DeviceMaster Security

This document is designed with the intent of providing a basic understanding of the DeviceMaster Security options, the repercussions of setting these options and the manner in which these options may be returned to default.

Additional information may be found in other Comtrol documents such as the DeviceMaster User Guide. The information in this document is a compilation of information from these other sources and may not be as complete as the original documents might be. This guide is meant as a overview of the subject.

This guide will not discuss the creation of CA Certificates as that is much too complex for this guide.

Understanding Security methods

(or, quick descriptions for the basic understanding of security)

(return to top)

Secure Data mode

- TCP connections which carry data to/from the DeviceMaster serial ports will be encrypted using **SSL** or **TLS** security protocols.

Secure Config mode

- Unencrypted access to administrative and diagnostic functions are disabled

Client Authentication

- A process using paired keys and identity certificates to prevent unauthorized access to the DeviceMaster.

Secure Monitor mode

- Allows monitoring of a single serial port on the DeviceMaster while the port is configured for secure data mode

SSL (Secure Sockets Layer)

- Secure Sockets Layer, predecessor of (TLS) Transport Layer Security
    - SSL is a commonly-used protocol for managing the security of a message transmission on the Internet. SSL has recently been succeeded by Transport Layer Security (TLS), which is based on SSL. SSL uses a program layer located between the Internet's Hypertext Transfer Protocol (HTTP) and Transport Control Protocol (TCP) layers. SSL is included as part of both the Microsoft and Netscape browsers and most Web server products. Developed by Netscape, SSL also gained the support of Microsoft and other Internet client/server developers as well and became the de facto standard until evolving into Transport Layer Security
    - SSL uses the public-and-private key encryption system from RSA, which also includes the use of a digital certificate.
- Two slightly different SSL protocols are supported by DeviceMaster: SSLv3 and TLSv1.

TLS (Transport Layer Security)

- TLS is a protocol that ensures privacy between communicating applications and their users on the Internet. When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any message. TLS is the successor to the Secure Sockets Layer (SSL).
- TLS and SSL are not interoperable. The TLS protocol does contain a mechanism that allows TLS implementation to back down to SSL 3.0.

SSH (Secure Shell)

- Secure Shell allows data to be exchanged using a secure channel between two networked devices
- Replaces telnet which has no security
- Requires password authentication – even if password is empty

PKI (public key infrastructure)

- A PKI (public key infrastructure) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The 'public key' infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates. Although the components of a PKI are generally understood, a number of different vendor approaches and services are emerging. Meanwhile, an Internet standard for PKI is being worked on.
- The public key infrastructure assumes the use of *public key cryptography*, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet. (The private key system is sometimes known as *symmetric cryptography* and the public key system as *asymmetric cryptography*.)
- A public key infrastructure consists of:

- A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key
- A registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor
- One or more directories where the certificates (with their public keys) are held
- A certificate management system

- **How Public and Private Key Cryptography Works**
  - In public key cryptography, a public and private key are created simultaneously using the same algorithm (a popular one is known as RSA) by a certificate authority (CA). The private key is given only to the requesting party and the public key is made publicly available (as part of a digital certificate) in a directory that all parties can access. The private key is never shared with anyone or sent across the Internet. You use the private key to decrypt text that has been encrypted with your public key by someone else (who can find out what your public key is from a public directory). Thus, if I send you a message, I can find out your public key (but not your private key) from a central administrator and encrypt a message to you using your public key. When you receive it, you decrypt it with your private key. In addition to encrypting messages (which ensures privacy), you can authenticate yourself to me (so I know that it is really you who sent the message) by using your private key to encrypt a digital certificate. When I receive it, I can use your public key to decrypt it.

- **Who Provides the Infrastructure**
  - A number of products are offered that enable a company or group of companies to implement a PKI. The acceleration of e-commerce and business-to-business commerce over the Internet has increased the demand for PKI solutions. Related ideas are the virtual private network (VPN) and the IP Security (IPsec) standard. Among PKI leaders are:
    - RSA, which has developed the main algorithms used by PKI vendors
    - Verisign, which acts as a certificate authority and sells software that allows a company to create its own certificate authorities
    - GTE CyberTrust, which provides a PKI implementation methodology and consultation service that it plans to vend to other companies for a fixed price
    - Xcert, whose Web Sentry product that checks the revocation status of certificates on a server, using the Online Certificate Status Protocol (OCSP)
    - Netscape, whose Directory Server product is said to support 50 million objects and process 5,000 queries a second; Secure E-Commerce, which allows a company or extranet manager to manage digital certificates; and Meta-Directory, which can connect all corporate directories into a single directory for security management

RSA Key pair

- This is an algorithm for public-key cryptography. It is the first algorithm known to be suitable for signing as well as encryption. RSA is widely used in electronic commerce protocols, and is believed to be sufficiently secure given sufficiently long keys and the use of up-to-date implementations. The system includes a communications channel coupled to at least one terminal having an encoding device, and to at least one terminal having a decoding device.

- **Public key**
  - The public key is a value provided by some designated authority as an encryption key that, combined with a private key derived from the public key, can be used to effectively encrypt messages and digital signatures.

- **Private Key**
  - One half of the “key pair” used in conjunction with a public key
  - Both the public and the private keys are needed for encryption /decryption but only the owner of a private key ever needs to know it. Using the RSA system, the private key never needs to be sent across the Internet.
  - The private key is used to decrypt text that has been encrypted with the public key. Thus, if I send you a message, I can find out your public key (but not your private key) from a central administrator and encrypt a message to you using your public key. When you receive it, you decrypt it with your private key. In addition to encrypting messages (which ensures privacy), you can authenticate yourself to me (so I know that it is really you who sent the message) by using your private key to encrypt a digital certificate.

Digital Certificate

- A digital certificate is an electronic "credit card" that establishes your credentials when doing business or other transactions on the Web. It is issued by a certification authority (CA). It contains your name, a serial number, expiration dates, a copy of the certificate holder's public key (used for encrypting messages and digital signatures), and the digital signature of the certificate-issuing authority so that a recipient can verify that the certificate is real. Some digital certificates conform to a standard, X.509. Digital certificates can be kept in registries so that authenticating users can look up other users' public keys.

DH Key pair used by SSL servers ŧ

- This is a private/public key pair that is used by some cipher suites to encrypt the SSL/TLS handshaking messages.
- Possession of the private portion of the key pair allows an eavesdropper to decrypt traffic on SSL/TLS connections that use DH encryption during handshaking.
- The DH (Diffie-Hellman) key exchange, also called exponential key exchange, is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming.
- The most serious limitation of Diffie-Hellman (DH key) in its basic or "pure" form is the lack of authentication. Communications using Diffie-Hellman all by itself are vulnerable to man in the middle attacks. Ideally, Diffie-Hellman should be used in conjunction with a recognized authentication method such as digital signatures to verify the identities of the users over the public communications medium

Man in the Middle attack

- A man in the middle attack is one in which the attacker intercepts messages in a public key exchange and then retransmits them, substituting his own public key for the requested one, so that the two original parties still appear to be communicating with each other.
- The attack gets its name from the ball game where two people try to throw a ball directly to each other while one person in between them attempts to catch it. In a man in the middle attack, the intruder uses a program that appears to be the server to the client and appears to be the client to the server. The attack may be used simply to gain access to the message, or enable the attacker to modify the message before retransmitting it.

CA (Client Authentication certificate) ŧ

- If configured with a CA certificate, the DeviceMaster requires all SSL/TLS clients to present an RSA identity certificate that has been signed by the configured CA certificate. As shipped, the DeviceMaster is not configured with a CA certificate and all SSL/TLS clients are allowed.
- This uploaded CA certificate that is used to validate a client's identity is sometimes referred to as a "trusted root certificate", a "trusted authority certificate", or a "trusted CA certificate". This CA certificate might be that of a trusted commercial certificate authority or it may be a privately generated certificate that an organization creates internally to provide a mechanism to control access to resources that are protected by the SSL/TLS protocols.

Notes:

ŧ All DeviceMaster units are shipped from the factory with identical configurations. They all have the identical, self-signed, Comtrol Server RSA Certificates, Server RSA Keys, Server DH Keys, and no Client Authentication Certificates. For maximum data and access security, you should configure all DeviceMaster units with custom certificates and keys.

References:

PKI (public key infrastructure) http://searchsecurity.techtarget.com/
How Public and Private Key Cryptography Works http://searchsecurity.techtarget.com/
Who Provides the Infrastructure
RSA Key pair http://en.wikipedia.org/wiki/RSA
Digital Certificate http://searchsecurity.techtarget.com/
DH Key http://searchsecurity.techtarget.com/
Man in the Middle attack http://searchsecurity.techtarget.com/

PortVision Considerations when setting security

(return to top)

- PortVision must scan the DeviceMaster BEFORE configuring security
- PortVision must be aware of the DeviceMaster before setting either Secure Data mode or Secure Config mode.
- The 'Upload' and 'Reboot' ICONS on the Launch bar will be 'grayed out' as part of the config options
- If PortVision discovers the DeviceMaster AFTER setting security, the following conditions occur
  - The IP address of the DeviceMaster will not be displayed
  - The "Configure Device" tabs will not be present
  - The IP will be displayed as DHCP without the ability to modify
  - The 'Upload' and 'Reboot' ICONS on the Launch bar will be 'grayed out'

DeviceMaster Security Feature Grid

	Weakest					Strongest
	0	1	2	3	3	4
Supported by:	None	Pass word	Authent-ication	Secure Config	Secure Data	Key & Certificate
RedBoot	yes	yes	yes	no	yes	no
SocketServer	yes	yes	yes	yes	yes	yes
NS-Link Driver / MAC	yes	yes	yes	no	no	no
NS-Link Driver / IP	yes	yes	yes	yes with Res NS-Link	no	no
Serial Monitoring	yes	yes	yes	no	yes §	no
TCP to serial ports	yes	yes	yes	no	No	no
SSH (web interface) SSL (serial ports)	no no	no no	no no	yes no	no yes	no no
UDP To serial ports	yes	yes	yes	disabled	disabled	disabled
Telnet/Port23	yes	yes	yes	disabled	yes §	disabled
SSH Port22	yes	yes	yes	yes	yes	yes
Telnet Port 4607	yes	yes	yes	disabled	yes	yes
SSH (putty) 4607	no	no	no	yes	disabled	disabled
HTTP (Port80)	yes	yes	yes	disabled	disabled	disabled
HTTPS (Port443)	no	no	no	yes	yes	yes
Secure Port Redirector	yes	yes	yes	yes	yes	yes
Email	yes	yes	yes	disabled	disabled	disabled
SNMP	yes	yes	yes	disabled	disabled	disabled
RFC1006	yes	yes	yes	disabled	disabled	disabled

§ Enable Monitoring Secure Data via Telnet must be enabled. SSH does not support port monitoring
From a web page you may set the "securemon enable" option. You may now use the monitoring feature.

TCP and UDP Socket Ports used by the DeviceMaster products

(return to top)

Following list is all of the logical TCP and UDP socket ports implemented in DeviceMasters.

22 SSH
23 Telnet

- TCP Ports 22 (ssh) and 23 (telnet) are used for administrative and diagnostic purposes and aren't required for normal use and are enabled by default and port 23 may be disabled.

80 HTTP
443 SSL or HTTPS

- TCP Ports 80 (http) and 443 (https) are used by the web server for administration and configuration and are enabled by default and cannot be disabled.

102 RFC1006

- TCP Port 102 is used for RFC1006 (ISO over TCP) serial port access. Not used for normal NS-Link SocketServer access. The RFC1006 server can be disabled by setting the server port number to -1 and is enabled by default.

161 SNMP

- UDP Port 161 is used by the SNMP agent if SNMP is enabled which is the default.

4606

- TCP Port 4606 is required if you want to use NS-Link or PortVision if you want to update firmware without setting up a TFTP server and this port cannot be disabled.

4607

- TCP Port 4607 is only used for diagnostic purposes and isn't required for normal operation and this port cannot be disabled.

- If SocketServer is to be used, then the user may enable usage of TCP or UDP ports for access to the serial ports. These ports are not enabled by default and are also user configurable to different values. Defaults for TCP would begin at 8000 and for UDP would begin at 7000.

TCP **8000 - +1**

- Incremented per serial port on DeviceMaster. (Ex: DeviceMaster 16 port would have 8000 through 8015)

UDP **7000 - +1**

- Incremented per serial port on DeviceMaster. (Ex: DeviceMaster 16 port would have 7000 through 7015)

DeviceMaster RTS Security Features

(return to top)

Overview

- Beta test SocketServer firmware version 6.13.02
- Adds secure web access using https protocol on port 443.
- Adds SSH console server on port 22.
- Two new modes:
  - Secure Data – SSL encryption for serial-port data streams for both NS-Link and SocketServer.
  - Secure Config – encrypts/authenticates configuration and administration operations (web server, IP settings, load SW, etc.).
- Two New checkbox fields on "Configure Security" web page to enable/disable secure data/config modes.
- Two new values for http READ and WRITE commands:
  - A2: Secure data mode enable
  - A3: Secure config mode enable
- New byte of User Data in admin ID response containing feature enable bits:

Bit	Bit Description
0	Telnet/ssh enable
1	SNMP enable
2	Secure data enable
3	Secure config enable

Security Modes

- Secure Data
  - Requires SSL encryption of TCP connections to SocketServer (ports 8000, 8001, 8002, ...).
  - Disables UDP access to SocketServer.
  - Disables RFC1006 (ISO-over-TCP) access to SocketServer.
  - Disables MAC-mode access to serial ports. MAC mode admin and ID commands are still allowed.
  - Requires SSL encryption of NS-Link TCP connections (port 4606). Not directly supported by Windows/Linux NS-Link drivers. Linux driver has been tested using stunnel, but manual setup is required.
  - Requires SSH instead of telnet connection to the diagnostic log (TCP port 4607).

- Secure Config
  - Disables MAC mode admin commands except for ID request†
  - Disables TCP/IP admin commands except for ID request†
  - Disables telnet console access (port 23)†
  - Disables unencrypted http:// access via port 80.
  - Disables e-mail notification and SNMP features.

† Affects both RedBoot and SocketServer/NSLink applications

SSH Server

- Requires password authentication – even if password is empty.
- Enabled/disabled along with telnet access independently of secure data and secure config modes.
- DeviceMaster uses third-party MatrixSSH library from PeerSec Networks: http://www.peersec.com/

SSL Introduction

- Provides both encryption and authentication
  - Encryption prevents a third-party eavesdropper from viewing data that is being transferred.

- Authentication allows both the client (e.g. web browser) and server(e.g. RTS) to ensure that only desired parties are allowed to establish connections. This prevents both unauthorized access and "man-in-the-middle" attacks on the communications channel.
- Two slightly different SSL protocols are supported by DeviceMaster: SSLv3 and TLSv1.
- DeviceMaster uses third-party MatrixSSL library from PeerSec Networks: http://www.peersec.com/matrixssl.html

SSL Authentication

- Authentication means being able to verify the identity of the party at the "other end" of a communications channel. A username/password is a common example of authentication.
- SSL/TLS protocols allow authentication using either RSA certificates or DSS certificates. DeviceMaster supports only RSA certificates.
- Each party (client and server) can present an ID certificate to the other.
- Each ID certificate is signed by another "authority" certificate or key.
- Each party can then verify the validity of the other's ID certificate by verifying that it was signed by a trusted authority. This verification requires that each party have access to the certificate/key that was used to sign the other party's ID certificate.
- Server Authentication: the mechanism by which the DeviceMaster proves its identity.
  - The DeviceMaster (generally an SSL server) can be configured by uploading an ID certificate that is to be presented to clients when they connect to the DeviceMaster.
  - The private key used to sign the certificate must also be uploaded to the DeviceMaster. NB: possession of that private key will allow eavesdroppers to decrypt all traffic to and from the DeviceMaster.
  - The corresponding public key can be used to verify the ID certificate but not to decrypt traffic.
  - As shipped, all DeviceMaster units will contain identical self-signed ID certificates and private keys. This means that somebody could (with a little effort) extract the factory default private key from the DeviceMaster unit firmware and use that private key to eavesdrop on traffic to/from any other DeviceMaster unit that is being used with the as-shipped factory private key.
  - The public/private key pairs and the ID certificates can be generated using openssl command-line tools.
  - If the server authentication certificate in the DeviceMaster is not signed by an authority known to the client (as shipped, they are not), then interactive SSL clients such as web browsers will generally warn the user.
  - If the name in server authentication certificate does not match the hostname that was used to access the server, then interactive SSL clients such as web browsers will generally warn the user.
  - Clients can generally be configured to accept a particular unknown server certificate so that the user is not subsequently warned.
- Client Authentication: the mechanism by which the DeviceMaster verifies the identity of clients (e.g. web browsers, port-redirectors, etc.).
  - The DeviceMaster (generally an SSL server) can be configured by uploading a trusted "authority" certificate that will be used to verify the ID certificates presented to the DeviceMaster by SSL clients. This allows the customer to restrict access to the DeviceMaster to a limited set of clients which have been configured with corresponding ID certificates.
  - DeviceMaster units will be shipped without an authority certificate and will not require clients to present ID certificates. This allows any and all SSL clients to connect to the DeviceMaster.
  - Both authority and signed ID certificates can be created using command-line openssl tools.
- Certificates for both the DeviceMaster and for clients can be created by customers in order to manage access to the DeviceMaster and to verify that nobody has intercepted traffic to/from the DeviceMaster.
- Certificates can be obtained from commercial certificate authorities (VeriSign, Thawte, Entrust, etc.).
- Certificates can be created by users for their own use by using openssl command line tools or other applications.
- Certificates and keys to be uploaded to the DeviceMaster must be in the .DER binary file format, not in the .PEM ASCII file format. (The openssl tools can create files in either format and can convert files back and forth between the two formats.)
- Configuring Certificates and Keys
- Configured by 4 uploaded files on the bottom "Key and Certificate Management" portion of the security configuration web page:

  1. RSA Key
     - Can be used to encrypt the initial connection handshaking.
     - Used along with RSA Server Certificate (below) to provide the client
     - with proof of the DeviceMaster's identity.
  2. RSA Server Certificate
     - Used along with RSA Key (above) to provide the client with proof of the DeviceMaster's identity.

3. DH Key
    - Can be used to encrypt the initial connection handshaking.
4. Client Authentication Certificate
    - Used to verify the identity of SSL clients. If configured, SSL clients must present an ID certificate that has been signed by this certificate.
    - Units are shipped without a client authentication certificate, so SSL clients are not required to authenticate themselves.

- For each of the above four items
    - The “Set” button allows the user to upload a file.
    - The “Delete” button allows the user to delete an uploaded file and return to using the factory configuration.
- The first two items (RSA Key, RSA Server Certificate) must be updated as a matched set: the uploaded certificate must have been created using the uploaded key.

SSL Performance

- Encryption/decryption is a CPU-intensive process, and using encrypted data streams will limit the number of ports that can be maintained at a given serial throughput. For example, the table below shows the number of ports that can be maintained by SocketServer at 100% throughput for various cipher suites and baud rates.

	9600	38400	57600	115200
RC4-MD5	32	16	10	5
RC4-SHA	32	13	9	4
AES128-SHA	28	7	5	2
AES256-SHA	26	7	4	2
DES3-SHA	15	3	2	1

Note that these throughputs required 100% CPU usage, so other features such as the web server are very unresponsive at the throughputs shown above. To maintain a usable web interface, one would want to stay well below the maximum throughput/port numbers above.

- The overhead required to set up an SSL connection is also significant. The time required to open a connection to SocketServer varies depending on the public-key encryption scheme used for the initial handshaking. Typical setup times for the three public-key encryption schemes supported by the DeviceMaster are shown below:
    RSA 0.66 seconds
    DHE 3.84 seconds
    DHA 3.28 seconds

- Since there is a certain amount of overhead for each block of data sent/received on an SSL connection, the SocketServer polling rate and size of bocks that are written to the SocketServer also has a noticeable effect on CPU usage. Writing larger blocks of data and a slower SocketServer polling rate will decrease CPU usage and allow somewhat higher throughputs.

SSL Cipher Suites

- An SSL connection uses four different facilities, each of which can use one of several different ciphers or algorithms. A particular combination of four ciphers/algorithms is called a “cipher suite”.
- A Cipher Suite consists of

1. Public Key Encryption Algorithm
    - used to protect the initial handshaking and connection setup.
    - typical options are RSA, DH, DHA, DHE, EDH, SRP, PSK
    - DeviceMaster supports RSA, DHA, DHE
2. Authentication Algorithm
    - used to verify the identities of the two parties to each other.
    - typical options are RSA, DSA, ECDSA
    - DeviceMaster supports only RSA

3. Stream Cipher
    - used to encrypt the user-data exchanged between the two parties.
    - typical options: RC4, DES, 3DES, AES, IDEA, Camellia, NULL
    - DeviceMaster supports RC4, 3DES, AES
4. Message Authentication Code
    - hash function (checksum) used to verify that each message frame has not be corrupted or changed while in transit.
    - typical options include MD5, SHA, MD2, MD4
    - DeviceMaster supports MD5, SHA

- In the design of the SSL/TLS protocols the choices of four of the above are not independent of each other: only certain combinations are defined by the standards. The standard combinations of protocol (SSL or TLS) and cipher suites support by DeviceMaster are shown in the attached table.

DM-RTS Supported Cipher Suits

Protocol	Public Key	Authentication	Cipher	MAC
SSL	RSA	RSA	3DES	SHA
SSL	RSA	RSA	RC4	SHA
SSL	RSA	RSA	RC4	MD5
SSL	DHE	RSA	3DES	SHA
SSL	DHA	RSA	RC4	MD5
SSL	RSA	RSA	NULL	MD5
SSL	RSA	RSA	NULL	SHA
TLS	RSA	RSA	AES128	SHA
TLS	RSA	RSA	AES256	SHA
TLS	DHE	RSA	AES128	SHA
TLS	DHE	RSA	AES256	SHA
TLS	DHA	RSA	AES128	SHA
TLS	DHA	RSA	AES256	SHA

SSL Resources

- Standard reference book is SSL and TLS by Eric Rescorla
- Wikipedia page on SSL/TLS provides a good overview:
- http://en.wikipedia.org/wiki/TLS
- openssl – contains command-line tools to
  - create/examine keys/certificates
  - act as client or server
  - http://www.openssl.org/
- ssldump – command line tool that displays a human-readable dump of an SSL connection's handshaking and traffic:
  - If provided with server's private key, can decrypt data stream
  - Can display decoded data stream in ascii/hex
  - Can display contents of handshaking packets (including ID certificates)
  - http://www.rtfm.com/ssldump/

DeviceMaster Security Feature Grid			
	Secure Data	Secure Config	Secure Data+Config
MAC (admin)	enabled	disabled*	disabled*
MAC (async)	disabled	enabled	disabled
TCP 4606 (admin)	SSL, enabled	clear, disabled*	SSL, disabled*
TCP 4606 (async)	SSL	clear	SSL
UDP	disabled	user-configured	disabled
telnet/RFC2217	user-configured	user-configured	user-configured
RFC1006	disabled	user-configured	disabled
4607 (diag log)	SSH	telnet	SSH
8000 (serial port)	SSL	clear	SSL
console (config)	telnet on port 23 SSH on port 22	SSH on port 22	SSH on port 22
web	clear on port 80 SSL on port 443	SSL on port 443	SSL on port 443
SMTP,SNMP	user-configured	disabled	disabled
RedBoot MAC	enabled	disabled*	disabled*
RedBoot 4606	enabled	disabled*	disabled*
RedBoot telnet	user-configured	disabled	disabled

* admin commands disabled except for read-only ID command required by NS-Link to identify device.
The intention is to allow NS-Link to operate via SSL connection to port 4606 while in secure-data mode, and to allow NS-Link to operate via MAC connection while in secure-config/insecure-data mode.

Configure/Enable Security Features

(return to top)

Security features may be enabled in either the web page of the DeviceMaster or by using PortVision. These options are based on selecting simple check boxes. Key and Certificate Management must be done using the web pages of the DeviceMaster.

This discusses the following topics divided into Four areas:

Security Configuration Area

- **Enable Secure Data Mode**
  - If Secure Data mode is enabled TCP connections which carry data to/from the serial ports will be encrypted using SSL or TLS security protocols. This includes the following:
    - TCP connections to the per-serial-port TCP ports (default is 8000, 8001, 8002, ...) are encrypted using SSL/TLS.
    - TCP connections to TCP port 4606 on which the DeviceMaster implements the Comtrol proprietary serial driver protocol are encrypted using SSL/TLS.
    - Since SSL/TLS can not be used for either UDP data streams or for the Comtrol proprietary MAC mode Ethernet driver protocol, both UDP and MAC mode serial data transport features are disabled.
    - In order to minimize possible security problems e-mail and RFC1006 features are also disabled in Secure Data mode.
  - In addition to encrypting the data streams, it is possible to configure the DeviceMaster so that only authorized client applications can connect using SSL/TLS. See the Client Authentication section for details.

- **Enable Secure Config Mode**
  - If Secure Config mode is enabled, unencrypted access to administrative and diagnostic functions is disabled. Secure Config mode changes DeviceMaster behavior as follows:
    - Telnet access to administrative and diagnostic functions is disabled. SSH access is still allowed.
    - Unencrypted access to the web server via port 80 (http:// URLs) is disabled. Encrypted access to the web server via port 443 (https:// URLs) is still allowed.
    - Administrative commands that change configuration or operating state which are received using the Comtrol proprietary TCP driver protocol on TCP port 4606 are ignored.
    - Administrative commands that change configuration or operating state that are received using the Comtrol MAC mode proprietary Ethernet protocol number 0x11FE are ignored.

- **Enable Telnet/ssh**
  - This option enables or disables the telnet security feature after you click Save and the DeviceMaster has been rebooted. This option is enabled by default.

- **Enable SNMP**
  - This option enables or disables the SNMP security feature after you click Save and the DeviceMaster has been rebooted. This option is enabled by default.

Key and Certificate Management (Done only in the Edit Security Configuration Web page)

- **Client Authentication**
  - If desired, controlled access to SSL/TLS protected features can be configured by uploading a client authentication certificate to the DeviceMaster. By default, the DeviceMaster is shipped without a CA (Certificate Authority) and therefore allows connections from any SSL/TLS client.
  - If a CA certificate is uploaded, the DeviceMaster only allows SSL/TLS connections from client applications that provide to the DeviceMaster an identity certificate that has been signed by the CA certificate that was uploaded to the DeviceMaster.
  - This uploaded CA certificate that is used to validate a client's identity is sometimes referred to as a "trusted root certificate", a "trusted authority certificate", or a "trusted CA certificate". This CA certificate might be that of a trusted commercial certificate authority or it may be a privately generated certificate that an organization creates internally to provide a mechanism to control access to resources that are protected by the SSL/TLS protocols.
  - To control access to the DeviceMaster's SSL/TLS protected resources you should create your own custom CA certificate and then configure authorized client applications with identity certificates signed by the custom CA certificate.

- **Keys and Certificates**
  - For secure operation, the DeviceMaster uses a set of four keys and certificates. These keys and certificates may be configured by the user.
  - **RSA Key Pair used by SSL and SSH servers**
    - This is a private/public key pair that is used for two purposes:
    - It is used by some cipher suites to encrypt the SSL/TLS handshaking messages. Possession of the private portion of this key pair allows an eavesdropper to both decrypt traffic on SSL/TLS connections that use RSA encryption during handshaking.
    - It is used to sign the Server RSA Certificate in order to verify that the DeviceMaster is authorized to use the server RSA identity certificate. Possession of the private portion of this key pair allows somebody to pose as the DeviceMaster.
    - If the Server RSA Key is to be replaced, a corresponding RSA identity certificate must also be generated and uploaded or clients are not able to verify the identity certificate.

- **RSA Server Certificate used by SSL servers**
  - This is the RSA identity certificate that the DeviceMaster uses during SSL/TLS handshaking to identify itself. It is used most frequently by SSL server code in the DeviceMaster when clients open connections to the DeviceMaster's secure web server or other secure TCP ports. If a DeviceMaster serial port configuration is set up to open (as a client) a TCP connection to another server device, the DeviceMaster also uses this certificate to identify itself as an SSL client if requested by the server.
  - In order to function properly, this certificate must be signed using the Server RSA Key. This means that the server RSA certificate and server RSA key must be replaced as a pair.

- **DH Key pair used by SSL servers**
  - This is a private/public key pair that is used by some cipher suites to encrypt the SSL/TLS handshaking messages.
  - Possession of the private portion of the key pair allows an eavesdropper to decrypt traffic on SSL/TLS connections that use DH encryption during handshaking.

- **Client Authentication Certificate used by SSL servers**
  - If configured with a CA certificate, the DeviceMaster requires all SSL/TLS clients to present an RSA identity certificate that has been signed by the configured CA certificate. As shipped, the DeviceMaster is not configured with a CA certificate and all SSL/TLS clients are allowed.

Using a web browser to set security features

- **Changing Security Configuration**
  - Use the following steps to change security settings in the DeviceMaster.
    - Enter the IP address of the DeviceMaster in the Address field of your web browser and press the Enter key.
    - Click Configure Security link.
    - Click the appropriate check boxes in the Edit Security Configuration area to enable or disable security accordingly. Refer to the DeviceMaster Edit Security Configuration Area subsection for detailed information.
    - After making changes to the Edit Security Configuration area, you must click Save and then OK.

- **Changing Keys and Certificates**
  - Use the following steps to update security keys and certificates in the DeviceMaster.
    - Enter the IP address of the DeviceMaster in the Address field of your web browser and press the Enter key.
    - Click Configure Security link.
    - Click Set for the appropriate key or certificate option in the Keys and Certificate Management area to configure security keys and certificates. Refer to the Key and Certificate Management subsection for detailed information.
    - Click Browse to locate the key or certificate file, highlight the file, and click Open.
    - Click Upload when you return to the Key and Certificate Management area.
    - The key or certificate notation changes from factory or none to User when the DeviceMaster is secure.
    - You do not need to click Save, but changes will not take effect until the DeviceMaster is rebooted.

Important Notes

- All DeviceMaster units are shipped from the factory with identical configurations. They all have the identical, self-signed, Comtrol Server RSA Certificates, Server RSA Keys, Server DH Keys, and no Client Authentication Certificates.
- For maximum data and access security, you should configure all DeviceMaster units with custom certificates and keys.
- You must reboot the DeviceMaster for any security changes to go into effect.

Removing DeviceMaster Security Features

(return to top)

When presented with a DeviceMaster that has had all security options set and the user is unaware of what the settings are, the restoring of a DeviceMaster can be very difficult, if at all possible.

It may be necessary to use the DeviceMaster debug dongle provided with the SDK (Software Developers Kit), or return the DeviceMaster to Comtrol after obtaining an RMA in order to have Comtrol reflash the unit to defaults.

In order for the user to be able to remove the security settings from a DeviceMaster, one of the following two conditions must be true

1) For a Bootloader Command Console using serial port 1 connection
    a. Bootloader timeout set to value greater than 10 seconds (default is 15 seconds)
    b. A known good null modem cable
    c. A com port on PC/Laptop
2) For a Bootloader Command Console using an Ethernet connection
    a. No password or a known password
    b. A known or discoverable IP address
        i. A utility such as Angry IP Scanner from www.angryip.org may be used to discover IP addresses. If the IP range is unknown, a full scan from 0.0.0.1 to 255.255.255.255 may take a LONG time
    c. An Ethernet cable
    d. A PC/Laptop with a telnet application installed such as PuTTY included in PortVision.

Using a null modem cable to access the Bootloader (RedBoot) command console
Use the following procedure to set up serial connection with a terminal server program (for example, Test Terminal (WCom2), HyperTerminal or Minicom) and the DeviceMaster. Test Terminal (WCom2) is available in PortVision or you could install the Comtrol Utility package.

- Determine the com number of the port on the PC to be used.
    - Generally this is com1 or com2
- Connect the null modem cable from comx on the PC to the first serial port on the DeviceMaster. Some DeviceMaster models supply a null modem cable.
- Open a terminal application such as Test Terminal which is supplied in PortVision
    - Configure the serial port to the following values
        - Bits per second = 57600
        - Data bits = 8
        - Parity = None
        - Stop bits = 1
        - Flow control = None
- Reset the DeviceMaster.
    - Note: Depending on the model, disconnect and reconnect the power cable (external power supply and no power switch) or turn the power switch on and then off (internal power supply).
- In the terminal program immediately type #!DM and press Enter.
    - Case does matter. Upper case must be used.
- At the RedBoot> prompt, type dis, and press Enter.
    - Note: If you do not disable the loading feature of the Bootloader within the time-out period (default is fifteen seconds), an application will be loaded from flash and started. If this happens, repeat Steps above. The #!DM command is the only case-sensitive command and must be in uppercase.

If you are able to connect to the RedBoot command console port, enter these commands:

Command	Result
Password ↵	clears password
Auth none ↵	removes authentication level
Secureconf disable ↵	disable the secure config mode
Securedata disable ↵	disables the secure data option
IP (ip information) ↵	reset the IP settings to default values
Timeout (value) ↵	set a reasonable timeout value (suggest 15 seconds or greater)

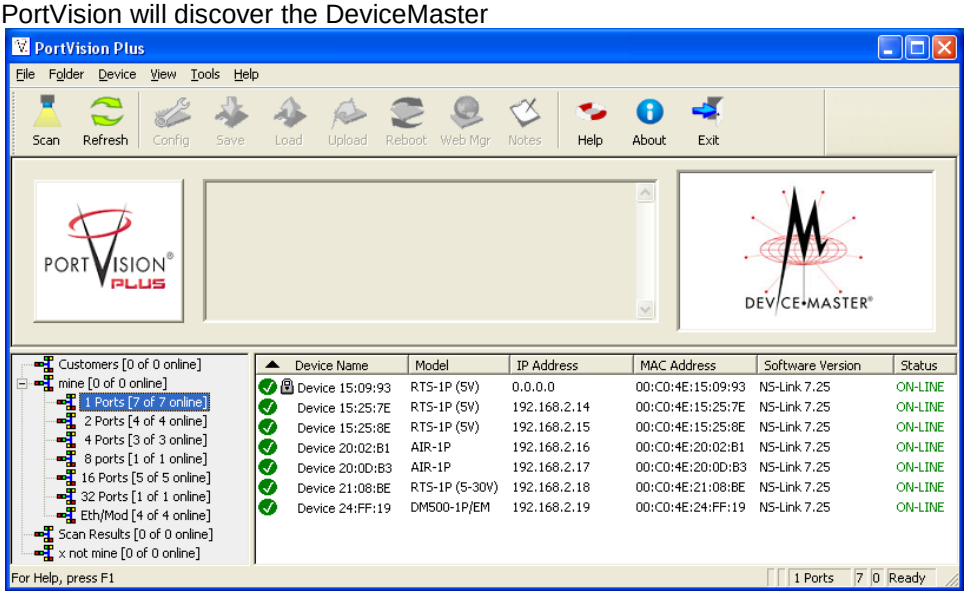
If the command is not available in RedBoot, then it must be entered in the dm> prompt using telnet or SSH. Enter all of the above commands that are possible before attempting a reset of the DeviceMaster. After resetting (rebooting) telnet to a dm> prompt and enter the remaining commands that were not entered above.

Once all of the above commands have been successfully entered, the web page may now be opened allowing access to the “Configure Security” page where the certificates may now be deleted.

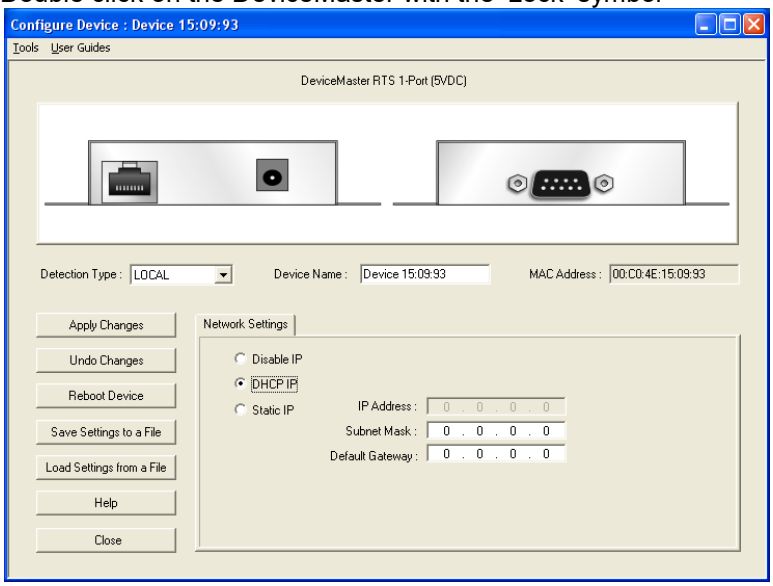
If the Bootloader timeout has been set too low to allow console port access, and the IP address cannot be discovered, then the DeviceMaster must be returned to Comtrol for reflashing.

If the IP address can be successfully determined, and there is no password, or a password is known, you may scan the DeviceMaster into PortVision. If a password has been entered and is unknown, the following will not work and the DeviceMaster must be returned to Comtrol to be reflashed back to factory defaults.

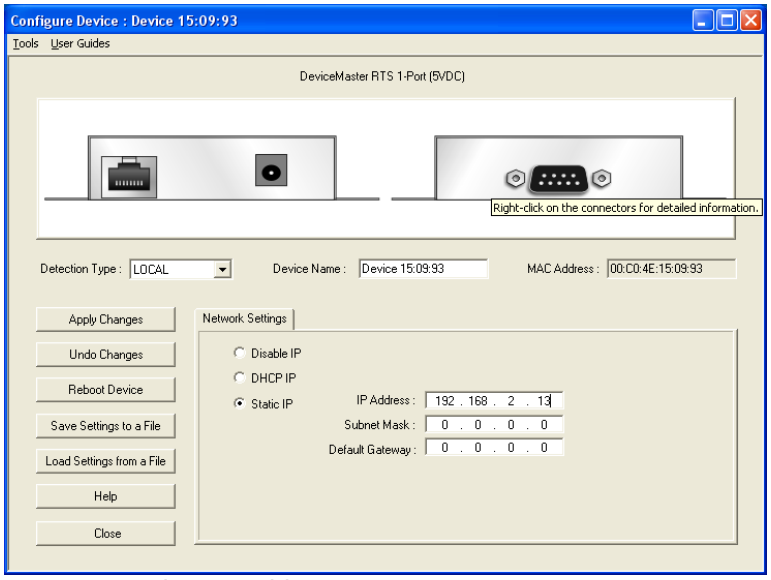
Connect the DeviceMaster directly to the PC running PortVision.
Open PortVision
Scan the network
PortVision will discover the DeviceMaster



Double click on the DeviceMaster with the ‘Lock’ symbol

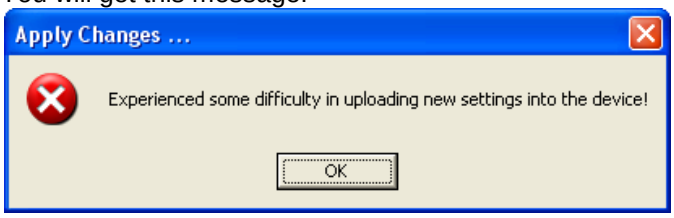


Change to ‘Static IP’



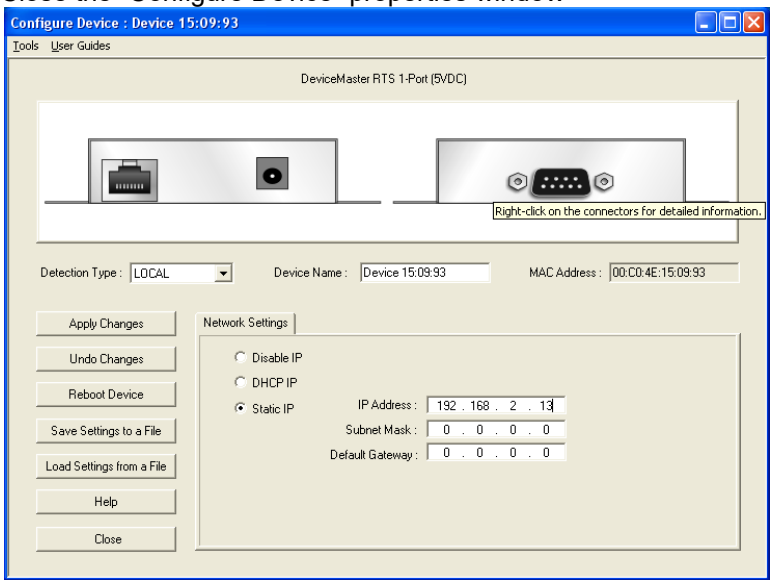
Enter appropriate IP address
Click ‘Apply Changes’ and ‘Close’

You will get this message:

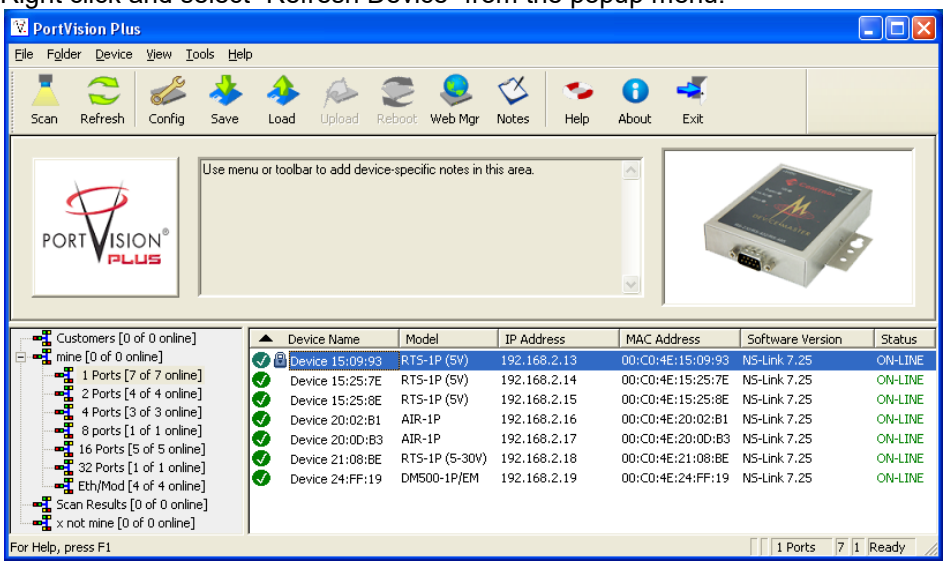


Click OK

Close the “Configure Device” properties window

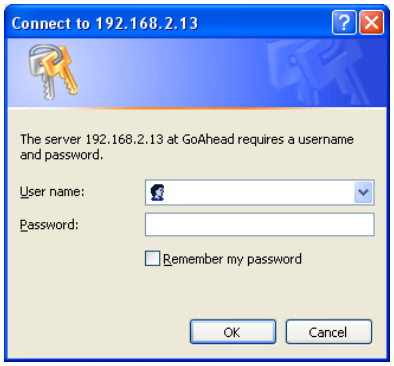


From PortVision
Right click and select “Refresh Device” from the popup menu.



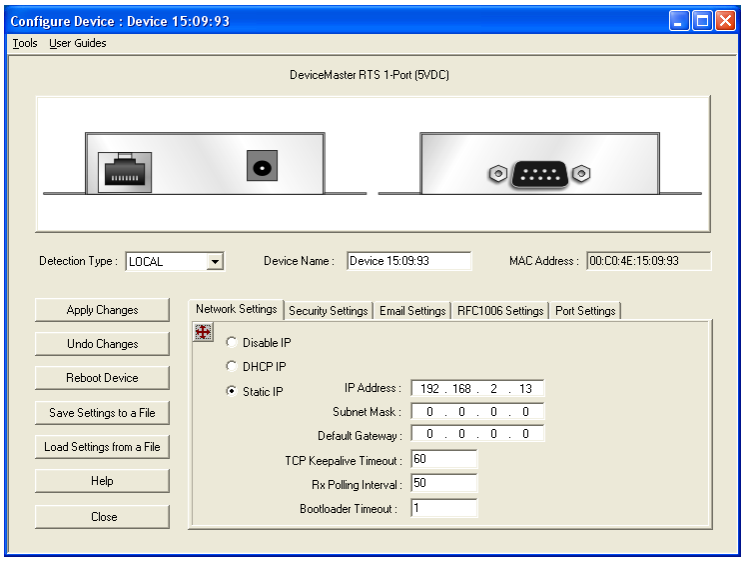
Now all columns should be properly displayed

Double click on the DeviceMaster with the ‘Lock’ symbol



Enter the password if prompted and click OK.

If the password is prompted for, and unknown, you cannot proceed further and the DeviceMaster will need to be returned to Comtrol for reflashing to factory defaults.



If the “Configure Device” properties window opens, you may now make the necessary changes to remove the security options.

Returning the DeviceMaster to Factory Defaults

(return to top)

DeviceMaster Memory:
The DeviceMaster RTS uses two types of memory, volatile and non-volatile. The volatile memory is in the form of DRAM and SRAM. They are used for program execution and buffers. Clearing the volatile memory, as its name suggests, requires powering off the DeviceMaster.

The non-volatile memory is in the form of flash and eeprom memories.

The flash memory is used for non-volatile program storage. Leaving the factory, there are two programs stored in the flash. They are the bootloader binary (Redboot.bin) and the default application binary (socketserver.bin). The bootloader binary is loaded into DRAM for execution, when the device is turned on. After a period of time, the bootloader loads the default application, socketserver.bin or, in some instances, a customer written custom application, into DRAM and it starts execution. It continues until the unit is powered off. The only access the user has to the binaries is if they decide to load a newer version. If this is done, the newer version overwrites that piece of flash. No user data is ever entered here.

The eeprom memory is programmed with a number of default values. The values that can be modified by the user are shown in the following list.

Parameter Name	Default Value	User Configurable	Method of Access		
			Web Telnet	Debug Server	Port
Authentication	None	Yes	No	No	Yes
IP Address	192.168.250.250	Yes	Yes	Yes	Yes
IP Mask	255.255.0.0	Yes	Yes	Yes	Yes
IP Gateway	192.168.250.1	Yes	Yes	Yes	Yes
Password	Blank	Yes	Yes	No	Yes
Telnet	Enable	Yes	Yes	Yes	Yes
Telnet Timeout	300 sec.	Yes	Yes	Yes	Yes
Bootloader Timeout	15 sec.	Yes	Yes	Yes	Yes
SNMP	Enable	Yes	Yes	Yes	Yes
SSL*	Disable	Yes	Yes	Yes	Yes

* SSL is a security feature available with SocketServer v7.00 and later.
Clearing FLASH:

The flash only has program binaries. There is no user data stored here. If it is necessary to erase the binaries, the default application, socketserver.bin, can be erased using the '**fis init**' command from the DeviceMaster Debug Port.

The use of the Debug Port is described in the **DeviceMaster RTS Installation and Configuration Guide**, Initial Configuration Section, Establishing a Serial Connection.

The **fis** command is explained in the **DeviceMaster RTS Installation and Configuration Guide**, Redboot Procedures Section, Redboot Command Overview.

There is no easy way to remove the bootloader binary. Removal of the bootloader binary would leave the DeviceMaster inoperable and essentially a brick. It will need to be returned to the factory to be reprogrammed before it can be used.

Clearing EEPROM

The user configurable values in the eeprom, can be accessed and set in three different ways. All of the values can be set using the Debug Port. Most of the values can be accessed by using the Web Server or Telnet.

Telnet Access

To use the **Telnet** to access the DeviceMaster configuration, reference the **DeviceMaster RTS Installation and Configuration Guide**, Redboot Procedures Section, Establishing a Telnet Connection.
Once the connection is established, entering 'help' will provide a list of the available commands.
Note: The method of authentication cannot be reset using this method.

Use the following commands to reset the factory default values.

'ip 192.168.250.250 255.255.0.0 192.168.250.1'	(resets ip, mask, gateway)
'password'	(resets the password)
'telnet enable'	(telnet is enabled)
'teltimeout 300'	(resets telnet timeout value)
'timeout 15'	(resets the bootloader timeout)
'snmp enable'	(enables snmp)
'ssl disable'*	(disables ssl)

*SSL command is only available on DeviceMaster products running SocketServer 7.0 and later.

Debug Port Access

To use the **Debug Port** to access the DeviceMaster configuration, reference the **DeviceMaster RTS Installation and Configuration Guide**, Redboot Procedures Section, Establishing a Serial Connection. Once the connection is established, entering 'help' will provide a list of the available commands. Use the following commands to reset the factory default values.

'auth none'	(resets authentication)
'ip 192.168.250.250 255.255.0.0 192.168.250.1'	(resets ip, mask, gateway)
'password'	(resets the password)
'telnet enable'	(telnet is enabled)
'teltimeout 300'	(resets telnet timeout value)
'timeout 15'	(resets the bootloader timeout)
'snmp enable'	(enables snmp)
'ssl disable'*	(disables ssl)

*SSL command is only available on DeviceMaster products running SocketServer 7.0 and later.

Web Server Access

To use the **Web Server** to access the DeviceMaster configuration, reference the **DeviceMaster RTS Installation and Configuration Guide**, Socket Port Configuration, Accessing Socket Configuration, Web Browser.
Once the connection is established, the web page **Help** system provides detailed configuration procedures and descriptions for all fields.

Note: The method of authentication and the password cannot be changed using this method.

Make sure you go to each of the following areas to reset the defaults values. Some of the values require resetting the box to take effect. Beware that changing the IP addresses, and resetting the box, it will not reconnect automatically. You will need to use the new IP address to reconnect.

Configure Network:

Check 'Use static configuration below.' box
Set IP Address to 192.168.250.250
Set Netmask to 255.255.0.0

Set Gateway to 192.168.250.1
Verify Bootloader Timeout is set to 15
Click Save
Click OK when reminded it is necessary to reboot to take effect.

Configure Security:

Verify Telnet Enable is checked.
Verify SNMP Enable is checked.

On devices running SocketServer v7.00 or later:

Verify SSL Enable is NOT checked.
Click Save
Click OK when reminded it is necessary to reboot to take effect.

Configure Email Messages:

Verify SMTP Server IP Address is 0.0.0.0
Verify all remaining options are clear.
Click Save
Click OK

On the Main page:
Click on Reboot.
On the Reboot DeviceMaster page:
Click on 'Set configuration for all ports to factory default settings.

Click on 'Yes: Reboot'

The DeviceMaster will reboot. When it starts running, everything will have been returned to factory default values. This can be verified by using Telnet, Debug Port or Web Server to read the values.
Remember, if you choose to verify the values, the IP address has been reset to 192.168.250.250.

Revision History
(return to top)

Version 1.0.0
Initial release