

User Manual

Get Console®

RapidSSH®

For Version: 2.01

Date: 20 August 2014

1. Table of Contents

1. Table of Contents	2
2. Introduction to Get Console	5
1. What are the Get Console® and RapidSSH Apps?	5
2. App Requirements	6
3. Terminal Overview	7
3. Main Application Settings	8
1. Terminal Settings	8
1. Columns (Default: 80)	8
2. Rows (Default: 24)	8
3. Scrollback (Default: 100)	8
4. Clear On Connect	8
5. LineWrap	9
6. Text Size	9
7. Console Logging (Default: Off)	9
8. Auto Upload Logs (Default: Off)	9
9. Advanced Terminal Settings	9
2. Script Manager	12
1. Create New Script	13
2. Wait for String Action Item	13
3. Send String Action Item	14
4. Terminate Script Action Item	15
5. Send Hex Bytes Action Item	15
6. Pause Script Action Item	16
7. Display Message Action Item	16
8. Start and Stop Logging Action Item	17
9. Upload Log Action Item	17
3.2.10. Run Script	18
3.2.11. Branch	18
3.2.12. Wait for Regex	18
3.2.13. Set Variable	19
3.2.14. Prompt for Variable	20
3.2.15. Compare Variable	20
3.2.16. Confirm	21
3.2.17. Send Config File	21
3.2.18. Comment	22
3. Keyboard Settings	22
1. Backspace Key	22
2. Enter Key	23
3. Use Option as Ctrl	23

3.3.4. Keyboard Transparency	23
4. Serial Settings	23
1. Auto Connect	24
2. Baud Rate	24
3. Stop Bits (Default: 1 Stop Bit)	24
4. Flow Control (Default: None)	25
5. Parity (Default: Off)	25
6. Databits (Default: 8 Bit)	25
5. Cloud Storage Settings	26
1. Keyboard Auth (SSH)	28
6. SSH Settings	28
1. Keyboard Auth (SSH)	28
2. SSH Agent	28
3. Private Keys	28
7. Sharing Settings	30
1. Remote Server	31
2. Private Server	31
3. Secure Connection (Default: Off)	32
4. Username	32
5. Password	32
8. About Get Console	32
1. Version	32
2. Copyright	32
3. About Get Console	33
4. Send Feedback	33
5. Send App Link to a Friend	33
4. Launching Connections (Session Manager)	33
1. Quick Connect	34
2. Connection Manager	34
1. Creating New Connections via Connection Manager	35
2. Managing Connections in Folders	39
3. In Session Options	40
4.4. Session Sharing	41
1. Start Session Sharing	41
2. Remote User Access	42
3. Stop Sharing Session	43
5. Terminal Features	43
1. Terminal Features	44
1. Keyboard Control	44
2. Keyboard Popup Bar Selector	44
3. Command Manager	45
4. Password Manager	46
5. Clipboard Viewer	46
6. Get Console File System	48
1. Get Console File Types	48

2.	User Created Text Files	48
1.	<i>Importing files from Get-Console.com</i>	49
2.	<i>Importing files from Dropbox account</i>	50
3.	Log Files	52
1.	<i>Log File Naming</i>	52
2.	<i>Uploading Log Files</i>	52
3.	<i>Auto-Uploading Log files</i>	53
4.	<i>.Script and .Connection Get Console Files</i>	54
7. General Troubleshooting		55
1.	Serial Connectivity Issues	55
1.	<i>C2-RJ45 Cable Pinouts</i>	56
2.	<i>Console Cable Not Detected</i>	56
3.	<i>Console Cable Detected, No Output on Screen</i>	57
2.	Detailed Serial Troubleshooting	57
3.	Session Sharing Issues	58
8. Appendix A – Specific Device Serial Port Pinouts		59
1.	<i>ADTRAN 550</i>	59
2.	<i>CBX / PhoneMail – DB9 and DB25 Adaptors</i>	59

2. Introduction to Get Console

1. What are the Get Console® and RapidSSH Apps?

Get Console is an Apple App Store distributed terminal application for iPads and iPhones that allows IT engineers to connect and control RS232 serial ports (such as to the console port of Cisco devices) or run Telnet, RAW, or SSH(v2) terminal sessions over WIFI and 3G.

With the Get Console app, IT engineers can perform IT equipment maintenance, troubleshooting and disaster recovery operations directly on the serial device console ports, or over Wifi/3G using the instant on, portable iPad, iPhone or iPod Touch - all WITHOUT jailbreaking.

The Get Console app now has many features including:

- Physical serial to network and other equipment via the Redpark L2-RJ45V cable
 - supports 300-115200 baud optional parity, optional flowcontrol, variable stop-bits and either 7 or 8 databits
- Serial connectivity via Serial over WIFI / Bluetooth Low Energy adaptors such as the Airconsole adaptor.
- SSHv2, Telnet, and RAW connectivity over WIFI/3G
- VT100, xterm and many other common Terminal emulations
- Encoding support for ASCII, UTF-8 and 17 other major encoding formats, including multibyte characters for non-latin text terminals
- Maintains multiple sessions concurrently, including when the App is in the background for upto 10 minutes.
- Session import from SecureCRT® and PuTTY via Windows tool and iTunes, or via Web converter
- Full Session Scripting using simple “Expect X, then Send Y” type scripting.
- One Tap Secure Screen Sharing allows a remote web user to view and interact with the iPad/iPhones terminal window
- Cloud connectivity to both get-console.com and Dropbox.com for dynamically downloading configuration files and scripts and uploading session log files
- Tight integration with iPhone/iPad clipboard and cut/copy/paste directly from the terminal window
- Comprehensive logging support, Command Shortcuts, Stored Passwords, Bluetooth Keyboard and many other features.

RapidSSH is a cut down version of Get Console app. The major feature missing is the ability to use a Redpark physical serial cable to connect to equipment. RapidSSH only supports Serial connectivity via our “Airconsole” WIFI / Bluetooth Serial adaptor.


2. App Requirements

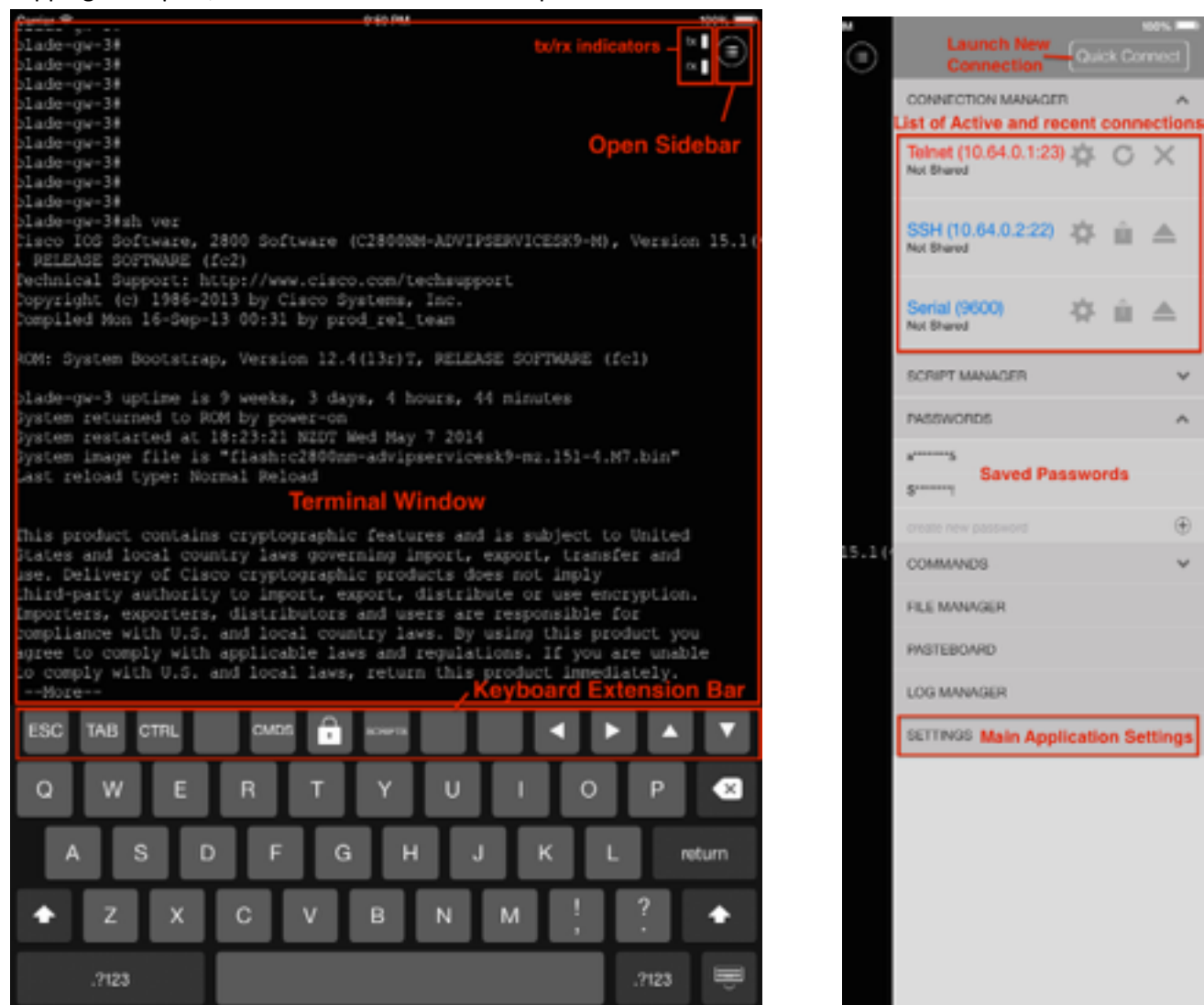
- iPhone4S + (iOS version 7.0 or later is required)
 - o iPhone 4S, iPad2, iPad 3 supports Redpark C2-RJ45V cable
 - o iPhone 5 onwards, iPad 4, iPad Air onwards support Redpark L2-RJ45V cable
 - o All Apps and iOS devices support Airconsole serial adaptor (WIFI or Bluetooth mode)

RapidSSH does not support Redpark Cables. Serial connectivity in RapidSSH is only possible via Airconsole adaptor.

3. Terminal Overview



The below picture shows the main Get Console window and identifies the major UI features. Get Console's UI has changed significantly since earlier 1.x versions.

Tapping the Open / Close Sidebar button  exposes the sidebar



Clicking on the Settings button in the sidebar where most of the apps features are configured.

3. Main Application Settings

The Main Settings page is accessible via the  button then . This sets the application Default settings for all sessions as well as many other features. Some of these defaults can be varied on individual saved connections (See Section 4 Connection Manager). The App settings cover 5 main areas:

- **Terminal Settings** - features specific to the terminal window - things like linewrap, local echo, scrollback size, but also access to the terminal scripting features.
- **Keyboard Settings** - features for how the keyboard behaves (with or without external Bluetooth keyboard running)
- **Serial Settings** - features specific to Serial Connections (ie baud rate)
- **SSH Settings** - features specific to SSH Connections (ie Certificates/Keys, Authentication methods)

Sharing Settings – settings requiring configuration in order to share the terminal window with a remote user via the get-console.com website or privately hosted Get Console server.

Cloud Storage Settings – settings requiring configuration in order to link the Get Console app file system with Dropbox (and in later versions iCloud).

The following section details each of these setting areas.

1. Terminal Settings



1.Columns (Default: 80)

Supports any column width up to 132 characters wide

2.Rows (Default: 24)

Supports any number of rows on the screen

3.Scrollback (Default: 100)

Number of lines retained by the console for review. Setting this above 250 will impact performance on iPhone4 and below. iPad2, iPad3 and iPhone 4S supports 500 scroll back without impact due to improved hardware specifications.

4.Clear On Connect

Defines whether the terminal should clear the screen when starting a new connection

The default is ON

5. LineWrap

Defines whether the terminal should automatically wrap the line to the next line when the output received from the remote device exceeds the column width and no carriage return has been received.

The default is ON

6. Text Size

Sets font size in the terminal window



7. Console Logging (Default: Off)

Logs all printable output to file stored on iOS device. The log files can be uploaded to www.get-console.com website, Dropbox.com folder, Uploaded to a Private Server, attached to email or extracted via iTunes. Log upload is performed via the Files section described below.

8. Auto Upload Logs (Default: Off)

Automatically upload log files to remote storage repository when the session finishes. The default is off. Note that this feature will fail if Wifi/3G connectivity is not available at the time the session finishes

Options:
- Disabled
- Get Console Remote Server
- Dropbox

Note that to upload logs automatically to a users Get Console Public Server, Private Server or Dropbox.com folder requires that the Remote Server username/password be configured in main settings (for Get Console Public or Private Servers), or Dropbox account be linked to Get Console Application (also in Main Settings).

9. Advanced Terminal Settings

1. Bell Behavior

Options:
- Play Alert (bell/vibrate phone)
- Do Nothing (silent)

Allows the iOS device to create an alert when there is an Terminal event or stay silent.

2. *Cursor*

Allow the choice of curs

Options:
- None
- Outline
- Block
- Vertical Line
- Underline

or style – the default is None

3. *Blink Cursor*

Cursor blinks – the default is Off

4. *Colour Scheme*

Changes the colour scheme of the text and background of the console

Options:	
- White on Black	White text on Black background
- Black on White	Black text on White background
- White on Black/Grey	White text on Black/Grey background
- Green on Black	Green text on Black background
- Rainbow	Helps debugging by colouring different parts of the screen <ul style="list-style-type: none">o Blue: Left and right bordero Red: Upper border of the scrollbar limito Black: Active area of the console within the scrollbar limit

5. *ANSI Colour*

Allows up to 256 colours on terminal window following the ANSI spec for signalling colour. The default is ON.

6. *Local Echo*

Controls whether the terminal will echo characters entered to the screen locally or rely on remote serial device to echo them. The default is Auto – where the terminal attempts to recognise if remote serial device is echoing characters, and if so disables local echo. If both local and remote echo are enabled then double characters for every single character typed will appear on terminal window (in which case use “Force Off” setting).

Options:

- Auto
- Force On
- Force Off

7. *TERM ENVIRONMENT*

Free Text field where Telnet/SSH Sessions can send the TERM=[xxx] to the remote server to indicate capabilities. Get Console is a VT100 terminal, however you can configure xterm or other terminal types to improve compatibility with other emulation methods that reject “vt100” terminal settings.

8. *Scroll Mode*

Defines where the terminal view is positioned. In versions prior to v1.82 there was no change to the terminal view position on input from keyboard or output from remote device. This has changed so the default setting is now “Keyboard Input”.

Options:
- None – use touch screen to move around the terminal and scrollbar window
- Keyboard Input (Default) – pressing any key on keyboard will return the window view to the active cursor position within the terminal window
- Terminal Activity – any terminal generated activity will center the terminal view to where the terminal activity is being generated

9. *Character Encoding*

Get Console supports 17 different encoding formats. The default is UTF-8 which supports single and double byte character sets and is the most common character encoding set in modern use. Prior to Get Console v1.7 the default encoding set was ASCII which is still available under this setting.

10. *Paste Rate Limit*

When pasting large text files into the terminal window, especially when using the older C2-RJ45 Serial Console cable, the remote serial device buffer or buffer memory inside the cable can overflow, producing garbled text.

Get Console has a speed control to allow for text from clipboard or scratchpads to be “fed” into the terminal at a slower speed to ensure that the receiving device and C2-RJ45 cable do not overflow their receive buffers. Internal testing has found that for Cisco console ports, configured at 9600baud, the best setting is “High Speed” and hence this is the default. The “High Speed” Setting allows for a 800 line configuration file to be pasted into a Cisco device without error in around 10 seconds.

11. Transfer Protocol

When uploading binary files via Serial cable, Get Console can use a variety of legacy protocols. The default is prompt user to select the protocol when the upload is invoked, however can be set here so user is not asked. The options are:

Options:

- XMODEM	Original X-Modem Protocol
- XMODEM-CRC	Xmodem with CRC checks every 1024 bytes
- XMODEM-1K	Xmodem with ACKs every 1024 bytes – goes faster than Xmodem as less acknowledgements
- YMODEM	YModem – goes faster than XModem
- RAW	Sends raw file data without any protocol framing

2. Script Manager

The script manager allows for the creation of terminal scripts. Once created, these scripts can be run immediately on connection by assigning them to saved connection in the Connection Manager, or alternatively by running once a terminal session is started via the Script button which can be added to the keyboard extension bar.

Scripts are built by adding one or more “action items” to the script in order that the terminal should execute them. When running the script will start with the first action item and progress through the action items in order.

Usually the script will begin by telling the terminal to wait to see something on the screen (ie the “Wait-for-String” action item) and then the next action item will be to do something (send command, start logging etc). These type of scripts are known as “Expect Scripts” in that while the script is running it will “Expect” to see “X” on the screen, and when it does it will respond with “Y”. After creating the script and running it the items are usually completed sequentially from first item to last - however complex scripts can manipulate the execution order via “branching” and “jump” action items.

The below section describes the various action item options are and their configurable parameters. Note that poorly constructed scripts will possibly cause the terminal window to lock or perform unpredictably. A common cause for script problems not configuring a timeout for a Wait-for-String action item – if this is not configured, the script will never progress to the next action item as it has not seen the string it is looking for.

1. Create New Script

To create a new script, navigate to the Script Manager and tap the Create New Script button. Give the script a name, and then tap on Action 1.

Cancel Script Details Save

Script Name New Script 1

ACTIONS

1. Wait for String >

Add New Action

Reorder Actions

By default the first action is Wait for String, however it can be any of the following action items.


Change by tapping item 1 which then allows one of all of the possible action types to be selected.

At the bottom of the screen is where the parameters for the action item is configured, along with some guidance text on which and the type values that can be entered.

Script Details	Action Details
ACTION TYPE	
Wait for String	>
Wait for String	> ✓
Send String	
Send Hex Bytes	
Terminate Script	
Pause Script	
Millisecond Pause	
Display Message	
Comment	
Jump to Action	
Start Logging	
Stop Logging	
Upload Log	
Run Script	
Branch	
Wait for Regex	
Set Variable	
Prompt for Variable	
Compare Variable	
Confirm	
Send Config File	

2.Wait for String Action Item



Note that the Wait for String action is quite powerful in that it can be used to create loops via the “Rule on timeout” parameter. By default the script executes each item in the script sequentially, however using the Rule on timeout, the script could alternatively jump to earlier or later action items. For example if the script must keep pressing enter until it sees “End of File” then the Wait For String rule is configured to wait to see “End of File”, and if it does not see it after 2 seconds then

ACTION TYPE
Wait for String 
ACTION PARAMETERS
String
Timeout
Rule on timeout
ACTION HELP
Wait for String: Waits for the specified string to appear on the console before continuing to the next script rule. - String: The string to wait for - Timeout: The maximum time in seconds to wait. Use blank for no timeout - Rule on timeout: If the timeout occurs execution will continue from this rule number. Use blank to continue at next rule

it jumps back to an earlier “Send String” action item that sends the enter key.

3. Send String Action Item

Use the send string to send normal ASCII text to the terminal window, with or without the enter key after it. If escape codes (ie CTRL-X) are needed to be sent, instead use the alternative “Send Hex

< Script Details	Action Details
ACTION TYPE	
Send String 	
ACTION PARAMETERS	
String	
Send Newline 	
ACTION HELP	
Send String: Sends the specified string to the console. - String: The string to send - Send Newline: On - will send the string followed by the configured enter character, Off - will send just the string. To send the contents of a stored Variable rather than static text, then use \$VARIABLE in the String field.	

Bytes” Action item.

< Script Details	Action Details
ACTION TYPE	
Terminate Script 	
ACTION HELP	
Terminate Script: Stops processing script rules.	

4. Terminate Script Action Item

By default when the script completes its last action item it will terminate by itself, therefore this action is only used where jumps between rules are configured (see Wait-for-String above) and one leg of a script branch requires the script itself to terminate.

5. Send Hex Bytes Action Item

The screenshot shows a web interface for configuring an action. At the top, there are two tabs: 'Script Details' (with a back arrow) and 'Action Details' (which is active). Below the tabs, the 'ACTION TYPE' is set to 'Send Hex Bytes'. Under 'ACTION PARAMETERS', there is a section titled 'Hex Data'. At the bottom, an 'ACTION HELP' section explains that the action sends specified hex bytes to the console, with an example: 'Hex Data: bytes encoded in hex - e.g. 0304 for Ctrl-C, Ctrl-D'.

Other common hex byte codes are:

Hex Code	Sends
0D	Carriage Return
09	Tab
08	Backspace

Find more hex byte codes at www.asciitable.com

6. Pause Script Action Item

Use the Pause action to halt the script for a number of seconds – this is useful when the device connected to will not be ready to accept a “Send String” or “Send Hex Bytes” action immediately after a “Wait for String” has been matched. The pause action is also useful to give the remote device time to process the previous “Send String” before sending the next one.

< Script Details	Action Details
ACTION TYPE	
Pause Script	
ACTION PARAMETERS	
Timeout	
ACTION HELP	
Pause Script: Pauses script processing for the specified number of seconds. - Timeout: The number of seconds to wait	

7. Display Message Action Item

Display Message is useful to provide feedback to the iPad/iPhone user that the script has completed or is at a certain point. The pop-up message does not have to be acknowledged for the script to continue or complete.

< Script Details	Action Details
ACTION TYPE	
Display Message	
ACTION PARAMETERS	
String	
ACTION HELP	
Display Message: Displays a popup message to the user. -String: The message to display	

8. Start and Stop Logging Action Item

Note that these actions override the default Terminal settings for Logging. If Logging is already enabled in the main App settings, there is no need to “Start Logging”, unless it had been previously stopped with the “Stop Logging” Action. These actions are useful if just a subset of terminal data is required to be captured. The parameters allow the log file to be appended to the existing session log file rather than a new one be created.

ACTION TYPE

Start Logging

ACTION PARAMETERS

Log Name

ACTION HELP

Start Logging: Starts logging console output. If console output is already logging this will restart the logging.
- Log Name: The name to use as the base for the log file. Use blank for default naming. If the filename does not contain a '.' then the date and extension .txt will be appended
To use todays date and time in the log file use \$DATE and or \$TIME in the Log name - i.e. Log Name: DeviceLog_\$DATE_\$TIME.log
- Overwrite: If Log Name is specified (and contains a dot) and the file already exists, this flag controls whether the existing log file will be overwritten or appended.

9. Upload Log Action Item

Upload log file created during the session (either because Logging is enabled by default, or due to Start Logging earlier action item). The log file can be either uploaded to the users portal space on the www.get-console.com website (Get Console option), or if the user has linked Get Console to their Dropbox.com account then uploaded to the My Apps/Get Console/ folder within their Dropbox.

< Script Details

Action Details

ACTION TYPE

Upload Log

ACTION PARAMETERS

Upload to

ACTION HELP

Upload Log: Uploads the console log to the configured remote server. Displays an error if no remote server is configured.

< Script Details	Action Details
ACTION TYPE	
Run Script ▼	
ACTION PARAMETERS	
Script Name >	
Return Control <input type="checkbox"/>	
ACTION HELP	
<p>Run Script: Launches another script. Optionally returns execution to this script after completing.</p> <p>- Script Name: The name of the script to run</p> <p>Return Control: On - after completing the specified script control will return to the next rule in this script, Off - control will not return to this script after the specified script has completed</p>	

3.2.10. Run Script

This action triggers another script to be launched. With is also having the Return control option which determines whether or not the script control will be passed on to the next rule in the script or not.

3.2.11. Branch

This action can be configured to create jumps in the scripts according what string is seen first. You are able to configure the string that initiates the jumps as well as the rule in the script that the jump should be made to. A timeout option is also available so the next rule in the script will continue being processed if none of the specified strings are seen within an allocated time frame. While the rule number that the script jumps to is a time out occurs can also be specified.

< Script Details	Action Details
ACTION TYPE	
Branch ▼	
ACTION PARAMETERS	
If this is seen	
Jump to rule number	
If this is seen	
Jump to rule number	
If this is seen	
Jump to rule number	
If this is seen	
Jump to rule number	
Timeout	
Rule on timeout	
ACTION HELP	
<p>Branch: Executes a jump in the script based on which string is seen first.</p> <p>- If this is seen: The string to wait for</p> <p>Rule Number: The rule in the script to jump to if the string is detected</p> <p>- Timeout: If none of the strings is seen within this number of seconds then continuing processing at the next rule. Use blank for an indefinite wait</p> <p>- Rule on timeout: If the timeout occurs execution will continue from this rule number. Use blank to continue at next rule</p>	

3.2.12. Wait for Regex

Like the Branch action this action creates rule jumps when a certain Regex (Regular Expressions) appear as well as a matching timeout feature. It also has an extra feature a Variable which provides a storage location for the first capture group (full match is also stored in the system variable \$LAST-

The screenshot shows the 'Action Details' screen for the 'Wait for Regex' action. The interface includes a back arrow and 'Script Details' label at the top left. The main title is 'Action Details'. Below this, the 'ACTION TYPE' is 'Wait for Regex'. The 'ACTION PARAMETERS' section contains several fields: 'If this is seen' (repeated four times), 'Jump to rule number' (repeated four times), 'Variable', 'Timeout', and 'Rule on timeout'. The 'ACTION HELP' section at the bottom provides detailed instructions: 'Wait for Regex: Executes a jump in the script based on which regular expression is matched first. - If this is seen: The regular expression to wait for - may optionally contain a capture group. Rule Number: The rule in the script to jump to if the regex matches. Variable: Store the first capture group in this variable (full match is additionally stored in the system variable \$LASTMATCH). - Timeout: If none of the regular expressions match within this number of seconds then continuing processing at the next rule. Use blank for an indefinite wait. - Rule on timeout: If the timeout occurs execution will continue from this rule number. Use blank to continue at next rule.'

MATCH.)

3.2.13. Set Variable

Sets a variable to a specified value. This can then be used in almost all the same places that strings are required by prefixing the variable name with a \$ character. The String action parameter is the value that the variable will be set to. Also the App will automatically prefix the entered Variable name with a \$.

The screenshot shows the 'Action Details' screen for the 'Set Variable' action. The interface includes a back arrow and 'Script Details' label at the top left. The main title is 'Action Details'. Below this, the 'ACTION TYPE' is 'Set Variable'. The 'ACTION PARAMETERS' section contains two fields: 'Variable' and 'String'. The 'ACTION HELP' section at the bottom provides detailed instructions: 'Set Variable: Sets a variable to the specified value. The variable can be used in most places where a string is required by prefixing the variable name with a \$ character. - Variable: The name of the variable to set. App will automatically prepend the \$ symbol to this name. - String: The value to set the variable to.'

3.2.14. Prompt for Variable

This action prompts the user for a variable. With it mostly able to be used where a string is as long as \$ is prefixed to the name. The String is the message (max 20 characters) shown to the user during the action. While the Secure button determines if the txt is obscured or not.

The screenshot shows the 'Action Details' configuration screen for the 'Prompt for Variable' action. The interface includes a back arrow and 'Script Details' link at the top left. The 'ACTION TYPE' is 'Prompt for Variable'. Under 'ACTION PARAMETERS', there are three fields: 'Variable', 'String', and 'Secure' (a toggle switch). The 'ACTION HELP' section at the bottom provides a description and a list of parameters.

Action Details	
ACTION TYPE	
Prompt for Variable	
ACTION PARAMETERS	
Variable	
String	
Secure	<input type="checkbox"/>
ACTION HELP	
<p>Prompt for Variable: Prompts the user for a variable. The variable can be used in most places where a string is required by prefixing the variable name with a \$ character</p> <ul style="list-style-type: none">- Variable: The name of the variable to set. App will automatically prepend the \$ symbol to this name.- String: The message displayed to the user during the prompt (Maximum 20 characters)- Secure: determines if the text entry is obscured	

3.2.15. Compare Variable

This action compares the variable to an allocated value or another variable. If matching a jump will occur with the Action Number specifying the rule number to jump to. The String is what will be compared to the Variable to see if they match.

The screenshot shows the 'Action Details' configuration screen for the 'Compare Variable' action. The interface includes a back arrow and 'Script Details' link at the top left. The 'ACTION TYPE' is 'Compare Variable'. Under 'ACTION PARAMETERS', there are three fields: 'Variable', 'String', and 'Action Number'. The 'ACTION HELP' section at the bottom provides a description and a list of parameters.

Action Details	
ACTION TYPE	
Compare Variable	
ACTION PARAMETERS	
Variable	
String	
Action Number	
ACTION HELP	
<p>Compare Variable: Compares the variable against a specified value or another variable. If the two match then jump to the action number given</p> <ul style="list-style-type: none">- Variable: The name of the variable to check (ie \$VARIABLENAME)- String: The string to check against - use a \$ character to use a variable- Action Number - the rule number to jump to if the two values are equal	

3.2.16. Confirm

This action displays a yes or no confirmation dialog to the user. The String specifies the message that will be used to prompt the user. Action Number indicates the rule that will be jumped to if “Yes” is entered. If “No” is tapped then it will continue to the next rule.

< Script Details	Action Details
ACTION TYPE	
Confirm	
ACTION PARAMETERS	
String	
Action Number	
ACTION HELP	
Confirm: Displays a confirmation dialog to the user prompting for a yes/no answer - String - the message to prompt the user with - Action Number - the rule to jump to if "Yes" is pressed. If "No" is pressed, execution continues at the next rule	

3.2.17. Send Config File

Sends a configuration file line by line to the terminal with blank lines skipped. The File name is the file that will be sent. It is important to include a pathname for the ‘files’ folder. Pause allow you to control the number of milliseconds to wait between sending each line to the terminal. Blank indicates no pause.

< Script Details	Action Details
ACTION TYPE	
Send Config File	
ACTION PARAMETERS	
Filename	
Pause	
ACTION HELP	
Send Config File: Sends a configuration file line by line to the terminal. Blank lines are skipped - Filename - the file to send. This should refer to a pathname within the 'files' folder - Pause - the number of milliseconds to wait between sending each line. Use blank for no pause	

3.2.18. Comment

Simply allows you to place a comment in the script. Action has no effect.

The screenshot shows the 'Action Details' screen for the 'Comment' action. At the top, there are two tabs: 'Script Details' and 'Action Details', with 'Action Details' being the active tab. Below the tabs, there are three main sections: 'ACTION TYPE', 'ACTION PARAMETERS', and 'ACTION HELP'. The 'ACTION TYPE' section shows 'Comment' with a dropdown arrow. The 'ACTION PARAMETERS' section shows 'String'. The 'ACTION HELP' section contains the text: 'Comment: Places a comment in the script. This action has no effect.'

3. Keyboard Settings

There are 4 configuration items under the Keyboard Settings section of the main App Settings.

The screenshot shows the 'KEYBOARD SETTINGS' section. It contains three items: 'Backspace Key' with a value of 'Control-H' and a right arrow; 'Enter Key' with a value of 'Carriage Return' and a right arrow; and 'Use Option as Ctrl' with a green toggle switch that is turned on. Below these items is a slider control with a blue bar and a white knob, flanked by two small keyboard icons.

Note: that it is possible to use the Get Console app with a Bluetooth keyboard and the settings configured here will affect the Bluetooth keyboard. Note also that full Bluetooth Keyboard support is not yet implemented as at version 2.01 – ESC and Arrow keys will not work on Bluetooth keyboards.

1. Backspace Key

Certain terminals expect Backspace to be sent as CTRL-? (Delete – ASCII 0x7F) rather than the Backspace (CTRL-H – ASCII 0x08). The default setting is CTRL-H.

Options:
- Control-H
- Control-? (127)

2. Enter Key

Get Console has 3 options for what is sent when the enter key is pressed. By default it sends just Carriage Return (0x0D) which is the OSX standard. This can be changed to Line Feed (0x0A) or both Carriage Return followed by Line Feed (0x0D0A) each time the enter key is pressed.

Options:
- Carriage Return
- Line Feed
- CR + LF

3. Use Option as Ctrl

Get Console implements a workaround for the control key not working on Bluetooth keyboards. If using a Bluetooth keyboard with Get Console, you can select the “Use Option and Ctrl” to use the Alt/Option key on the keyboard to send CTRL-[key] sequences. The default setting is OFF.

3.3.4. Keyboard Transparency


On the newer version of the Get Console application the keyboard can be toggled between transparent and solid on the keyboard itself with the toggle keyboard icon. The slider on keyboard settings allows you to control the transparency of the keyboard when in transparent mode.

4. Serial Settings

Get Console’s Serial Settings allow for the full range of physical serial settings available on the Redpark L2 or C2-RJ45V adaptor to be set.

Get Console also supports Serial connections over Bluetooth Low Energy or WIFI using our Airconsole adaptors. When the RJ45 based serial cable is further adapted to either DB9 or DB25 presentation, only the pins available on RJ45 connector can be carried through – ie Pin 9 on DB9 connector (Ring Indicator) cannot be carried.

Below describes the parameters available for each Serial option:

SERIAL SETTINGS		
Auto Connect		
Baud Rate	9600 Baud	>
Stop Bits	1 Stop Bit	>
Flow Control	None	>
Parity	None	>
Data Bits	8 Bits	>

1. Auto Connect

Get-console app automatically connects to the physical cable connection (RJ-45 or serial) when it's plugged into the iPhone/iPad/iPod port. This applies whether or not there is an existing Telnet or SSH connection (a new Serial connection will be added to your connection list). The default is ON

2. Baud Rate

Allows the choice of different baud rates to suit communication with the network device. Failure to select the correct Baud rate for your serial device will result in either no output or garbled screen output.

Note while Get Console will build a higher speed (57600 or 115200 baud) connection over Bluetooth Low Energy (BLE), the underlying BLE transmission rate is not this fast so buffering and in some cases packet loss can occur.

Options:
- 1200 Baud
- 2400 Baud
- 4800 Baud
- 9600 Baud
- 19200 Baud
- 38400 Baud
- 57600 Baud - not recommended for BLE
- 115200 Baud – not recommended for BLE

3. Stop Bits (Default: 1 Stop Bit)

Options:
- 1 Stop Bit
- 2 Stop Bits

4. Flow Control (Default: None)

Use flow control where your serial device requires it. Examples of devices that require Hardware Flow control to be enabled include older Cisco 3500XL/2900XL series. If the Flow control setting is changed the Redpark C2-RJ45 cable must be removed and reinserted to reset its configuration. Flow control changes on Airconsole occur immediately.

Options:
- None
- Hardware (RTS/CTS)
- Hardware (DSR/DTR)
- Software (XON/XOFF)

5. Parity (Default: Off)

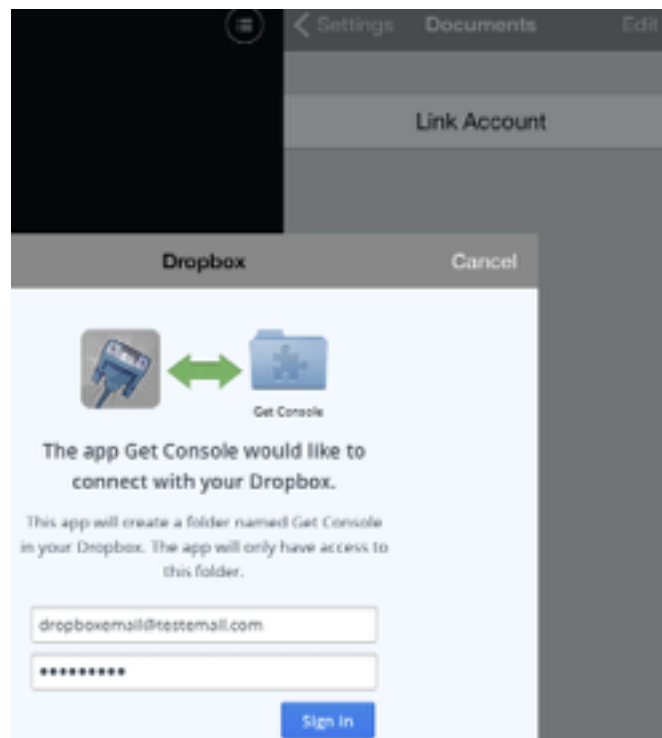
Options:
- Off
- Odd
- Even

6. Databits (Default: 8 Bit)

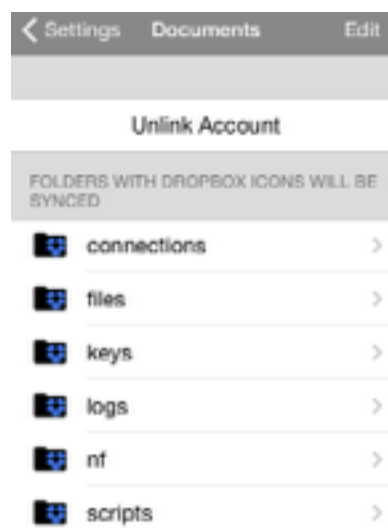
Set Databits required by your serial device.

Options:
- 8 Bits
- 7 Bits

5. Cloud Storage Settings



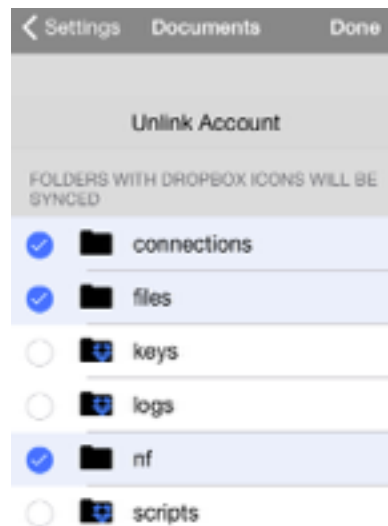
In version 1.120 and later of Get Console (including 2.0), Dropbox has become tightly integrated with the Get Console file system. Linking Get Console to Dropbox allows for the synchronisation of the local Get Console App file system with a dedicated Dropbox/Apps/Get Console/ folder (and sub folders) stored on the users Dropbox account.



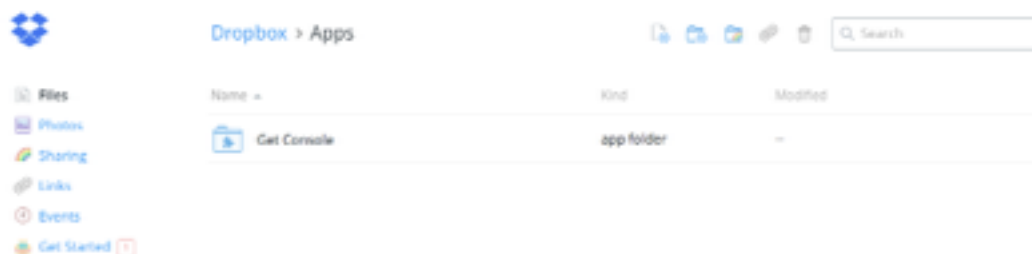
To link Get Console to a users account use the Sync Settings button

By default all folders on the Get Console app are synced with Dropbox, however this can be changed by using the edit button after logging into your Dropbox account and entering sync settings in the main settings side bar.

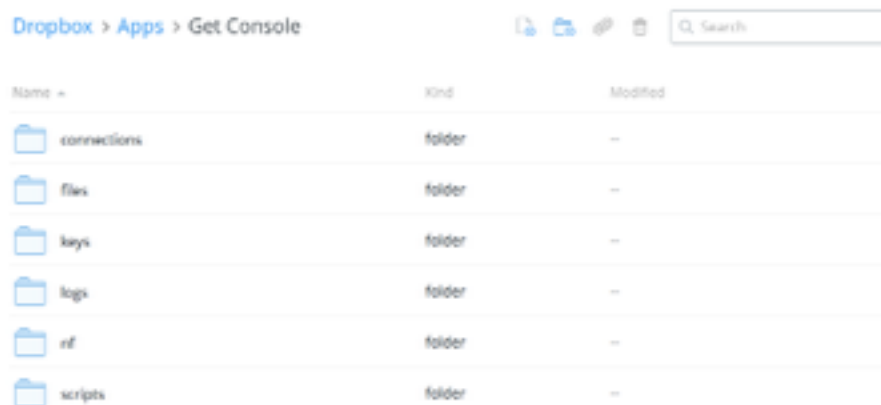
Those folders that appear with the Dropbox symbol on them are the ones that are being synced. While those that have a blue tick are ones that you are choosing not to sync.



To unlink a linked account use the Unlink button. No files are deleted from Dropbox or Get Console's local file system when accounts are linked or unlinked.



From within your Dropbox account online under apps there will be a Get Console folder that will house all of your folders. So you are able to access all your folders whether they are currently being synced or not will determine how up to date they are.



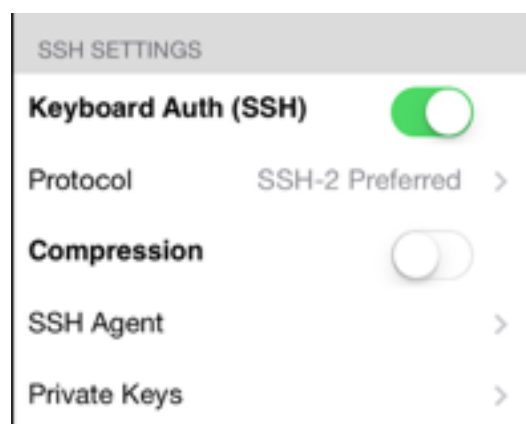
1. Keyboard Auth (SSH)

Allows for interactive keyboard password authentication in addition to the usual SSL security. Use Keyboard Interactive method to SSH into servers that require this method – for example to SSH to Mac OSX Lion Server requires “Keyboard Interactive” method. Try alternating the method if you can launch and to connect to an SSH server but no login or password: prompt appears. The default method is OFF.

6. SSH Settings

Get Console supports SSH version 2 only. SSH version 1 support will be added in a future release.

The SSH Settings section is used to determine the default authentication method when password authentication is used, and also allows for the import of OpenSSH format certificates for use in SSH connections that use certificate based authentication.



1.Keyboard Auth (SSH)

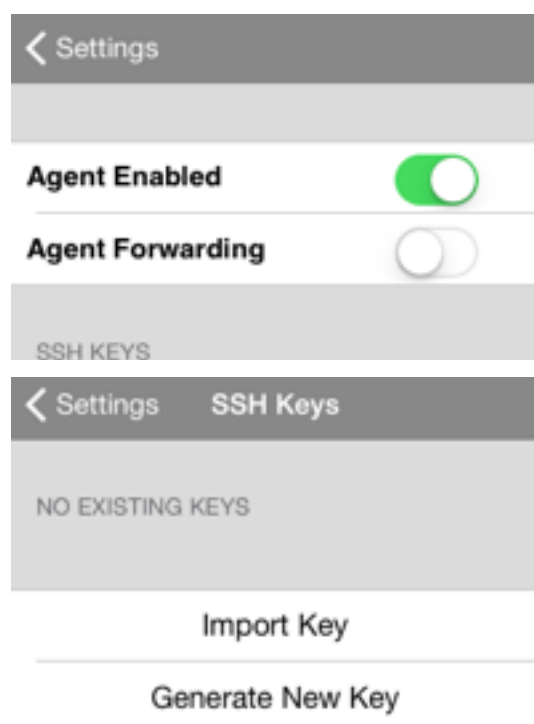
Allows for interactive keyboard password authentication in addition to the usual SSL security. Use Keyboard Interactive method to SSH into servers that require this method – for example to SSH to Mac OSX Lion Server requires “Keyboard Interactive” method. Try alternating the method if you can launch and to connect to an SSH server but no login or password: prompt appears. The default method is ON.

2.SSH Agent

SSH Agent allows Get Console to store and reuse RSA keys as required by end user to avoid having to re-enter key pass phrases during subsequent authentication challenges by SSH Server. If SSH Agent Forwarding is also enabled, then the SSH server can also use the same keys on ssh sessions originated by the user from one SSH server to another.

3.Private Keys

Allows for private keys to be imported and used in console sessions



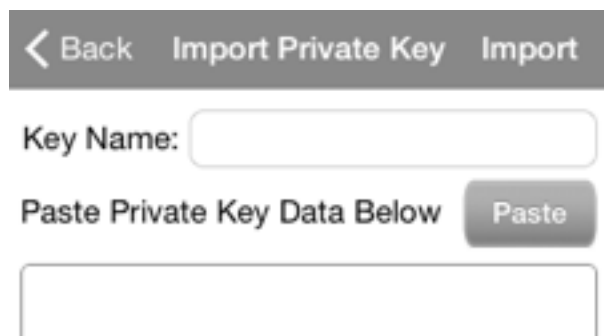
1. *Import Key:*

- Key Name:

Define a key name for identification purpose when assigning to SSH sessions.

- Paste Private Key Data Below:

Paste the OpenSSH formatted key data into this window then press “Import” button. If successful a dialog box will appear confirming import and the key will be available to use / assign to saved SSH connections.



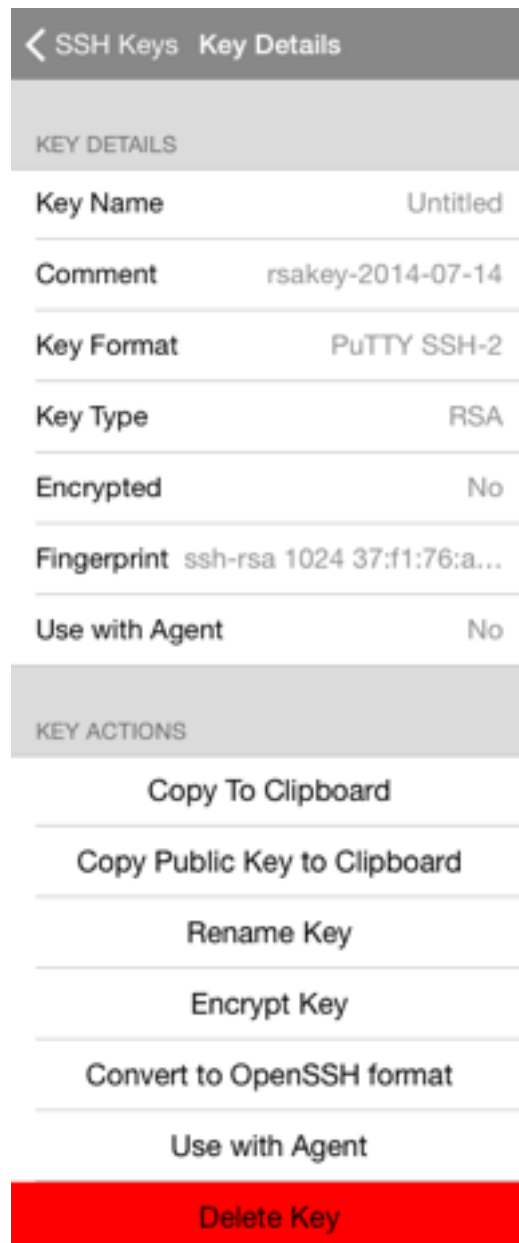
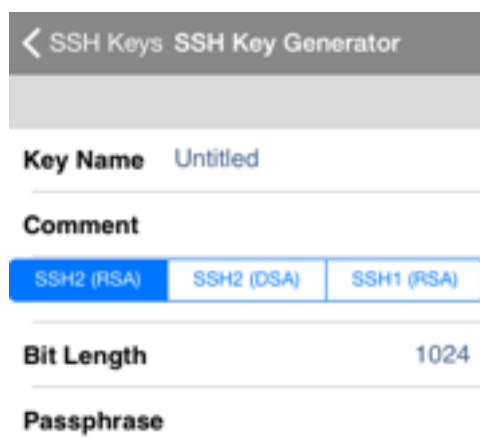
2. *Generate Key:*

Get Console can generate new RSA (SSH2), DSA or RSA (SSH1) keys for use in SSH sessions. These keys are stored in the /keys directory and will be synced also with Dropbox.

Options for generating keys

- Key Name:

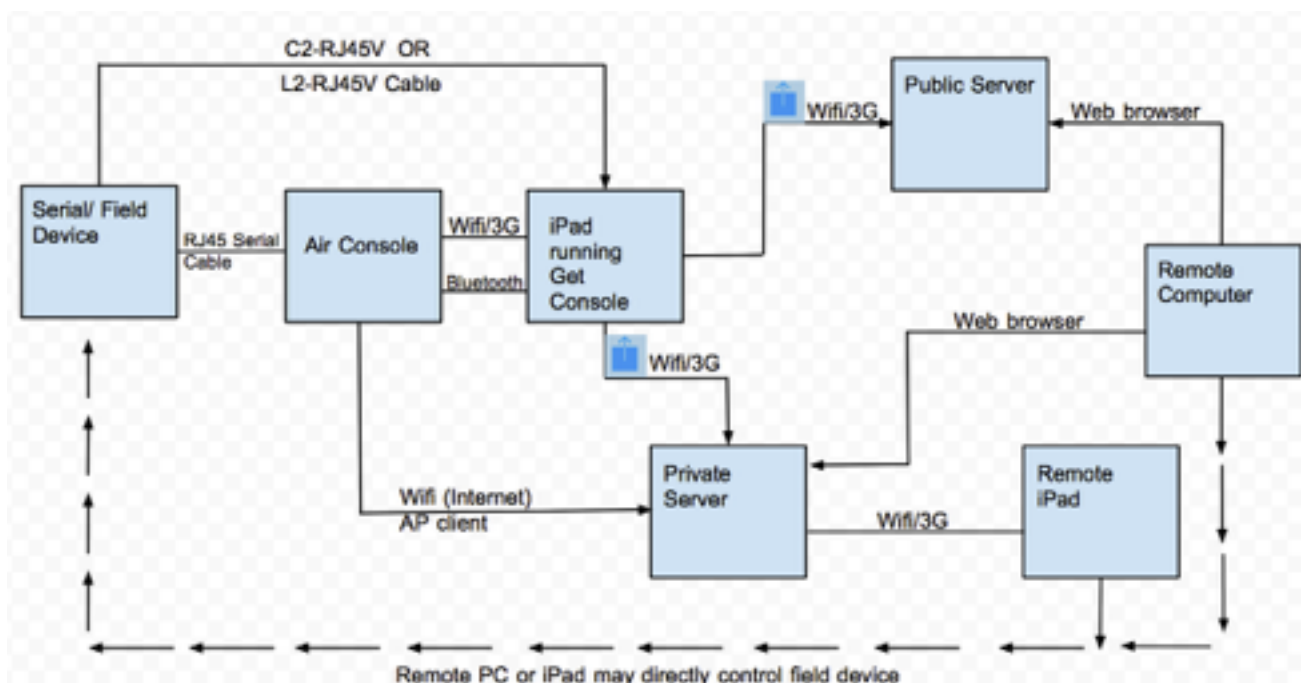
Define a key name for identification purpose when assigning to SSH sessions.



7. Sharing Settings

Get Console allows you to share your terminal window(s) with remote web users or other Get Console users.

Each concurrent terminal session on the iOS device can be shared independently with the get-console.com website or a customers own hosted Private Server. Once sessions are shared (via individual one-time token codes) they can be accessed by other Get Console users or via a web browser assuming those remote users know the one-time code. The below drawing provides a overview of how the session sharing feature works:



In order to enable session sharing, the “Sharing Sessions” details must be populated in the App settings.

SHARING SETTINGS

Remote Server Private Server >

Private Server usdemo.get-console.com

Secure Connection ☐

Username [demo](#)

Password [••••](#)

1. Remote Server

Select the closest Public server to the iPad/iPhone user for session sharing with a remote user, or if Private Server has been installed then select Private Server.

Public versions of the Server are hosted in US, UK and New Zealand. These can be used at no charge by any registered purchasers of the Get Console App to allow remote web access the terminal session via the public website www.get-console.com. The performance of the end to end terminal varies greatly on the current load of the server and the latency of both the Apple device and the Remote web users from the selected public server.

Options:	
- Asia Pacific	
- Europe	
- North America	
- Private Server	The Private Server is hosted on a customer's own network or at Amazon EC2, and secured by their own network security policy
- Disabled	The option to share remote session will be disabled

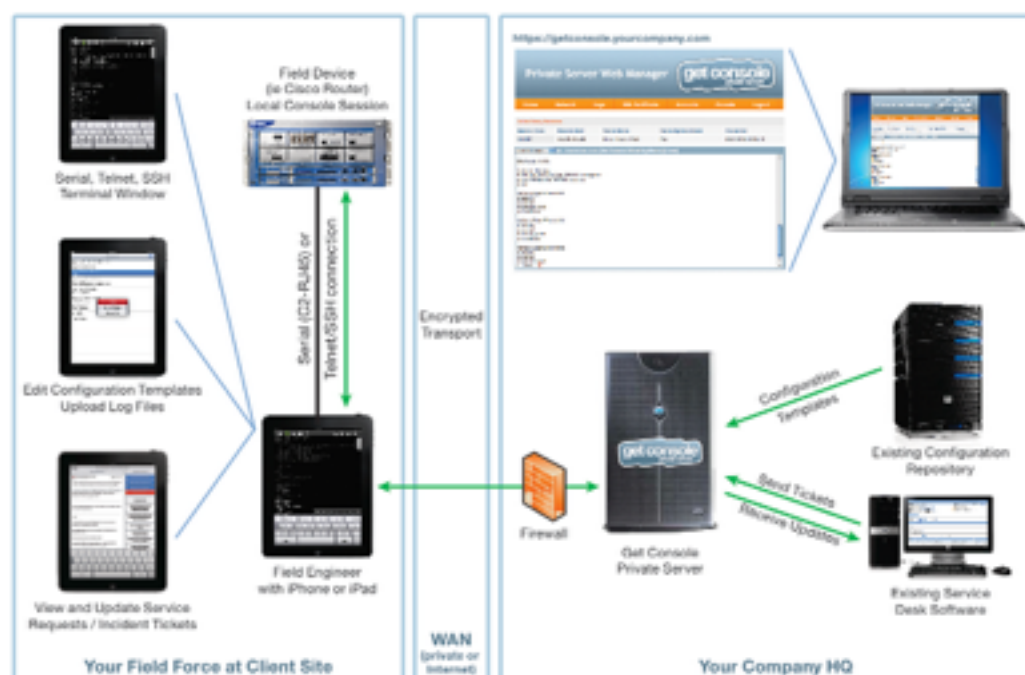
2. Private Server

Private Server is an optional server side component to Get Console that allows for end users to host their own Get Console server on their own network and infrastructure.

If Private Server is selected for Remote Server, set its hostname or IP address in this field.

Below is the diagram representation of the working of the Private Server

For more information, please view the Get Console Private Server manual available from www.get-console.com/private-server



3. Secure Connection (Default: Off)

Use SSL for session sharing for more secure connection. SSL option should only be selected with Private Server if the Private Server has a valid and publicly trusted (ie trusted natively by iPhone / iPad) SSL certificate for the entered hostname. If secure connection is enabled, and the target Private Server does not have an SSL certificate properly installed then a generic “Failed to Connect to Private Server” error will display.

Enabling Secure Connection also has a small impact on the remote performance. For maximum responsiveness when encryption is not required leave Secure Connection OFF.

Options:
- On
- Off

4. Username

Registered email address on www.get-console.com – it’s free to register. Or if Private Server has been chosen, the user name for use with the Private Server login

5. Password

Password for your registered account on www.get-console.com website. Or if Private Server has been chosen, the password for use with the Private Server login.

8. About Get Console

ABOUT GET CONSOLE	
Version	1.120
Copyright	Copyright 2014 Amix Capi...
About Get Console	>
Send Feedback	>
Send App Link to A Friend	>

1. Version

Version number of the Get Console App. Certain features are only available in later versions. Updates to Get Console occur automatically via the App Update process in the iTunes App Store.

2. Copyright

Year and holder of the copyright. Get Console is a registered trademark owned by Cloudstore Limited.

3. About Get Console

About Get Console app – Displays version, copyright information, developers, LibSSH2 license and any other licenses for third party software that is incorporated into Get Console.

4. Send Feedback

This will automatically bring up an email addressed to support@cloudstore.com with which you can provide your feedback with. Your feedback is valuable to us so if you discover any issues with our product please do not hesitate in letting us know so we can fix them as quickly as possible.

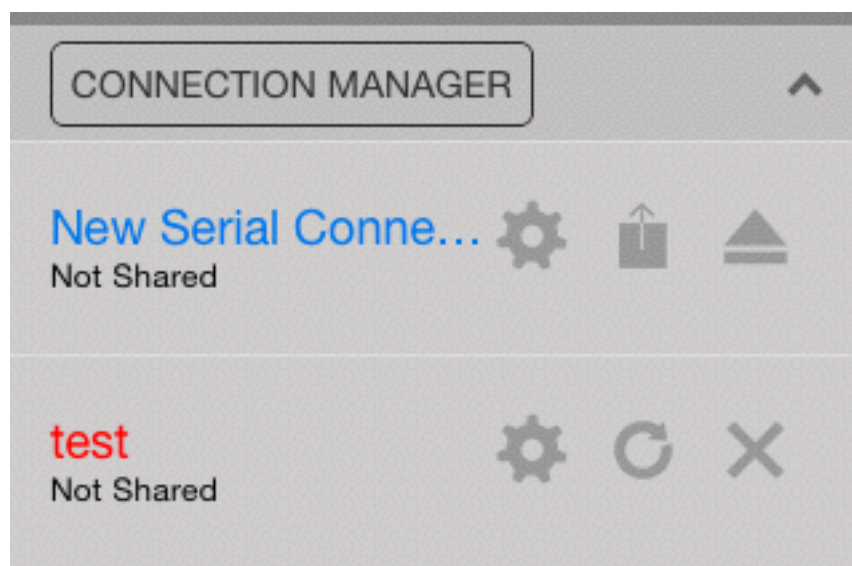
5. Send App Link to a Friend

This will create an email with the iTunes link to the Get Console application that you can send to people who may find it useful.

4. Launching Connections (Session Manager)

Pressing the Connection manager button will launch the new connection Session Manager bar (iPad) or full screen Session Manager (iPhone). Get Console provides 2 ways to launch connections – first via the plus button for a Quick Connection, or via the Connection Manager to launch a saved or recent (unsaved) quick connection. The below drawing highlights the main features of the Connection Manager.

As can be seen, the Connection Manager pull down shows all active and also recently closed sessions (that can be restarted). Within each session the options (ie like scrollbar length, window size etc) can be varied from the Settings defaults via the settings cog button on each connection bar. Any live session can be shared a remote user via the “share” button and the session can be stopped via the “eject” button.



If a session is stopped or times out, then it still remains visible in the Session Manager until it is deleted via the X button or is restarted via the restart button.

There are hardware limits to the number of concurrent sessions that can be run in Get Console, depending on the Apple iOS device in use. The below table provides these limits:

Apple Hardware	Number of Concurrent Live Sessions Allowed
----------------	--

iPhone 4S	8
iPad 2	5
iPad 3	8
iPod Touch 4	5
iPhone 5 / iPad 4 and later	12

1. Quick Connect

Pressing the Quick Connect button on the top of the main menu bar the Quick Connect option is launched. This dialog allows for the instant launching of a new Telnet, SSH, Raw or Serial connection in addition to launching an automatically saved recent connection (Either Quick connected or launched from the Connection Manager).

Depending on the type of connection, various session parameters (such as hostname/IP address) will need to be completed and then tap the Connect button.

Get Console can only maintain a single live Serial connection in the Session Manager as only one adaptor can be connected, however it can store multiple instances of serial connections each with a different session options (ie different baud rates) to make it easy to swap between serial connection templates.

2. Connection Manager

The Connection Manager is used to create, modify and launch saved connections. Tapping Connection Manager in the Session Manager popup will launch Connection Manager where all saved connections can be administered.

Creating new connections can be done one of 3 ways

- 1) Via the Connection Manager “New Connec-

CancelQuick Connect

QUICK CONNECTION

SSHTelnetSerialRemote

Hostnamelocalhost

Port22

Username

SSH KeyNone >

CONNECT

RECENT CONNECTIONS

telnet

test new*i* >

telnet

testing1*i* >

Serial

Serial*i* >

telnet

test*i* >

telnet

Telnet (10.64.8.1:23)*i* >

Airconsole (BLE)

Airconsole (BLE)*i* >

Remote 0081358

Remote 0081358

Remote 0081358

Remote 0081358

Serial

Serial*i* >

Airconsole (BLE)

Airconsole (BLE)*i* >

Clear Recent Connections

tion” dialog box (discussed in this section)

- 2) By converting a Recent Quick Connection to a saved connection via the blue arrow (Give the recent connection a Description and then tap Save)
- 3) By importing “.connection” files into Get Console via iTunes. Get Console provides a free tool to convert existing PuTTY and SecureCRT .ini files into Get Console “.connection” files. This tool can be downloaded from www.get-console.com/tools . The method to import the “.connection” files into Get Console via iTunes is documented at www.get-console.com/tools/importutility/

1. Creating New Connections via Connection Manager

1. SSH

For creating new SSH connections, complete the following items

Connection Name: Define a name for this connection

Hostname: Enter hostname or IP address of the device to connect to via SSH

Port (Default: 22): Enter port number of the device to connect to via SSH

Username: Enter username to login to the device

SSH Key: Choose an SSH key to login to the device if using certificate based authentication. The default is None

Terminal Settings: This section allows for the override of the Main Settings default Terminal Settings to apply to this connection. Change any of the allowed settings.

Click CONNECT when complete

The screenshot shows the 'Connection Details' screen in the Get Console app. At the top, there are navigation buttons: '< Back', 'Connection Details', and 'Save'. Below this is a section for 'Connection Name' with the text 'new SSH Connection'. There are four tabs: 'SSH' (selected), 'Telnet', 'Serial', and 'Remote'. A large 'CONNECT' button is visible. Below the tabs is the 'SSH SETTINGS' section, which includes fields for 'Hostname', 'Port' (set to 22), 'Username', 'Protocol' (set to 'Default' with a chevron), and 'SSH Key' (set to 'None' with a chevron). At the bottom is the 'TERMINAL SETTINGS' section, which includes 'Columns' (set to 'Default') and 'Rows' (set to 'Default').

2. Telnet/RAW

For creating new Telnet connections, complete the following items

Connection Name: Define a name for this connection

Hostname: Enter hostname or IP address of the device to connect to via Telnet

Port (Default: 23): Enter port number of the device to connect to via Telnet. Note that if a non-standard port is chosen, the Telnet preamble will still be sent. If a connection over a non standard port is required without the Telnet preamble (ie for when “Telnetting on port 80 to a webserver” to check that it responds.

If you want to create a RAW connection simply follow the same steps as the Telnet Connection but click the button next to “As RAW:” ensuring that it is green.

Port (Default: 23): Enter port number of the device to connect to via Telnet. Note even if port 23 is selected the Telnet preamble will NOT be sent. Common use of Telnet RAW is to test-open connections to web servers and mail servers to see if they are responding on their standard port 80 / port 25

The screenshot shows the 'CONNECT' screen for a new Telnet connection. At the top, there are navigation buttons: '< Back', 'Connection Details', and 'Save'. Below this is a header 'Connection Name' with the subtitle 'new Telnet Connection'. There are four tabs: 'SSH', 'Telnet' (which is selected and highlighted in blue), 'Serial', and 'Remote'. The main title 'CONNECT' is centered. Below it is a section titled 'TELNET SETTINGS'. It contains fields for 'Hostname', 'Port' (set to 23), and 'Username'. To the right of the 'Port' field is a toggle switch labeled 'As RAW:' which is currently turned off. Below the 'TELNET SETTINGS' section is a section titled 'TERMINAL SETTINGS'. It contains two rows: 'Columns' and 'Rows', both of which are set to 'Default'.

4. Serial

For making serial connections via any supported method.

- Redpark C2-RJ45 cable (30 pin Apple connector)
- Redpark L2-RJ45 cable (Lightning Apple connector)
- Airconsole Serial over Bluetooth Low Energy
- Airconsole Serial over WIFI

For connecting via Airconsole, usually enabling Bluetooth on your iOS device is sufficient or (alternatively) join the Airconsole-XX WIFI network. Get Console will then auto-detect and use Airconsole for Serial connections. For detailed instructions on connecting to serial ports via Airconsole see the Airconsole user manual.

For connecting to serial ports via Redpark cable, simply connect the cable to the iOS device first, and then to the serial device. Select “Cable” in the “Connect with:” field.

Regardless of the way the serial port is connected, the following pa-

The screenshot shows the 'CONNECT' screen for a new Serial connection. At the top, there are navigation buttons: '< Back', 'Connection Details', and 'Save'. Below this is a header 'Connection Name' with the subtitle 'new Serial Connection'. There are four tabs: 'SSH', 'Telnet', 'Serial' (which is selected and highlighted in blue), and 'Remote'. The main title 'CONNECT' is centered. Below it is a section titled 'SERIAL SETTINGS'. It contains several rows: 'Baud Rate' (set to 38400 Baud), 'Stop Bits' (set to 1 Stop Bit), 'Flow Control' (set to None), 'Parity' (set to None), 'Data Bits' (set to 8 Bits), and 'Connect with' (set to Cable). Each row has a right-pointing arrow indicating it can be expanded. Below the 'SERIAL SETTINGS' section is a section titled 'TERMINAL SETTINGS'. It contains one row: 'Columns', which is set to 'Default'.

Parameters can be configured on a per session basis, other serial settings are inherited from the Main Settings defaults:

Connection Name: Define a name for this connection

Serial Settings: The Baud rate, Stopbits, Flow Control, Parity and Databits can be set on a per connection basis. All other settings are globally configured in the Main Settings page.

Terminal Settings: As per all other connections the terminal settings defines a non-default Terminal characteristics for just this connection.

Keyboard Settings: As per all other connections, the Keyboard settings can override the default Keyboard settings on a per connection basis.

4.2.1.5. Remote

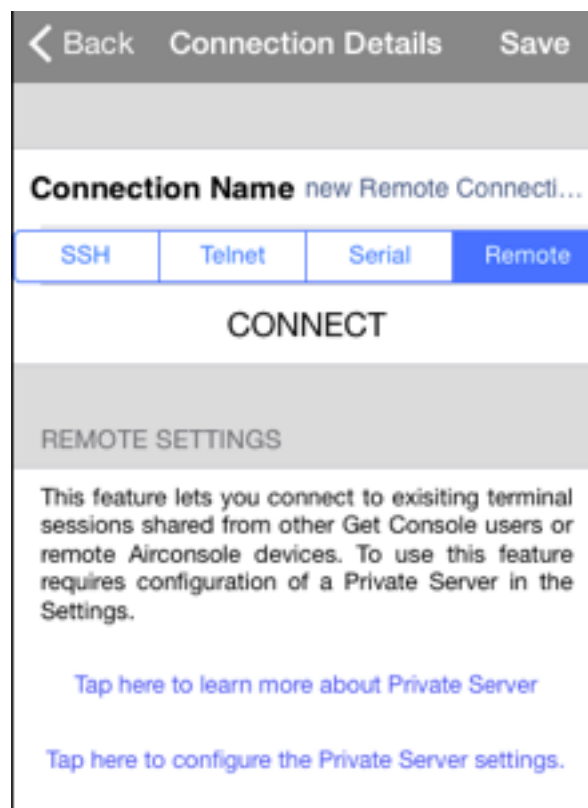
For this feature to be used the purchase of the Get Console Private Server is required. It allows you to remotely connect to shared existing terminal sessions created by other users or directly to *remote* Airconsole devices.

Connection Name: Define a name for this connection.

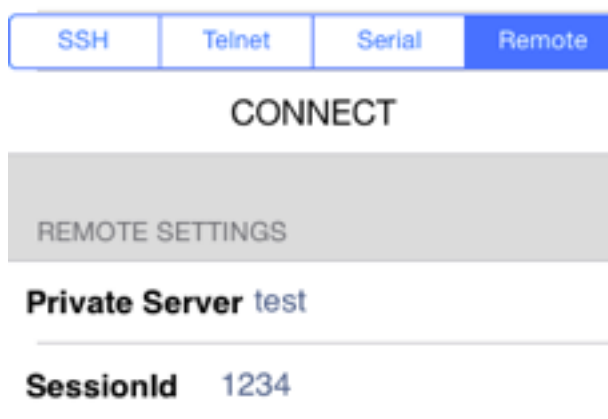
Learn about Private Server: This link will send you too the private server page on get-console.com where you can learn about the features of a Get Console Private Server and purchase one if you are interested.

Private Server Settings: This Link will just send you to the the Sharing Settings within the main Settings menu where you can connect to your Private Server.

SessionId: This is a token code that is required to gain access to a remote session. Generally this is a random code for remote Get Console sessions, however for Airconsole units it can be persistent and therefore it is useful to save it in a saved connection for future use



Connection Name Example



2. Managing Connections in Folders

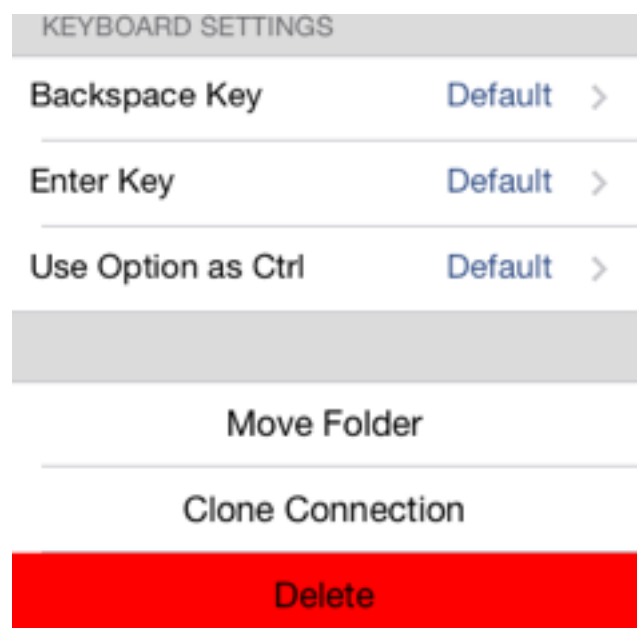
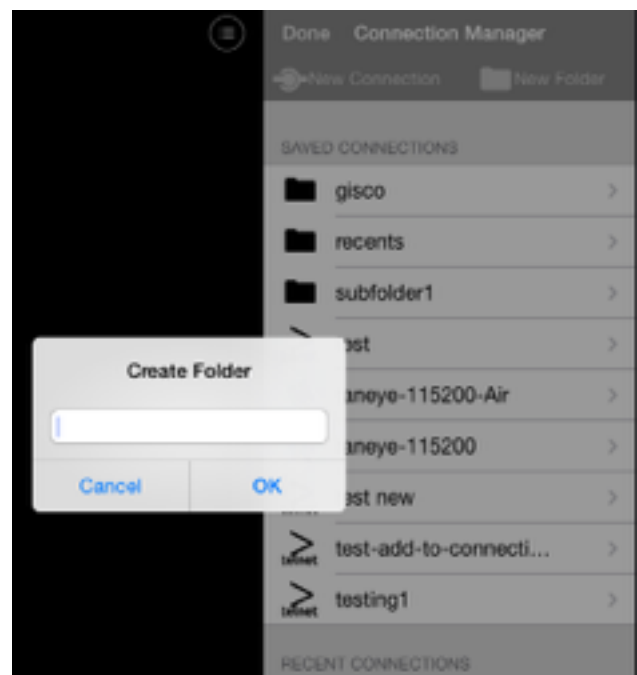
Connection Manager allows for the creation of connections, grouping of connections into subfolders and also for cloning connections and moving them between subfolders. These folders are stored in the Get Console Application's file system area within the iPhone/iPad. They can be exported via iTunes, however the easiest method to extract or import new connections is to use Dropbox connectivity. The internal /connections folder is automatically synced with Dropbox so can be modified on remote Mac or PC with the changes pushed to the Get Console app immediately.

1. *Creating and Managing folders and sub-folders for Connections*

Tap "New Folder" to create a new folder and enter name in dialog box. The folder appears at the top of the Saved Connections List. Folders can be nested, so to create a subfolder navigate to the new folder by tapping it then tap the New Folder button again.

Saved connections can be moved from one folder to another or to the root. To move a connection tap the blue arrow next to a saved connection, scroll to the bottom where there are options to Move (to a) Folder, as well as Clone the Connection or Delete.

A sub-folder can also be created under another folder with connections in it. Note that if you delete a folder which contains a connection, the connection will be moved to the root folder instead of being deleted together with the folder.

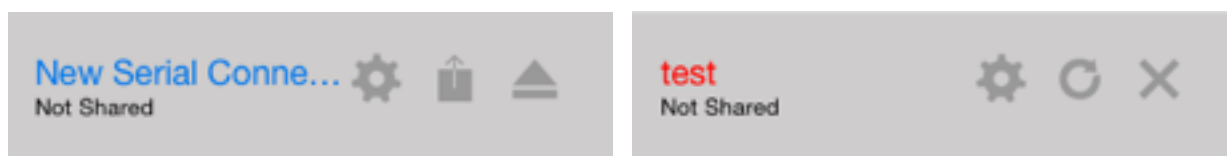


2. Recent Connections






Get Console Connection Manager presents the 10 most recent connections below the Saved Connections. The connections listed in Recent connections include Quick Connections and Saved Connections. This list can be cleared by tapping the “Clear Recent Connections” button at the very bottom of the list. Alternatively, a recent connection that was made via “Quick Connect” method can be saved to the Connection Manager via tapping the blue arrow, providing a name (description) for the connection and then tapping save. The connection will then appear in the root Saved Connections folder of the Connection Manager.

3. In Session Options

Once a connection has been established (or attempted to be established) it will appear in the Session Manager. From this popup the actions that can be performed depends on whether the connection is live (connected - blue) or closed (disconnected - red)



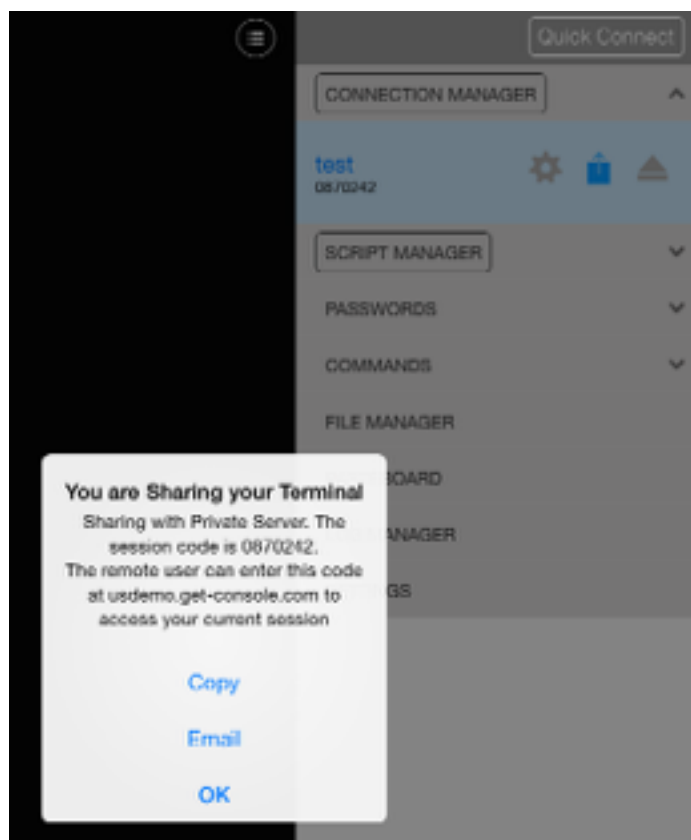
The following table describes the actions available.

	Session Options: Allows for editing of changeable settings while the session is live, for example changing the terminal width, scrollbar, colour scheme etc)
	Share Session: Will share the terminal screen with the configured remote server (Public or Private), and generate a one-time token code. See the session sharing Settings (section 3.6 above) and how to operation below.
	Disconnect Session: Disconnect from the remote server (Telnet/SSH/Raw) or disconnect the serial cable (Serial). The session will stay visible in session manager list of sessions with the 2 below options available.
	Restart Disconnected Session: Attempts to reconnect the session to the remote server (Telnet/SSH/Raw) or cable (Serial).
	Remove Disconnected Session from Session Manager

4.4. Session Sharing

Tapping the share button will attempt to share the selected terminal session with the Get Console website or configured Private Server so that a remote web user can see and interact with the terminal session at the same time. In order to use this feature the settings for session sharing must be configured in the main settings page (see section 3.6 above).

Each shared terminal session is dynamically given a one-time token code at the time session sharing is initiated. This code is



used to secure access to the users iPad/iPhone. The remote web user must know this 7-digit token code in order to see the iPad/iPhones terminal window.

Only sessions that are shared are visible to remote users, however multiple concurrent sessions can be shared – each with their own unique one-time token code.

1. Start Session Sharing

Once the notification “You are Sharing your Terminal” appears, the remote user can access the iPad via the token code displayed in the notification. To make it easy for the remote user to learn the code it can be emailed or copied to clipboard via this notification window.

Options	Function
- OK	Return to the active terminal window
- Copy	Copy the remote access token code to the clipboard so that it can be used in txt message or other iOS device message
- Email	Email the remote access token code to a recipient with instructions how to connect to the shared session

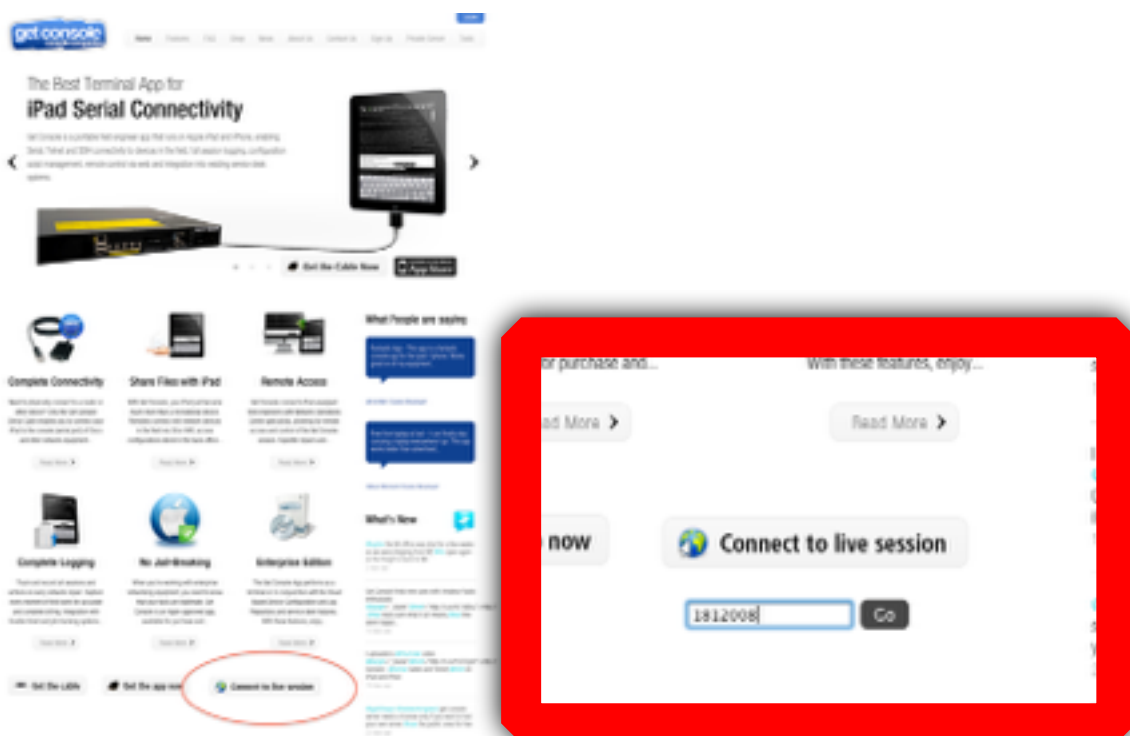
Once shared, the token code will also appear in the Session Manager, and the share icon will show blue.



2. Remote User Access

Remote access to shared terminal sessions is dependent on whether they have been shared with the Get-Console.com public servers or via the users own privately hosted server (Private Server). For the purpose of this manual it is assumed shared with the Get-Console.com public servers. To learn more about Get Console Private Server visit www.get-console.com/private-server

When the remote user has obtained the one-time token code, they enter it on the home page at www.get-console.com, or if the token code was shared with the remote user via email there is a single link to click in the email to directly connect.




After entering the token code, the code is checked against currently active and available codes, and if correct the Get Console web terminal will launch. The remote user is presented with a plugin free web terminal that mirrors what is on the screen of the Get Console app running on the iPad/iPhone.



Both iPad/iPhone user and remote user can interact with the terminal session at the same time, however only one remote user can access any given session. If a later remote user enters the same token code, their later session will take over from any existing remote user.

Note that the remote user can only interact with the terminal while the Get Console app is foreground on the iPad/iPhone. If Get Console is in the background, then it will still maintain its Serial/Telnet/SSH session for upto 10 minutes, however the remote user will not be able to interact with the session until Get Console is brought to the iPad/iPhone foreground again.

3. Stop Sharing Session

To stop sharing an iPad/iPhone terminal session with a remote user, tap the connection manager button to activate the Session Manager popover, then tap the  button for the terminal tab that sharing should be disconnected. The world icon will turn grey and the subtitle will change from the token code to say "Not Shared".

5. Terminal Features

This section discusses the following Get Console terminal features:

- Keyboard popup bars

- Command Shortcuts
- Password Shortcuts
- Clipboard Viewer

1. Terminal Features

1. Keyboard Control



Use the button to hide the onscreen iPad keyboard and double tap the terminal to reveal it again. Used to increase screen real estate, especially useful for debugging in the iPhone/iPod with its smaller screen. Note that when a Bluetooth keyboard is connected then there is no software keyboard on the screen.

2. Keyboard Popup Bar Selector

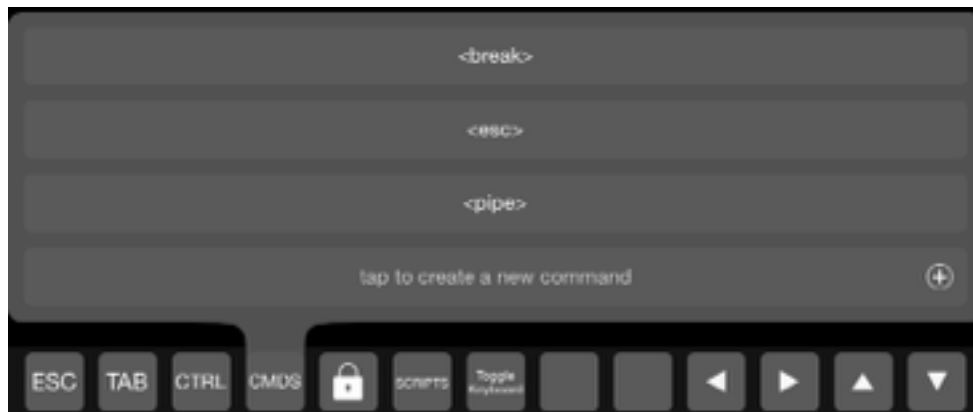
Options:

- **F1-F12 – Function Keys**
- **Custom – user selectable keys**

For the Default option, the user can create their own combination of buttons on the top bar by pressing and holding each individual button. A button selection popup will allow these keys to be changed to one of many options. In version 1.81 or later any key from the Apple keyboard can be installed on the Custom keyboard bar, in addition to the ones available in the popup picker. Using the CTRL key on the picker prior to selecting a Apple keyboard key will put CTRL-[selected key] instead of the normal key.



3. Command Manager



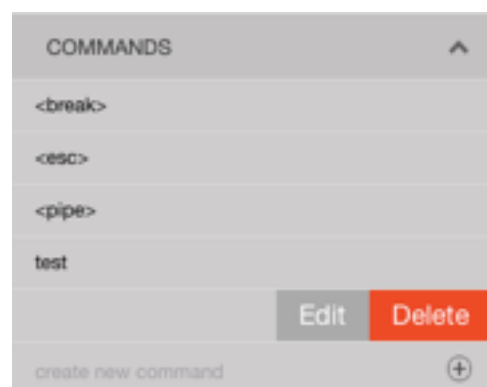
Command shortcuts are added in the Command Manager. Once entered they are available with one tap via the “Cmds” popup from the top keyboard bar. Each command is sent with a carriage return after it.

Tapping CMDS will show the list of current command shortcuts. A few shortcuts have already been defined by default and user

Defaults:
- Command Manager (Launches Command Manager page for entering more quick commands)
- <break> - sends terminal break sequence
- <esc> - sends ESC sequence for selected encoding method
- <pipe> - sends the symbol WITHOUT a Carriage Return

can add their own custom command shortcuts for use in the terminal by simply typing in the new command then tapping the “+” button. Adds command shortcuts just below your current shortcuts.

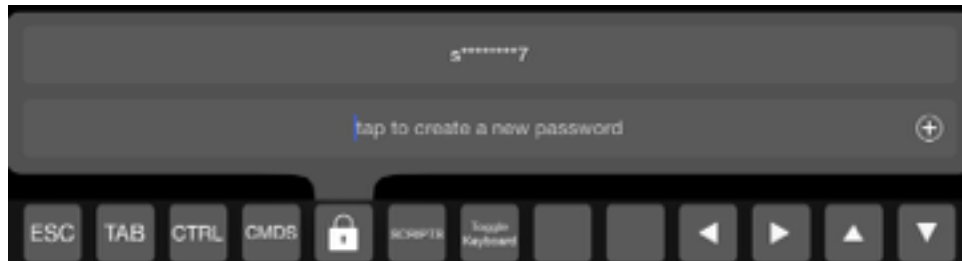
Swipe to left on Command Side menu: Changes the right side of the row to edit and delete buttons which then can delete the saved command shortcut.




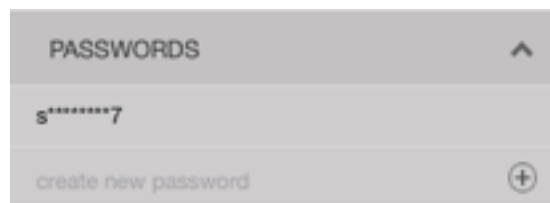
4. Password Manager

Like the Command Manager, Allows user to save passwords to be used as shortcut instead of typing the entire password all over again

Add passwords here for quick use in terminal via keyboard icon



Saved passwords are recalled in terminal sessions by tapping the  button. While the passwords middle characters are obfuscated when recalled via the terminal or the password menu to ensure

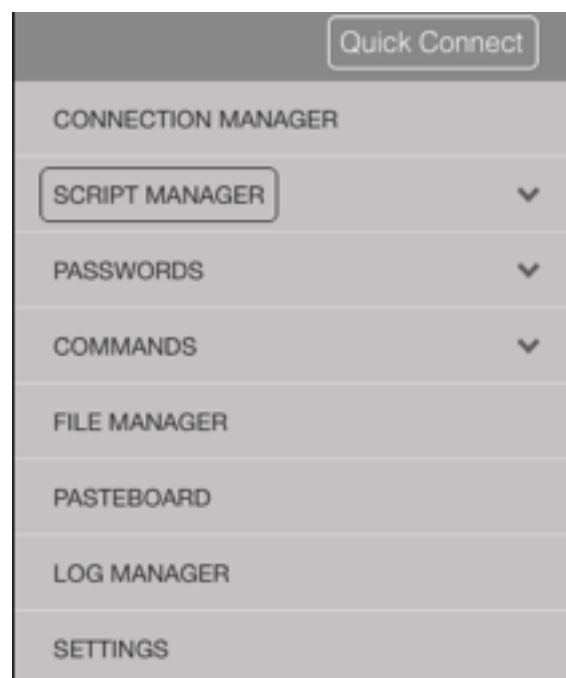


security of sensitive information.

5. Clipboard Viewer

The iPad version of Get Console has a full screen clipboard window accessed via the Pasteboard button on the side menu. This allows the user to see and edit the contents of the iPad's clipboard and then paste.

The iPhone version of Get Console has a whole page clipboard viewer with the same buttons for pasting to the terminal window.



1. *Paste Clipboard to Console*

Pastes the content of the clipboard to the console. If the user has made changes to the clipboard viewer text, but has not selected and copied the clipboard viewer to the internal clipboard, then this button will paste the clipboard viewer contents *prior* to any edits.

2. *Paste Editor to Console*

Pastes whatever is in the editor (the box containing the content of the clipboard) to the console. This allows the user to modify the content of the clipboard first before pasting to the console. Effectively what this button does is copy the edited view to the clipboard prior to then pasting to the terminal window.

Once the editor is closed, the content of the clipboard will revert to the original clipboard again unless the edited “Editor” is copied which then replaces the clipboard content.

3. *Launch Scratchpads*

Get Console v1.82 and above has 3 additional “scratchpads” that can be used by user for loading in text files for editing and pasting to the active terminal session.



Selecting an unused Scratchpad can be used to either create new text or load a file into for editing. Selecting cancel returns to the active terminal window.

6. Get Console File System

The File Manager can be accessed directly from the side menu.

1. Get Console File Types

Get Console stores files within a portion of the iPad/iPhones file system that is dedicated to the Get Console App. There are 5 types of files that are stored:

- User created text based files that are typically used in a terminal session via cutting/pasting into the terminal window via the Clipboard Viewer.
- Log files, which are Get Console generated logs for each terminal session that has logging enabled.
- RSA Keys for use in certificate based authentication in SSH. These files are imported via the Main Settings -> Private Key dialog box.
- “.connection” files which are saved connections visible in the Connection Manager. These files are created either in the Get Console Connection Manager, imported by Get Console File Manager, or imported via iTunes.
- “.script” files which are saved scripts that can be executed on login or during a terminal session via the Script Manager. These files are created either in the Get Console Script Manager, imported by Get Console File Manager or imported via iTunes

2. User Created Text Files

While there are 5 types of files, the File Manager is concerned only with

- a) the text files type - User Created text based files.
- b) .connection files - saved connections that can be imported into Connection Manager
- c) .script files - saved terminal scripts that can be imported into the Script Manager

Other types of files are created / viewed / deleted / uploaded via their respective managers as discussed below.

User Created text files are typically used in terminal sessions – they could be snippets of configurations or other text based templates that are edited in the clipboard viewer or scratch pads before being pasted into the Terminal window.

To be used by Get Console, the files must be local to the iPad/iPhone device. Get Console offers 2 ways to get User Created text files into the local file system:

- via import from the users portal on the www.get-console.com website
- via import from the users /My Apps/Get Console/ folder in their Dropbox.com cloud storage

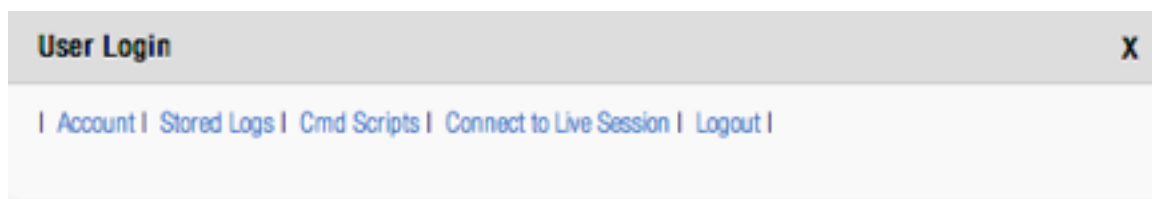
1. Importing files from Get-Console.com

To import text files from the Get-Console.com portal the user must first have signed up to a free account at www.get-console.com/signup. After sign up, the user logs in via the Login panel at the top of any webpage.

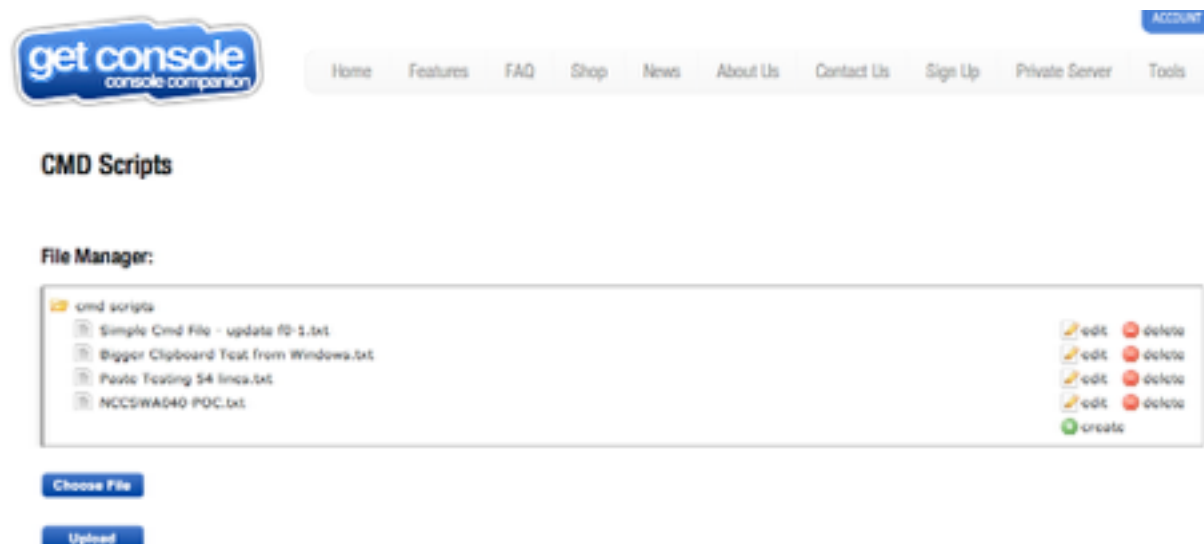
1. Creating / Uploading files on Get Console Website

A screenshot of the 'User Login' form. It has a title bar with 'User Login' and a close button 'X'. Below the title bar, there are two input fields: 'Username' with the value 'simon@ipguys.com' and 'Password' with masked characters '.....'. To the right of the password field is a 'Login' button. Below the input fields, there are two links: 'New User? Sign up' and 'Forgot your password?'.

Once signed in the website “Login” button changes to an “Account” button. Clicking on this reveals the users portal options:

A screenshot of the user portal options. It has a title bar with 'User Login' and a close button 'X'. Below the title bar, there is a horizontal menu with the following options: 'Account', 'Stored Logs', 'Cmd Scripts', 'Connect to Live Session', and 'Logout'.

User Created Text files (including .script and .connection files) are accessed via the “Cmd Scripts” button. This button will be renamed “User Created Files” in a later release of the website.

A screenshot of the 'CMD Scripts' user portal. At the top, there is a navigation bar with the 'get console' logo and a list of links: Home, Features, FAQ, Shop, News, About Us, Contact Us, Sign Up, Private Server, and Tools. Below the navigation bar, there is a section titled 'CMD Scripts'. Under this section, there is a 'File Manager:' section. The file manager shows a list of files: 'cmd scripts', 'Simple Cmd File - update RD-5.txt', 'Bigger Clipboard Text from Windows.txt', 'Paste Testing 54 lines.txt', and 'NCCSWAD40 POC.txt'. Each file has an 'edit' icon (pencil) and a 'delete' icon (trash). At the bottom of the file manager, there is a 'Choose File' button and an 'Upload' button.

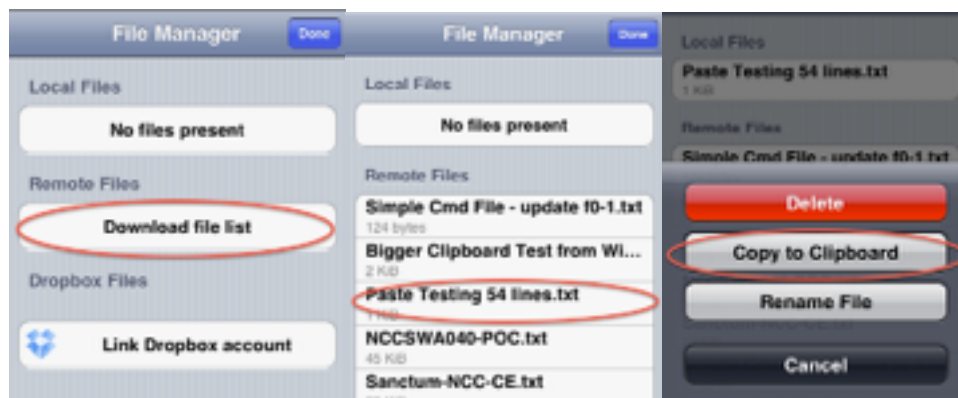
From the CMD Scripts user portal page, the user can upload, edit or create text files that will be visible to the iPad/iPhone Get Console application under the

2. Downloading Files from Get Console Website

Once files are uploaded to the users Get-Console.com portal page, they are visible via the File Manager by tapping the “Download File List” button. The users get-console.com credentials (email / pass-

word) must be entered into the Main Settings -> Session Sharing field as described in section 3.6 above in order for the remote files to be visible.

The list of files is presented to the user. Tapping on one of the remote files provides option to download to the local store. Once in the local store tapping the file name again allows for the file to be copied to the clipboard viewer for editing/pasting, renaming or deleting. If the file tapped in the local store is a .script or .connection file, the Get Console pop-up will allow these files to be imported into the Script or Connection manager respectively. However we do recommend a Dropbox account as apposed to remote files it allows 2 way syncing between the local files and those on the dropbox.



2. Importing files from Dropbox account

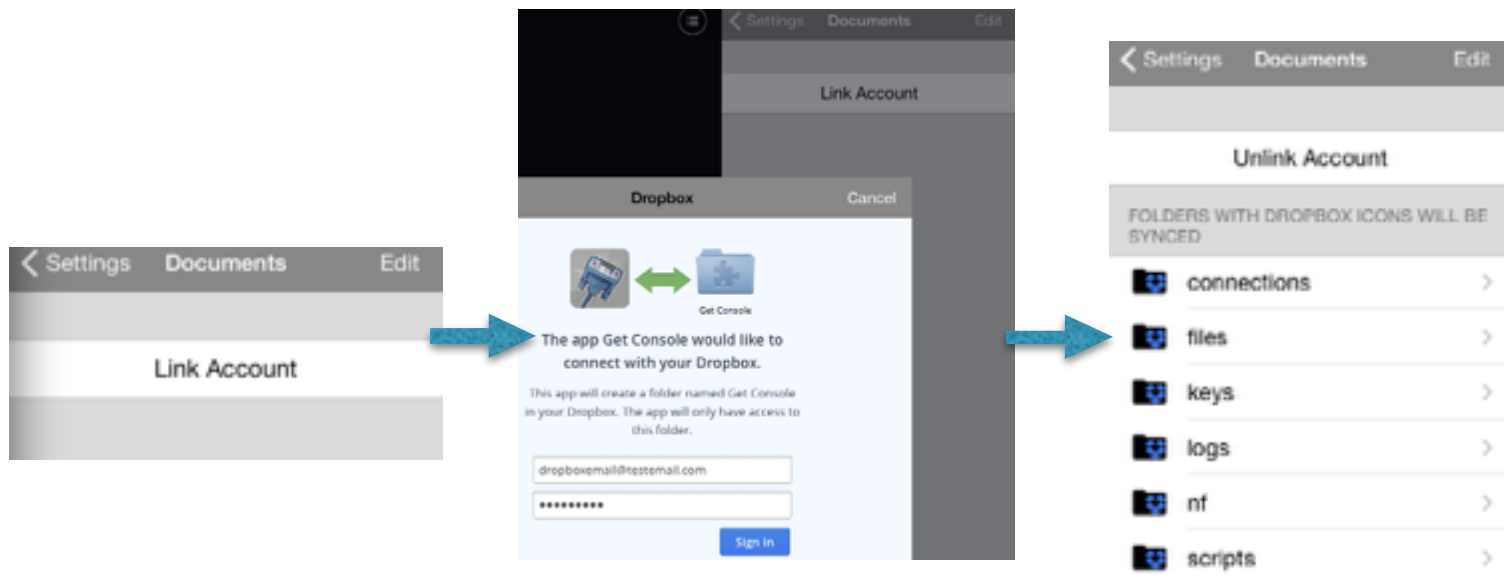
To import text files from a dedicated Get Console folder within the users Dropbox.com account, the user must first link their Dropbox account with the Get Console app. After successfully linking, the File Manager will be able to see User Created text files stored in the users Dropbox/Apps/GetConsole folder, and download them to the iPad/iPhones local storage.

Once in local storage the same operations are permitted as described for Get Console portal files are possible (Copy to Clipboard, Rename or Delete).

1. *Linking Dropbox Account to Get Console*

From version 1.82, linking to Dropbox is performed from the Main application settings. In the main settings, tap the Link to Sync Settings button under Cloud Storage Settings. This will start the mobile safari browser where the user will enter their dropbox.com credentials to login, or alternatively, if they already have the Dropbox iOS application installed on their iPad/iPhone it will present a simple allow/disallow dialog box. Tap Allow, after which control will return to Get Console. From Get Console File Manager, it will now be possible to download and Dropbox folder contents to the local storage area and visa versa

Note that once linked, the user can unlink Get Console from their Dropbox account either within the Main settings or via the Dropbox website. Only a single folder is linked to the users Dropbox repository (not including any subfolders) therefore any files the user wants visible to the File Manager must be in this folder to be accessible.



3. Log Files

Get Console offers comprehensive logging of individual terminal sessions. Once logging is enabled in the main settings, all printable screen output is captured to a logfile.

One log file is created for each session. Unless a script is being used to append a log file, a new log file is created each time a session is stopped and restarted.

1. Log File Naming

For connections that are started with the “Quick Connect” method the log file name will be:

Log_YYYY-MM-DD_HHMMSS.txt

For connections that are started from a saved connection in the Connection Manager, the file name will instead be appended with the connection name followed by date: ie

Server1_YYYY-MM-DD_HHMMSS.txt

Invoking log operations from a script can change the name of the log files. See the Script Manager section above for details on log operations that are possible via scripting.

2. Uploading Log Files

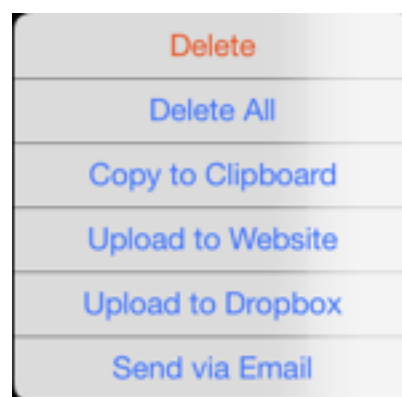
Log files can be extracted from the iOS device via 2 possible methods:

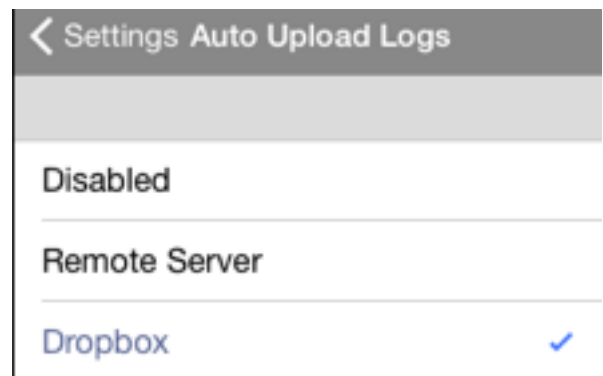
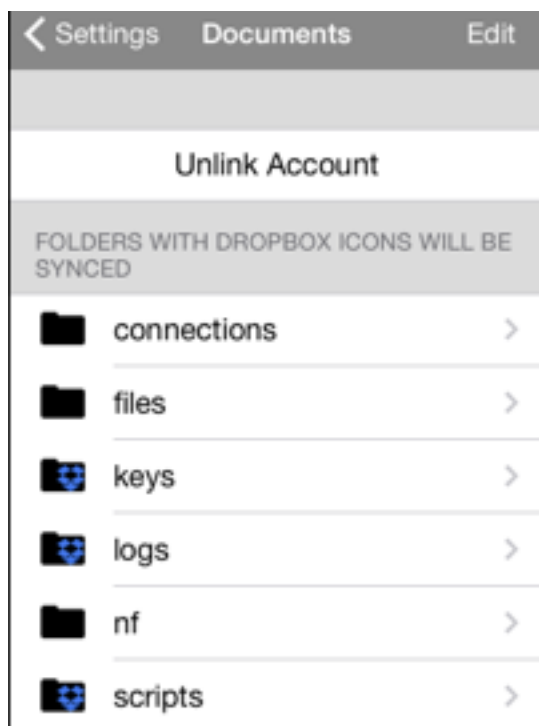
- Via upload to the users portal page on the get-console.com website
- Via upload to the users Dropbox folder dedicated to Get Console (default is Apps/Get Console)
- Via email of individual log files from the iPad/iPhone mail client

Navigate to the Log Files page, then tap a log file name. Select the one of the 3 upload options. Note to upload to Dropbox, the users Dropbox account must be linked to the Get Console app as described in section 6.2.2.1 above.

Upload to Get Console website user portal is invoked via the “Upload to Website” button. As per the File Manager, the user must have a (free) Get-Console.com account and the credentials entered as described in section 3.6 above.

Access to uploaded logs on the Get-Console.com website is via the Stored Logs button.





3. Auto-Uploading Log files

As discussed at section 3.1.8 Get Console can be configured to automatically synch log files on session completion to either Dropbox or to the users portal page on the Get Console website. To enable Auto-Upload, select the appropriate option from the Synch Settings.

In this image the keys, scripts and logs are all synching to drop box while the others are not. This can be changed by tapping the edit button where you can select which icons you do not wish to synch.

Note that when Auto-Upload logs is enabled, the iPad / iPhone must have Internet connectivity via 3G or Wifi at the completion of the terminal session. If there is no Internet connectivity, then the auto-upload will fail, and Get Console will NOT re-attempt to auto upload that log file. The log file will still be saved locally and available for manual upload.

4. .Script and .Connection Get Console Files

Get Console stores .connection and .script files for saved connections and saved terminal scripts respectively. These are created and managed in the Connection Manager (see section 4.2) or Script Manager (see section 3.2). These are simple XML files and can be manipulated externally from the Get Console application and then re uploaded.

Existing .connection and .script files can be extracted via Dropbox syncing to a folder, or alternatively via iTunes. Dropbox connectivity is covered in the File Manager section above.

To extract via iTunes, connect iPhone/iPad to PC/Mac, run iTunes, navigate to the iPad/iPhone device and then select Apps.

File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

Apps

	Evernote
	Get Console
	Get SatCom
	Google Earth

Get Console Documents

	connections	20/02/12 3:57 PM	24 KB
	scripts	17/02/12 2:34 PM	8 KB

The .connection and .script files can be extracted via the “save to...” button at the bottom of the page.

Individual .connection and .script files that have been created or edited externally can be uploaded via the “Add...” button. When Get Console starts it checks the root folder for .script and .connection files and moves these into the correct internal folder.

From version 1.82 and later, .connection and .script files are detected via the local file system and when tapped in the file system viewer are optionally installed into Script Manager or Connection Manager automatically. It may be necessary to stop and restart the Get Console application on the iPad/iPhone to trigger the auto-installation of connections.zip files.

7. General Troubleshooting

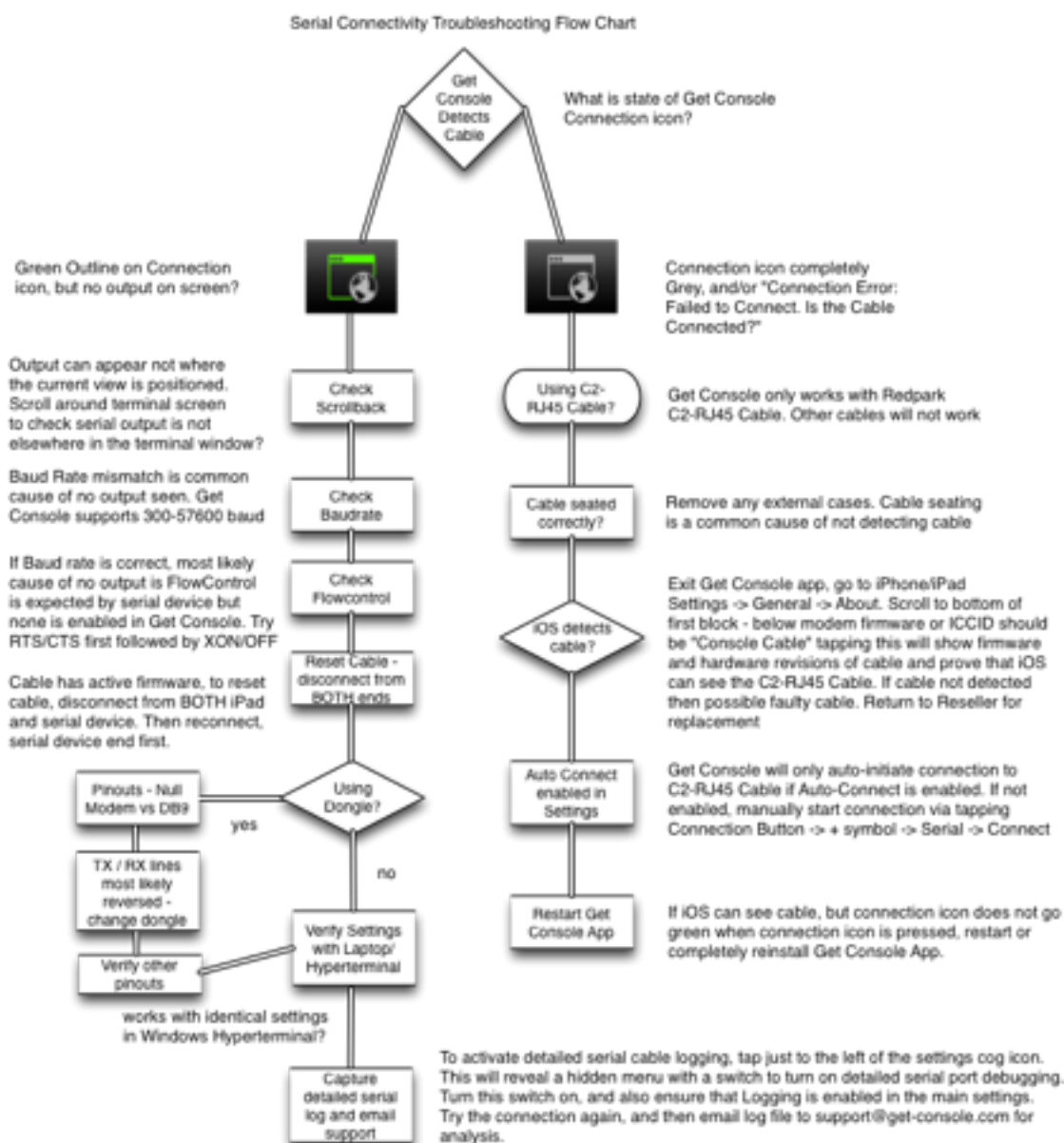
This section covers common problems reported by users, and typical fixes / work arounds

1. Serial Connectivity Issues

Serial connectivity issues can be divided into 2 types:

- Where Get Console cannot recognize / communicate with Redpark C2-RJ45 cable
- Where Get Console can recognize Redpark C2-RJ45 cable, but no console output is seen on the screen

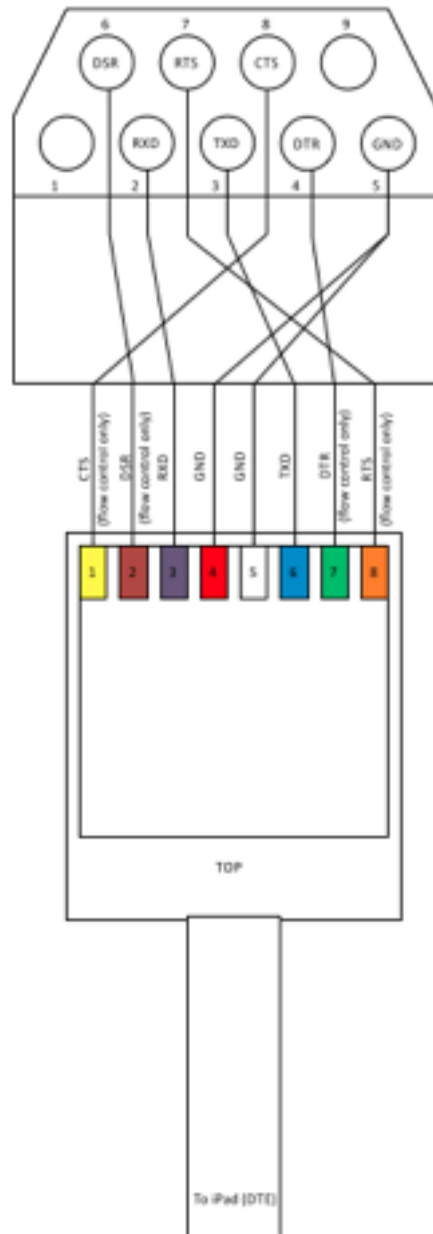
The following flowchart helps answer the most common reasons why no output can be seen on screen when serial cable is connected to a device. Note that why Get Console is generally compatible with many serial devices, Cloudstore will only support / troubleshoot connections to devices that we



have officially tested with. The most recent list of Cloudtore supported serial devices for use with Get Console is contained in the support forum at <http://www.get-console.com/forum/>

1. C2-RJ45 Cable Pinouts

The following drawing show the standard C2-RJ45 cable pins and what serial signal they send, and also the corresponding pins when the RJ45-to-DB9 adaptor is used (as purchased from the www.get-console.com/shop)



2. Console Cable Not Detected

If this error is seen and there is a cable plugged into the iOS device, then check that the iOS device is detecting the cable by going to the iOS devices general settings -> about -> console cable. This will show the cable hardware and firmware detected. Get Console only works with the Redpark C2-RJ45 cable. If you have a C2-RJ45 cable connected but it is not detected by the apple iOS operating system, then check it is sea-

ted correctly, reinstall Get Console and if still having issues contact the cable vendor for a replacement.


3. Console Cable Detected, No Output on Screen

If the cable is detected (Green outline on Connection icon for Serial session), but no output appears on the screen the most common reasons are:

- Baud Rate mismatch - note that Get Console does not support 115200 baud on current Redpark C2-RJ45 cables
- Flow control is required but not set. Note disconnect cable from both iPad/iPhone and serial device after changing Flow Control to reset cable.
- Pinouts of device are reversed from what is expected by C2-RJ45 cable. This is very common. The pinouts of the C2-RJ45 cable are as per Cisco console port. Using a RJ45-to-DB9 adaptor either makes the DB9 interface DTE or DCE depending on whether it reverses TX/RX and flow control/signalling pins. Use adaptors available from the get-console.com/shop which have been specifically wired to convert the C2-RJ45 to DB9 (std) and DB9 (null modem) including all the control/signaling.
- Cable needs to be reset - the cable has active electronics in it. Occasionally it needs to be reset by removing BOTH ends of the cable from iPad and Serial device, and then reconnecting it first to the Serial device end

2. Detailed Serial Troubleshooting

Get Console has a “hidden” diagnostics page that can be accessed via holding down the Side bar icon at the top right of the screen. This page is used primarily by Get Console developers to detect the active state of the serial cable pins, and to enable detailed serial cable debugging. Enabling detailed serial cable debugging will write hex values for every frame sent to and received from the C2-RJ45 cable, so if this mode is enabled it replaces normal log files of readable ASCII text from the remote serial device.

Console Status				Done
Connected	0:44			
Baud Rate	N/A	Baud rate negotiated with C2-RJ45 cable - does NOT mean end device is operating at this baud rate		
RX Bytes	109	Counters for Received frames for this session		
TX Bytes	2	Counters for Transmitted frames for this session		
Remote PIN	N/A			
Serial Debug Log		Off by default - enable only when instructed by Get Console Support		
Modem Status	CTS DSR CD RI	State of Serial Flow Control/ Modem pins. Only useful where flow control is enabled. By default Get Console will set RTS and DTR high when hardware flow control is enabled		
	RTS DTR			
Flow Status	-	Flow status - options = OK (Get Console will transmit frames to serial port), or waiting on remote side to raise DSR, CTS or both(TX will not)		

3. Session Sharing Issues

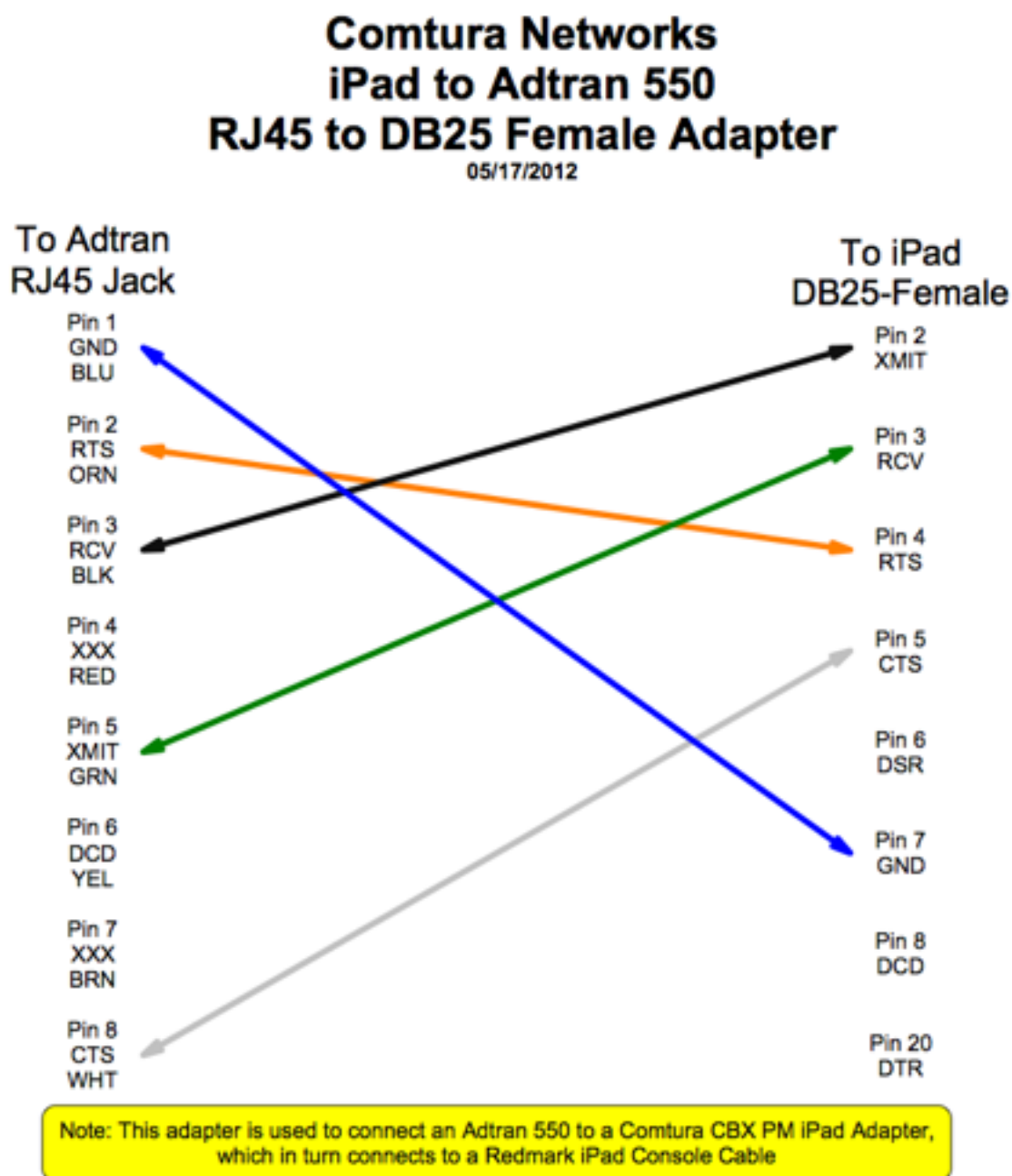
Session Sharing to Get Console Public Servers fails. If user receives “Failed to connect / read from remote control server”:

- Check your internet connection (can you browse in mobile safari to www.get-console.com)
- Check Session Sharing settings configurations defined in Main Settings -> Session Sharing are correct (see section 3.6).
- If everything is correctly configured, try again - depending on latency the first attempt to connect can fail due to underlying encryption key exchange taking too long. The next attempt works as the encryption keys are cached by the Get Console app.
- If still having issues with a public Get Console server, try to use an alternative one (ie Asia Pacific rather than North America) .
- If having problems with a Private Server contact the Server Administrator, or if support has been purchased, contact Cloudstore.

8. Appendix A - Specific Device Serial Port Pinouts

This section includes some pinouts for non-Cisco devices that have successfully been tested with Get Console by other customers.

1. ADTRAN 550



2. CBX / PhoneMail - DB9 and DB25 Adaptors



iPad to CBX / PhoneMail RJ45 to DB9 Male Adapter

05/17/2012

To iPad
RJ45 Jack

Pin 1
CTS
BLU

Pin 2
DSR
ORN

Pin 3
RCV
BLK

Pin 4
GND
RED

Pin 5
GND
GRN

Pin 6
XMIT
YEL

Pin 7
DTR
BRN

Pin 8
RTS
WHT

To Equipment
DB9-Male

Pin 1
DCD

Pin 2
RCV

Pin 3
XMIT

Pin 4
DTR

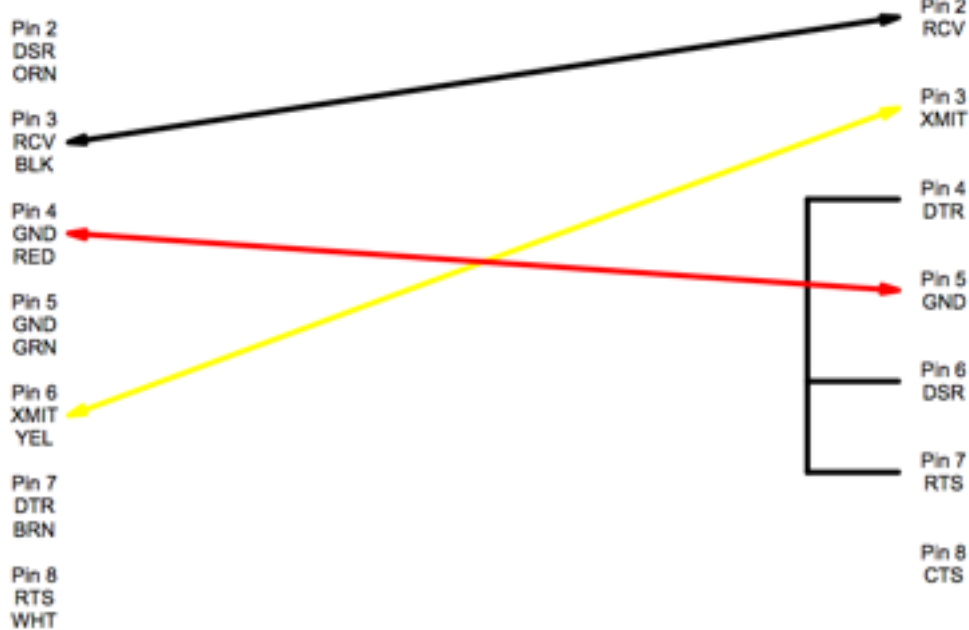
Pin 5
GND

Pin 6
DSR

Pin 7
RTS

Pin 8
CTS

Pin 9
RI



Note: This adapter is designed to be used with the Redpark iPad Console Cable and Get Console App



iPad to CBX / PhoneMail RJ45 to DB25 Male Adapter

05/17/2012

To iPad
RJ45 Jack

Pin 1
CTS
BLU

Pin 2
DSR
ORN

Pin 3
RCV
BLK

Pin 4
GND
RED

Pin 5
GND
GRN

Pin 6
XMIT
YEL

Pin 7
DTR
BRN

Pin 8
RTS
WHT

To Equipment
DB25-Male

Pin 2
XMIT

Pin 3
RCV

Pin 4
RTS

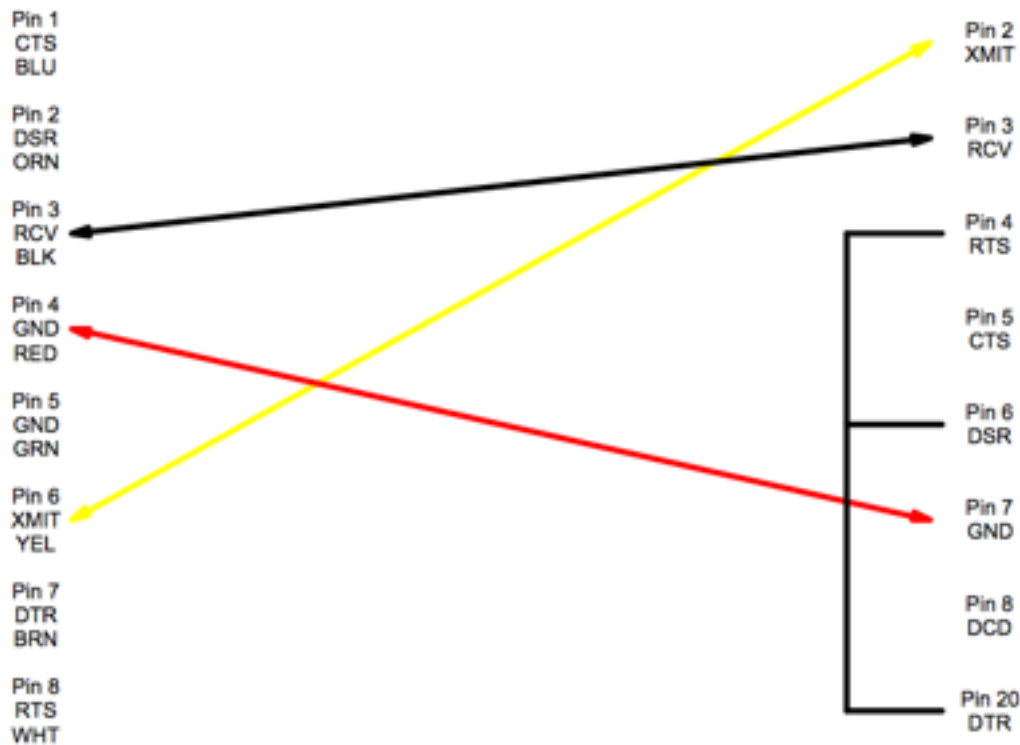
Pin 5
CTS

Pin 6
DSR

Pin 7
GND

Pin 8
DCD

Pin 20
DTR



Note: This adapter is designed to be used with the Redpark iPad Console Cable and Get Console App