script* or *→ Edit Script (External Editor…)* menu items.

For actions 1 and 2, the use of the internal editor or an external editor can be configured in *TestManager* options.

14.1.1.1 The Built-In Editor

The build-in editor (shown in Figure 20) provides a simple and quick method of editing scripts. In the left-hand pane, the script code is presented with syntax highlighting, and the right-hand pane shows the script execution and output logs.

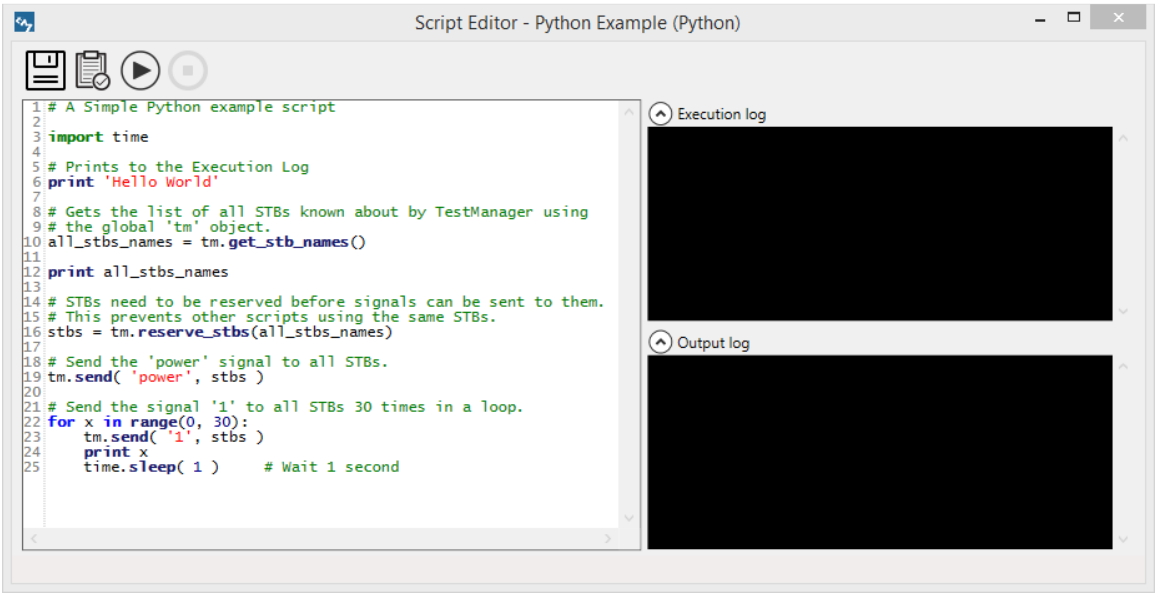


Figure 20. The Built-In Script Editor

14.1.1.2 Using an External Editor

The mechanism TestManager uses with an external editor is to write the script to a temporary file on disk, and then open 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 *Scripting* tab), and different editors can be selected for Python and TM Script. Additional command line arguments can be given here if the text editing program requires that, for example to get it to open each file in a new window. The actual filename is automatically appended to the set of command line arguments by *TestManager*.

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.

14.1.2 TM 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.

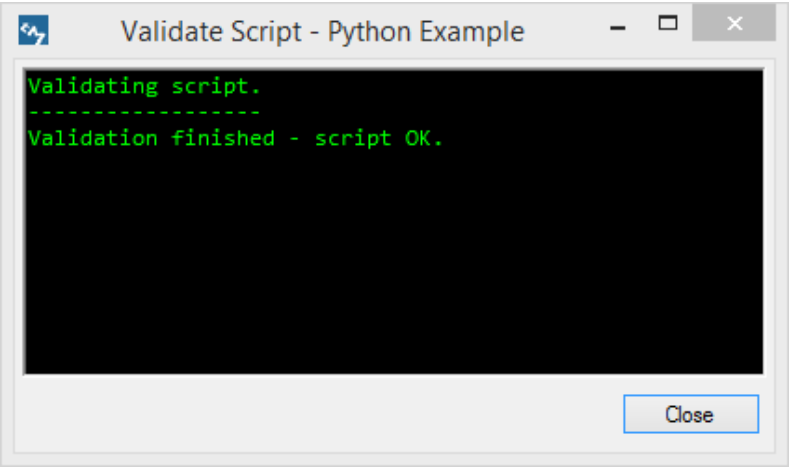


Figure 21. Output from script validation, showing errors.

Figure 21 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.

14.1.3 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 22 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.

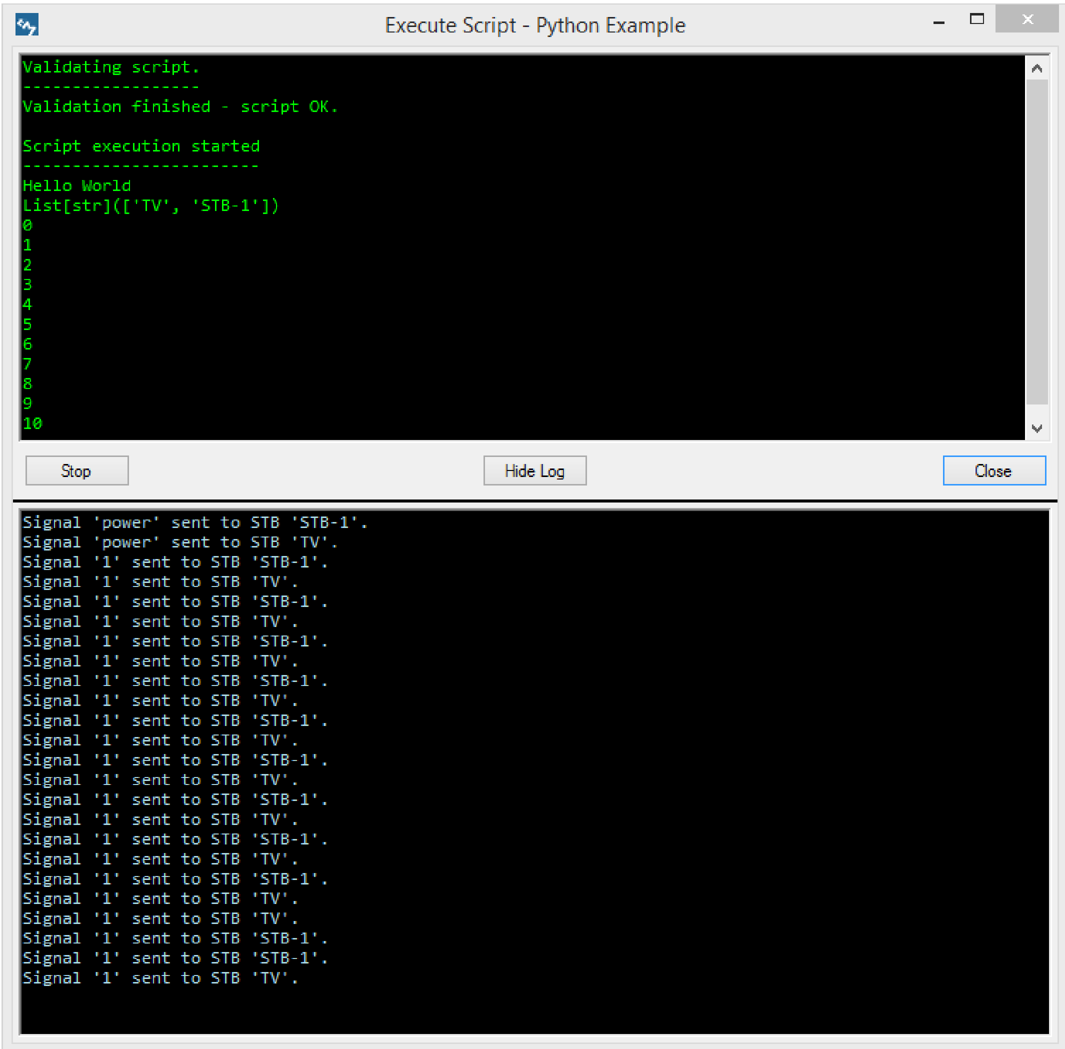


Figure 22. Script execution output.

14.1.4 TM Script Capture

Section 14.1.1 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 or RedRat-X device for IR signal input. Once a TM 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.

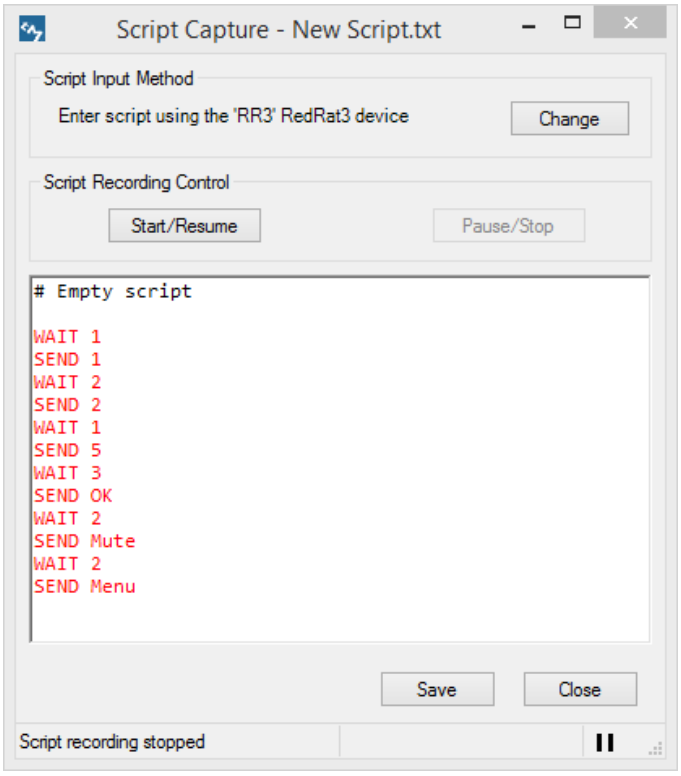


Figure 23. The Script Capture Dialog

14.1.4.1 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/X device can be selected as the input method, and if a RedRat3/X if 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 or RedRat-X: 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.

14.1.4.2 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.

14.1.4.3 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.

14.1.4.4 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.

14.2 The *TM Script* Language

This is a simple scripting language which supports STB selection, the output or control instructions, time delays and loops. It is adequate for many scripting tasks, but if a complex script operations are needed then Python can be used.

14.2.1 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-List | Zone | 'ALL']
```

Examples:

```
select humax-pvr
select STB-1, STB-2, STB-3
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.

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

14.2.3 Wait

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

```
wait N
```

Examples:

```
wait 1
wait 360
```

14.2.4 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
```

14.2.5 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.

14.2.6 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.

14.2.7 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
```

14.3 Python Scripting

TestManager supports scripting using IronPython, which can interact with TestManager's .NET APIs. Much of the Python standard library is not included with *TestManager* and must be installed separately. The download is available from the IronPython webite - http://ironpython.net/.

Note that TestManager expects a certain version of IronPython to be available, basically the version with which the application was built:

TestManager Versions	IronPython Version
V4.19 to V4.68	2.7.5
V4.69 to V4.82	2.7.7
V4.80 to V4.83	2.7.8
V4.84 or above	2.7.9

Once IronPython is installed on your machine you must update the options within TestManager so that the IronPython library directory can be located, as illustrated in Figure 24.

IronPython scripts can be run from either inside *TestManager*, or from outside (*TestManager* V4.61 onwards).

14.3.1 TestManager Python environment

The Python environment within *TestManager* is much the same as that provided by IronPython, but with two important differences. Firstly, *TestManager* defines a global object ***tm*** to allow access to the *TestManager* API. Secondly, the import mechanism has been extended to allow scripts defined within *TestManager* to be imported as normal Python modules.

14.3.2 Importing scripts from TestManager

All *TestManager* scripts are encapsulated within the pseudo-package *scripts* in the same hierarchy as displayed by *TestManager*. Consider the example where we have a script called utils within the Python folder containing the following code:

```
def foo():
      print 'Hello, World!'
```

We can call *foo* so:

```
import scripts.Python.utils as utils
utils.foo()
```

Notice that the names of the folders and scripts involved must be valid Python module names for the import to succeed, but TestManager does not enforce this. Note also that folder and script names are case sensitive.

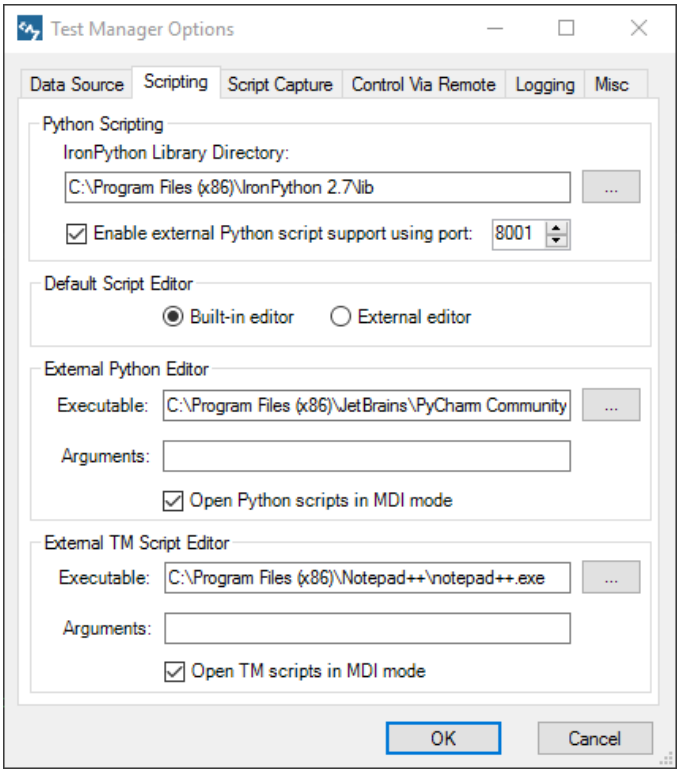


Figure 24. Python scripting configuration

14.3.3 Using IronPython Outside TestManager

Python scripts can also be executed outside *TestManager* (from V4.61 onwards) in an almost identical fashion to scripts running within *TestManager*. This allows *TestManager* itself to be used as the STB and IR signal dataset management system, while gaining the benefits of working with Python scripts placed on the file system:

- Script source code management systems can be used (Git, SVN etc.)
- Python code can be written, debugged and run from within a development environment, such as PyCharm.

Python scripts running outside *TestManager* use .NET's Windows Communication Foundation (WCF) to communicate with *TestManager*, so IronPython has to be used.

14.3.3.1 Enabling External Script Execution Support

This is done in the Scripting tab of the Options dialog, enabling support and setting the port to use. The default is port 8001.

Once this is done, the computer's firewall may prompt you to allow communication on this port. It is recommended that communication is only allowed on domain or private networks.

14.3.3.2 Setting up a Python Script to use TestManager

A Python script executed within *TestManager* is almost identical to one operating outside *TestManager*, however the external execution environment has to be setup to include the necessary files. These are:

- ***TestManager.Core.dll*** – The core .NET code for the Python client to communicate with TestManager.
- ***tm.py*** – The Python module which provides the *tm* API, i.e. wraps the communication code.
- ***tmtasks.py*** – A Python class to support concurrent control of multiple STBs. See section 14.3.6.

The simplest method to use these files is to place them in the same directory as your Python script.

The Python client code *tm.py* needs to know how to connect to *TestManager*. It is setup to use the default values of machine ***localhost*** and port ***8001***. If your setup is different, then please edit these values near the top of the ***tm.py*** module.

14.3.3.3 Running an External Python Script

When an external Python script is started or stopped, this is listed in the main Log Output window - Figure 25.

TestManager also opens a script execution log window, in a similar fashion to internal script execution. This window is closed as soon as the external script is terminated. Multiple external scripts will cause multiple monitor windows to open.

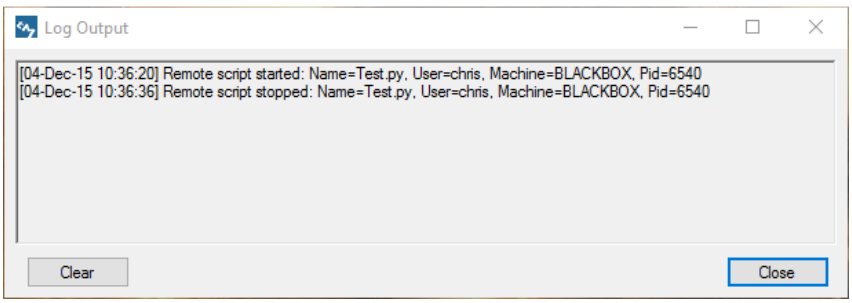


Figure 25. Log output from Python script execution.

14.3.3.4 A Simple External Python Script

```
import tm

stbs = tm.reserve_stbs(['STB-1', 'STB-2', 'STB-3'])
tm.send('play', stbs)
tm.send('pause', stb[0])
```

The folder *ExamplePythonClient* in the *TestManager* installation directory shows an example Python script which can be executed with IronPython.

14.3.4 The TestManager Python API

All functions defined by *TestManager* are accessible through the *tm* object. The following sections document these functions and give hints on when to use them.

14.3.4.1 Data retrieval functions

Several functions are provided to retrieve information about the objects defined in *TestManager*. These functions are not required to write simple scripts, where the name of the STB, signal etc. are already known, but can be useful when writing more complicated scripts that operate with subsets of these. For example, you might use the **get_stb_names()** function when writing a script that uses some or all of the STBs defined in *TestManager*.

get_stb_names()

Returns the names of the STBs defined in TestManager as a list of strings.

get_zone_names()

Returns the names of the zones defined in TestManager as a list of strings.

get_signal_names()

Returns the names of the signals defined in TestManager as a list of strings.

get_macro_names()

Returns the names of the macros defined in TestManager as a list of strings.

get_named_operation_names()

Returns the names of the named operations defined in TestManager as a list of strings.

get_prefdefined_stb_names()

Returns the set of STB names which are defined externally from the script, for example in the STB Layout window. This means that a script can be interactively run against STBs selected on screen.

14.3.4.2 Signal output functions

Signals can be sent either to all STBs or to a subset of them. In the latter case, this subset must first be reserved, either by calling **reserve_stbs()** or **reserve_zones()**. Notice that both of these functions return a list of STB objects. There is no function to release STBs or zones – TestManager does this for you.

reserve_stbs(stb_names)

Reserves one or more STBs specified by *stb_names*, which is a list of strings, and returns a list of STB objects.

reserve_zone(zone_name)

Reserves the zone specified by *zone_name*, which is a string, and returns the list of STB objects in that zone.

send(name)

Transmits the named operation, macro or signal specified by *name*, which is a string, to all STBs.

send(name, duration)

Transmits the signal specified by *name*, which is a string, to all STBs for the given duration. The duration is a floating point number, specified in seconds, and should be larger than 0.1s.

The duration of an IR signal is extended by TestManager through increasing the number of signal repeats, so simulating a longer remote control button press. If the IR signal does not have repeat data, or the duration given causes the number of repeats to exceed 255, then an error will be generated. This only applies to signals, and not to named operations or macros.

send(name, stb)

Transmits the named operation, macro or signal specified by *name*, which is a string, to one *stb* object.

send(name, stb, duration)

Transmits the signal specified by *name*, which is a string, to one *stb* object for the given duration. See section 0 for more details on the duration parameter.

send(name, stbs)

Transmits the named operation, macro or signal specified by *name*, which is a string, to the list of STB objects *stbs*.

send(name, stbs, duration)

Transmits the signal specified by *name*, which is a string, to the list of STB objects *stbs* for the given duration. See section 0 for more details on the duration parameter.

14.3.4.3 STB object

The STB object has the following properties, which must not be changed.

id

The ID of this STB, used internally by TestManager.

name

The name of the STB.

14.3.5 Example scripts

14.3.5.1 Send a signal to a known STB

```
stbs = tm.reserve_stbs(['My STB'])
tm.send('play', stbs)
```

14.3.5.2 Send a signal to all STBs except one

```
all_stbs_names = tm.get_stb_names()
my_stbs = [stb for stb in all_stbs_names if stb != 'STB-8']
stbs = tm.reserve_stbs(my_stbs)
tm.send('play', stbs)
```

14.3.5.3 Send a signal to a zone

```
rack1 = tm.reserve_zone('Rack 1')
tm.send('play', rack1)
```

14.3.5.4 Send every signal to a zone

```
rack1 = tm.reserve_zone('Rack 1')
for signal in tm.get_signal_names():
      tm.send(signal, rack1)
```

14.3.6 Parallel Execution of STB Control Operations

Generally, multiple STBs are controlled by sending the same sequence of control signals to a list of them. However, it can sometimes be necessary to send different sets of commands to different STBs, in effect doing multiple, different operations at the same time.

The .NET framework has a lot of support for these kind of concurrent operations, one form of this support being the *Task Parallel Library*. This can get quite complex to use, so parts of it have been wrapped my the *tmtasks.py* module to give simple but effective use in Python.

To use it in code, take the following steps:

1. Import the *tmtasks* module:

   ```
   import tmtasks
   ```

2. Create a *tmtasks* object:

   ```
   tmtasks = tmtasks.TmTasks()
   ```

3. Create a function which you want to call multiple times, for example once for each STB in a list.

4. Call this function multiple times:

   ```
   for i in range(len(stbs)):
         tmtasks.startTask( lambda x=stbs[i]: sendSeq(x))
   ```

These functions will then be started up in the background. The slightly strange syntax: 'x=param' is needed to capture the actual parameter instance. See lambda function closures.

5. Wait for completion of all the tasks:

```
tmtasks.waitAll()
```

When printing out information from functions being run in parallel, the text from different functions can become mixed up, i.e. is not *thread safe*. To void this, use the **tmtasks** print mechanism which ensures that each line is printed sequentially:

```
tmtasks.printMessage("Starting output to " + stb.name)
```

14.4 Script Execution from the STB Layout Window

This feature allows the user to select the set of STBs from the STB Layout window, and then run a script against this set of STBs. This can be useful in a number of situations, for example:

- With short scripts when used with interactive control of STBs, or
- The target STB list of a script needs to be selected more dynamically than a set list at the top of the script.

For scripts to be called from the STB Layout window, they need to be setup correctly:

14.4.1 TM Scripts

There should be no “SELECT” statement at the top of the script as the selected STBs are inserted directly into the script execution by *TestManager* from the STB Layout window.

14.4.2 Python Scripts

The selected STBs can be obtained in the Python script using the **tm** object as shown in the code snippet below. The call **get_predefined_stb_names()** returns the list of STBs selected in the STB Layout window.

```
# Test Python script for use in STB Layout

stbNames = tm.get_predefined_stb_names()

stbs = tm.reserve_stbs(stbNames)

for stb in stbs:
    print stb.name

tm.send('1', stbs)
tm.send('2', stbs)
tm.send('3', stbs)
tm.send('4', stbs)
tm.send('5', stbs)
```

14.4.3 Making Scripts Available to the STB Layout Window

Once the scripts have been created, then to make them available for execution in the STB Layout window, right-click on it and select ***Script available in STB Layout***.

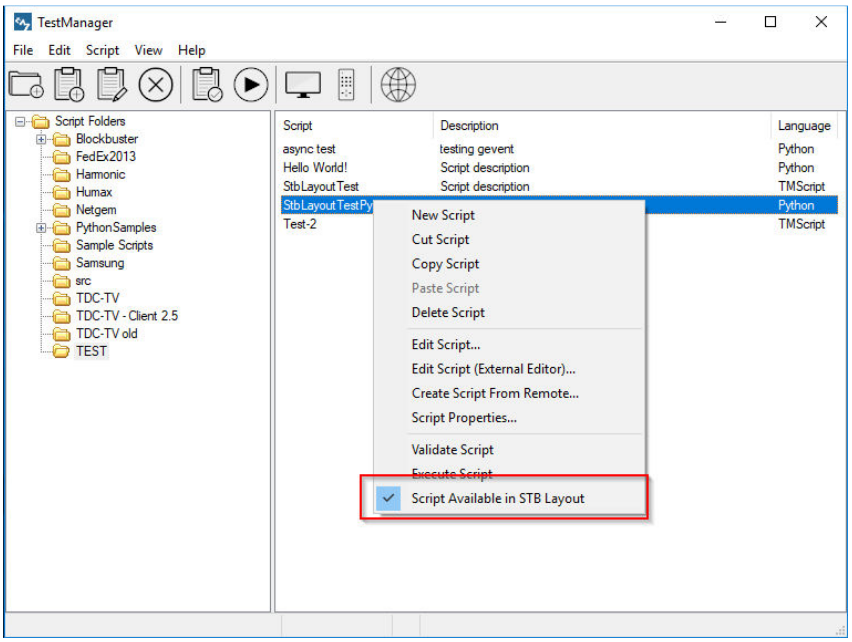
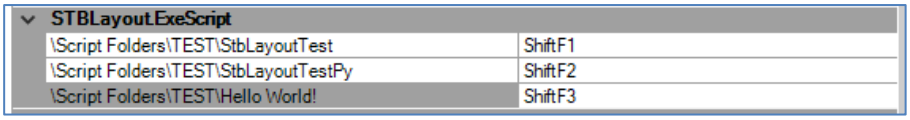
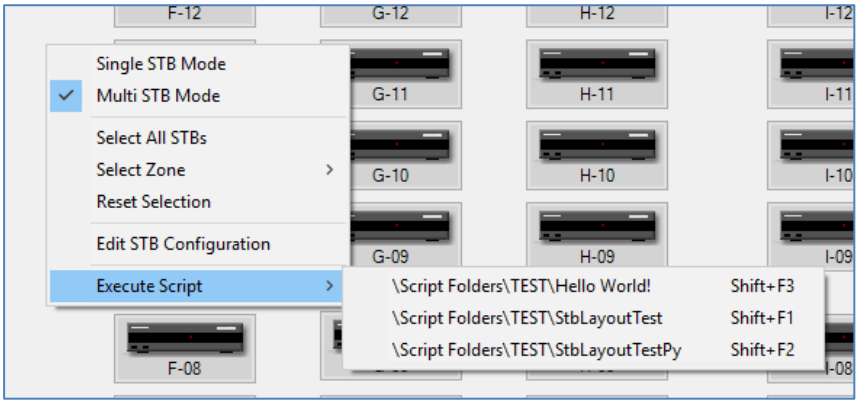


Figure 26. Making a script available to the STB Layout window.

For each of these scripts, a keyboard shortcut can be defined in the Options Dialog:



In the STB Layout window, the script can be started using the context menu or keyboard shortcut:



15. RF4CE STB Control

TestManager supports RF4CE control of certain STBs, where work has already been done by RedRat to provide STB control via our RF4CE.

When using IR control in TestManager, each STB is configured to be controlled by a specific IR output on one RedRat device. While the principle is similar when using RF4CE, the setup is a little more complex as a result of the need to keep TestManager's configuration data in sync with the actual physical pair state. Each time a RedRat device search is done and RF4CE modules discovered, TestManager will attempt to read all pairing information from the modules and update its own internal state. Various dialogs allow viewing and editing of this information.

The representation of these concepts in TestManager is shown in Figure 27.

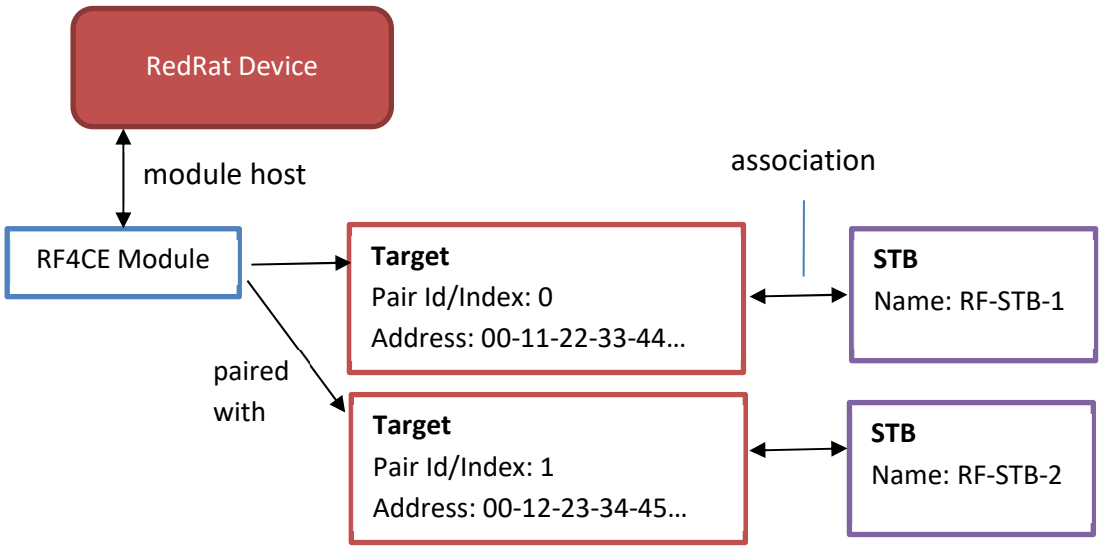


Figure 27. TestManager's representation of RF4CE pair state.

When a physical STB is paired with an RF4CE module, and is given a *Pair Id* or *Index*, which TestManager stores as a **Target**, containing the target's unique address.

TestManager has the concept of an **STB**, which can be controller vie IR or RF4CE. All control operations within TestManager operate on **STBs**, so part of the configuration process if to associate an **STB** object with the correct **Target** information so that commands are routed via the correct RF4CE module to the STB.

15.1 RF4CE Module Configuration

Figure 28 shows the main RF4CE configuration dialog box.

1. **Module selection:** All RF4CE modules discovered by TestManager are listed in the drop-down list.

2. **Module configuration:** Used to set the correct RF4CE configuration for the module. Once it has been given the right configuration, this will not generally be changed.
3. **Paired Target List:** When a module is selected, the list of paired devices is read from the module, sync'd with TestManager's internal state and displayed. If an STB object has been associated with the target address, its name and location is also shown. (See section ??? for associating STBs with target addresses.)
4. **Discovery and Pairing:** This section of the dialog is used to pair physical STBs with RF4CE modules. The normal set of steps are:
    a. Use the *Discover* button to find RF4CE targets (physical STBs) with which to pair.
    b. Select one of the addresses shown in the list and press *Pair*.
    c. If the STB requires a PIN code needs to be entered to confirm pairing, use the *Show Remote* button to bring up an on-screen remote control for PIN number entry.

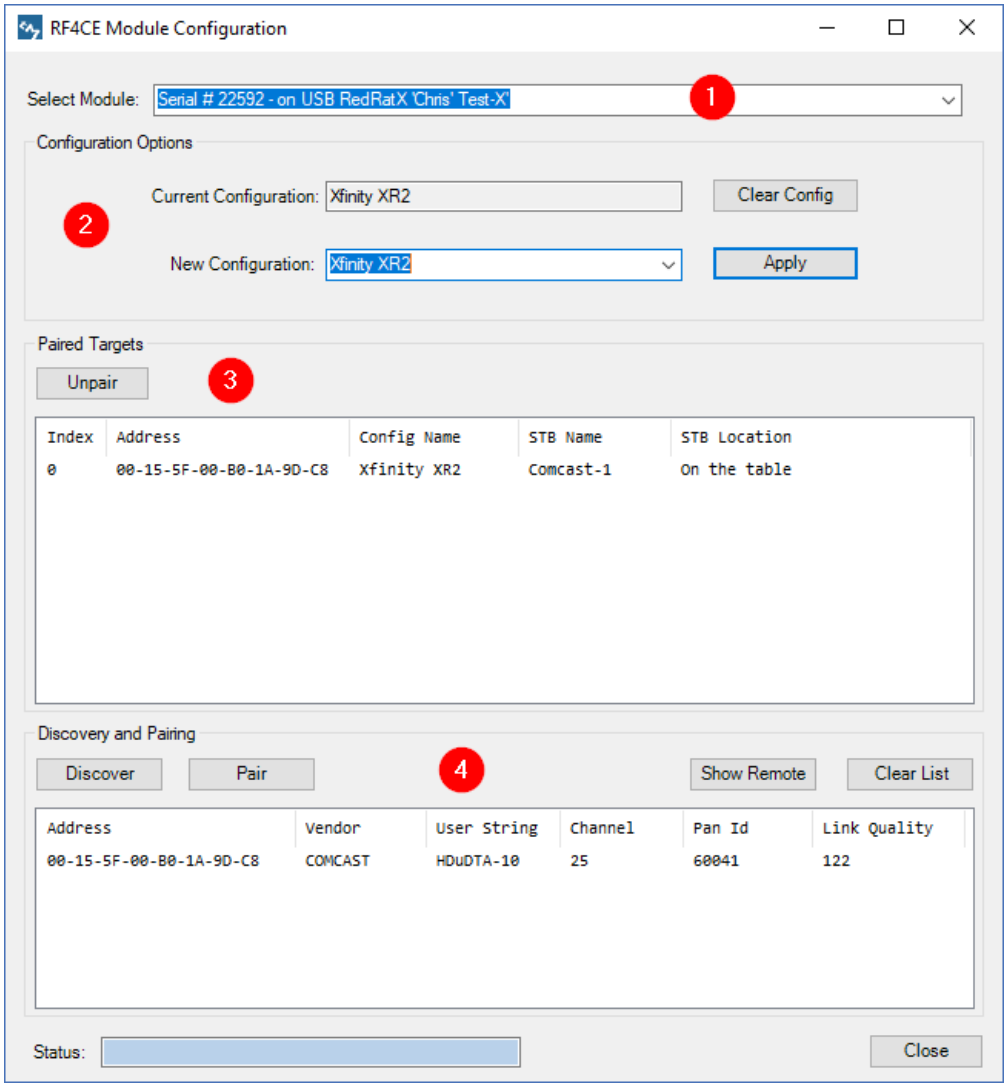


Figure 28. RF4CE module configuration.

15.2 STB and Target Association

Once an RF4CE target (physical STB or TV) has been paired with an RF4CE module, it needs to be associated with an STB object within TestManager. This is done in the *STB/TV Configuration* dialog, using *the Edit -> Configure -> STB/TV Properties* menu item.

In this window, new STB rows can be added, or pre-existing STB rows edited, and in both cases the dialog box shown in Figure 29 is used.

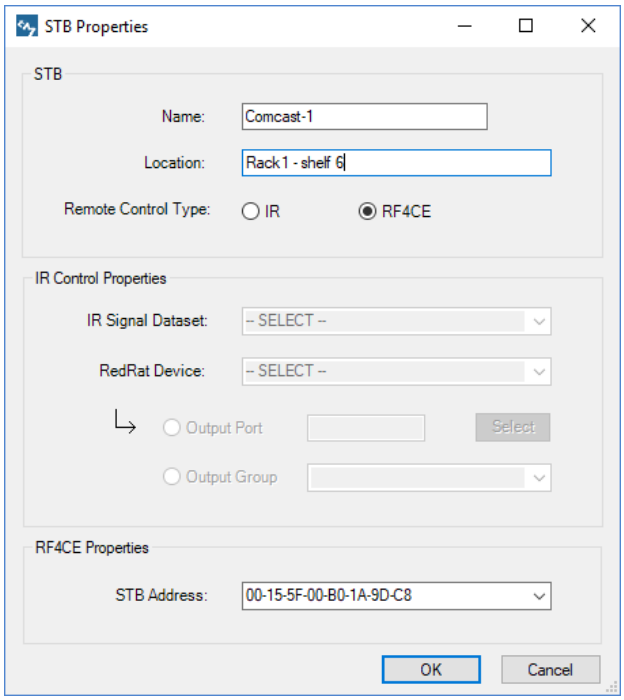


Figure 29. Using the STB Properties dialog to associate a target address with an STB.

Select RF4CE control and then choose the target address of the physical STB to be associated with this name and location.

15.3 The RF4CE Module Dialog

This dialog is used to view the list of RF4CE modules and the targets with which they are paired. It is shown from the *View -> RF4CE Targets* menu item.

This dialog allows deletion of RF4CE modules from TestManager, for example if it is no longer in use. However, a new device search or bringing up the *RF4CE Module Configuration* dialog will re-instate the information about this module if it is still discoverable.

15.4 The RF4CE Target List

Opened from the *View -> RF4CE Targets* menu item, this displays full list of all known RF4CE targets.

Target details can be deleted here if there is target details which are no longer needed. This does not unpair the target if it is paired, but just deletes the information from TestManager's database.

16. Log Output

16.1 Application Log Output

General log output can be viewed from the *View → Log Output* menu item which will bring up a log window. This shows information such as when a script was started, or any errors encountered during operation.

If the ***Log script execution and other output to file option*** is checked in the Options dialog then application log output is sent to a file called *TestManagerAppLog.Log* in the log directory.

16.2 Script 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.

17. Initiating Script Execution from Other Applications

If infrared remote control based testing 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.

17.1 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:

17.1.1 -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.)

17.1.2 –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*.

17.1.3 –help

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

17.2 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.
```

17.3 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, so 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.*

17.4 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.

18. Using TestManager with RedRat Hub

The standard method of operation of RedRat software and hardware is that an application will take full control of any RedRat devices it uses. This is primarily to ensure that operations using a set of STBs controlled by a RedRat device are not interrupted by other applications and users.

It can sometimes be useful to share the use of a RedRat device between applications and users, and so the RedRat Hub service is designed to support this, being able to take connections from multiple applications and route commands to the appropriate RedRat device.

From *TestManager* version 4.62 onwards, TestManager is able to use a RedRat Hub instance for output of IR signals, rather than connecting directly with any RedRat device it uses.

18.1 Setting up RedRat Hub

RedRat Hub is a console application which should be started and left to run. Further details on the various command line options and switches are given in the RedRat Hub documentation.

The command line argument *--conns=X* is used to inform RedRat Hub how many concurrent connections from client applications it should support. Connections are a relatively expensive resource, and so there is a balance to be struck; if not enough connections are opened this may reduce the speed at which TestManager can control STBs as operations have to be queued sequentially rather than being performed in parallel.

A rule of thumb is for each *TestManager* instance to open one connection for every two STBs it is to control. RedRat Hub should then be set to make available sufficient connections for all *TestManager* instances.

For example, if four *TestManager* instances are using the same RedRat Hub, each controlling 10 STBs, then each TestManager instance should open 5 connections to RedRat Hub. RedRat Hub can be started in the following way:

```
RedRatHubCmd.exe DeviceDB.xml --conns=20
```

In fact, it would be sensible to make a few more connections available in case additional *TestManager* instances were started.

18.2 Configuring TestManager to Use RedRat Hub

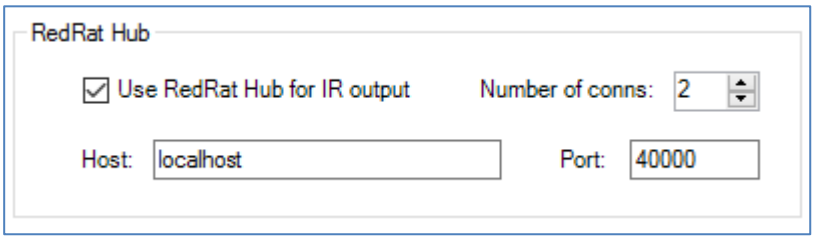
This is done in the options dialog, under the *Misc* tab. The options are:

Use RedRat Hub for IR output: The main switch to enable RedRat Hub use. If this is not checked, then TestManager will connect directly to any irNetBoxes it needs to use.

Number of conns: The number of connections to be opened to RedRat Hub.

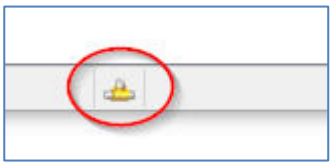
Host: The computer on which RedRat Hub is running.

Port: The port RedRat Hub is listening on. Default is 40000.



18.3 RedRat Hub Connection Status

This is shown at the bottom on the main TestManager window:



If the icon is shown then TestManager is connected to RedRat Hub.

If RedRat Hub is closed or otherwise not reachable, then TestManager will attempt to reconnect to it. This is indicated by the icon flashing.

19. The TestManager Web Interface

TestManager contains a build-in web server which can provide remote access to the STB Layout interactive STB control features.

Basically, it presents the STB Layout display, but in a web page, including an on-screen remote control. The STBs can be selected (individually or as a multi-select), and then the commands from the on-screen remote control are sent to those STBs.

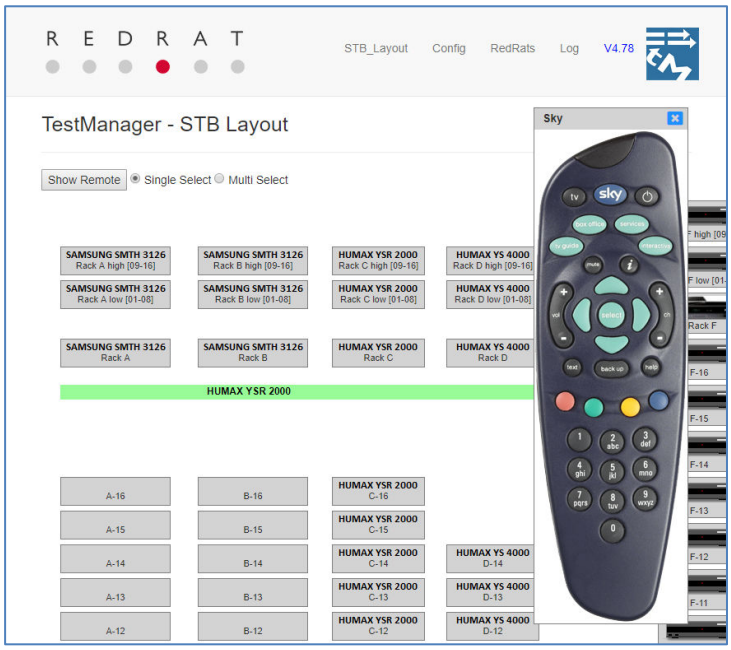


Figure 30. STB Layout shown in a web page

The web server is enabled from the *Misc* tab in the *Options* dialog.

20. TestManager Options

These control various aspects of *TestManager* and can be seen in from the *Edit -> Options* menu item. They have been described where appropriate in the various sections of the document, but they are all listed here.

20.1 Data Source Tab

This controls where *TestManager* stores configuration and script information. See section 4 for further details.

Use internal database: If checked, a local file based database is used for storage, only accessible from the application running on the local machine. This is easier for installation but does not support multiple-users.

File: If an internal database file is used, this gives its location.

Server name: If an external database is used, this is the name of the database server.

Database Login: Details of the login to the database for *TestManager*.

20.2 Scripting Tab

Python Installation Directory: If Python scripting is to be used, IronPython needs to be installed and TestManager told where this is. (See section 14.3.)

Default Script Editor: When double-clicking on a script or using the Edit Script icon, it can be configured as to whether the internal or an external editor is used.

External Python Editor: The external application used for editing Python scripts, plus any command line arguments needed when starting the editor. There is also an option to save the user's preference regarding whether to open Python scripts externally in multiple-document-interface (MDI) mode or not.

External TM Script Editor: The external application for TM Script editing, plus command line arguments. The option of opening scripts externally in MDI mode is also available for TM scripts.

20.3 Script Capture Tab

See section 14.1.4 for further details.

Use On-Screen Remote for Input: Clicking buttons on the on-screen remote control will insert these commands into the script.

User RedRat3 for Input: Select the RedRat3 or RedRat-X device to be used for IR input. When the remote control handset is used, these commands are inserted into the script.

Dataset for IR Signal Matching: To recognize the IR signals from the remote control handset, the correct dataset needs to be set here.

20.4 Control via Remote Tab

When using a physical remote control handset to control STBs via *TestManager*, this sets the IR input mechanism. Section 13 explains this in more detail.

20.5 The Logging Tab

TestManager operation and STB control logging is described in section 15:

Log script execution output to file: Allows selection of a directory into which TestManager will write log files. The name of the output file is the same as the script.

Log script execution to socket: Sets the IP address and port number for a log listening application.

20.6 The Shortcuts Tab

Many menu operations within *TestManager* can be associated with a keyboard shortcut in this option tab for ease or speed of use.

20.7 The Misc Tab

Snap remote controls to screen edge: On-screen remote controls can be "parked" at the screen edge which can make their placement more convenient. When using multiple monitors, this can prevent the remote control window being dragged from one monitor to another, so un-checking this box will allow unrestricted window movement.

On-screen Remote for STB Layout Window: STBs shown in the STB Layout window (section 8.2) can have an on-screen remote control associated with them, however if multiple STBs are selected with different remote controls, which one should be used? This sets the default on-screen remote to use.

This remote control is also used in the web page view of the STB Layout.

irNetBox Auto Disconnect on Idle: If *TestManager* does not send any commands to an irNetBox within a certain period of time, it can be instructed to disconnect from the irNetBox so that another user can access it. By default this is switched OFF, but it can be enabled here, and the idle time before disconnect set.

Enforce Single STB User: By default, *TestManager* will ensure that if a script is controlling an STB, then no other script can use it. It will also prevent control of STBs via any other mechanism, such as on-screen remote controls. There may be circumstances when it is useful to be able to control STBs simultaneously automatically via scripts and manually via on-screen remote controls, so this box should be un-checked.

RedRat Hub: RedRat Hub is an application which can multiplex commands to RedRat devices from many applications and send them on to the correct RedRat device. Section 18 gives full details on the use and configuration of RedRat Hub by *TestManager*.

HTTP Server: The in-build web server can be enabled here, and the port to be used configured. See section 19.

21. Saving/Restoring window properties

For the majority of cases, when a window is opened or closed, TestManager will automatically restore or save the size and location of the window on the screen. To disable this feature, the ctrl key can be pressed while the window is being opened/closed. Note, this does not work when opening windows via keyboard shortcuts. If this feature is disabled while the window is opening, the window will appear in the Windows default location.