IP-VCON

VIRTUAL CONSOLE APPLICATION

USER'S MANUAL

FOR

DT-6XXX

EMBEDDED NETWORK PROCESSORS

RELEASE.VERSION 10.0

ISSUE 2

379 Campus Drive, Suite 100
Somerset, NJ 08873
fax: 732 667-1091
phone: 732 667-1080
email: sales@datatekcorp.com
http://www.datatekcorp.com

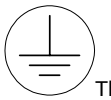
Datatek
APPLICATIONS, INC

TABLE of CONTENTS

Important Safety Instructions

The exclamation point within an equilateral triangle is intended to alert the user to the presence of important operating and maintenance (servicing) instructions in the literature accompanying the product. When installing, operating, or maintaining the DT-6XXX equipment on which this application runs, basic safety precautions should always be followed to reduce the risk of fire, electric shock, and injury to persons, including the following:

- ❑ Read and understand all instructions.
- ❑ Follow all warnings and instructions marked on this product.
- ❑ For information on proper mounting instructions, consult the User's Manual provided with this product.
- ❑ The telecommunications interface should not leave the building premises unless connected to telecommunication devices providing primary and secondary protection.
- ❑ This product should only be operated from the type of power source indicated in the User's Manual.
- ❑ This unit is intended to be powered from either –48 V DC or AC voltage sources. See User's Manual before connecting to the power source.
- ❑ The –48 V DC input terminals are only provided for installations in Restricted Access Areas locations.
- ❑ Do not use this product near water, for example, in a wet basement.
- ❑ Never touch uninsulated wiring or terminals carrying direct current or leave this wiring exposed. Protect and tape wiring and terminals to avoid risk of fire, electric shock, and injury to service personnel.
- ❑ To reduce the risk of electrical shock, do not disassemble this product. Service should be performed by trained personnel only. Opening or removing covers and/or circuit boards may expose you to dangerous voltages or other risks. Incorrect re-assembly can cause electric shock when the unit is subsequently used.
- ❑ For a unit intended to be powered from –48 V DC voltage sources, read and understand the following:
  - This equipment must be provided with a readily accessible disconnect device as part of the building installation.
  - Ensure that there is no exposed wire when the input power cables are connected to the unit.
  - Installation must include an independent frame ground drop to building ground. Refer to User's Manual.



This symbol is marked on the DT-6XXX, adjacent to the ground (earth) area for the connection of the ground (earth) conductor.

- ❑ This Equipment is to be Installed Only in Restricted Access Areas on Business and Customer Premises Applications in Accordance with Articles 110-16, 110-17, and 110-18 of the National Electrical Code, ANSI/NFPA No. 70. Other Installations Exempt from the Enforcement of the National Electrical Code May Be Engineered According to the Accepted Practices of the Local Telecommunications Utility.
- ❑ For a unit equipped with an AC Wall Plug-In Unit, read and understand the following:
  - A DT-6061 unit was tested with the K'TRON, Model KA-52A Wall Plug-In Unit and a DT-6X60 with a Phi Hong Model PSA-30U-240 Wall Plug-In Unit (110-240 V AC to 24 V DC).
  - Unplug this product from the wall outlet before cleaning. Do not use liquid cleaners or aerosol cleaners. Use a damp cloth for cleaning.
  - Do not staple or otherwise attach the power supply cord to the building surfaces.
  - Do not overload wall outlets and extension cords as this can result in the risk of fire or electric shock.
  - The socket outlet shall be installed near the equipment and shall be readily accessible.
  - The Wall Plug-In unit may be equipped with a three-wire grounding type plug, a plug having a third (grounding) pin. This plug is intended to fit only into a grounding type power outlet. Do not defeat the safety purpose of the grounding type plug.
  - Do not allow anything to rest on the power cord. Do not locate this product where the cord may be abused by persons walking on it.
  - Unplug this product from the wall outlet and refer servicing to qualified service personnel under the following conditions:
    a) When the powers supply cord or plug is damaged or frayed.
    b) If liquid has been spilled into the product.
    c) If the product has been exposed to rain or water.
    d) If the product does not operate normally by following the operating instructions. Adjust only those controls that are covered by the operating instructions because improper adjustment of other controls may result in damage and will often require extensive work by qualified technician to restore the product to normal operation.
    e) If the product has been dropped or the cabinet has been damaged.
    f) If the product exhibits a distinct change in performance.

Save These Instructions

1 INTRODUCTION

There should be two significant questions immediately apparent to anyone reading this document for the very first time. Those questions are:

1. What is a Virtual Console?
2. Why do I need a Virtual Console?

Before either of those questions can be answered, we must step back and look at what has become of networking today. Gone are the days of centralized call control, and large monolithic networks. Gone too are the capabilities those networks allowed. Common control and monitoring functions were simple features in a large network. Today, a network consists of nothing more than interconnected routing and terminus equipment. Multiple vendors inter-operate with data communications standards, but every device needs to be separately monitored, separately administered, and separately maintained.

In the not too recent past, the Simple Network Management Protocol (SNMP) was viewed as a means to allow a disjoint set of equipment become a single cohesive network. The strength of the SNMP management was in its ability to monitor equipment. This was done through a standard collection of data objects known collectively as the Management Information Base (MIB). There are two significant problems in administration (as opposed to monitoring) of devices using the SNMP. The first is that each device almost always implements a *proprietary* set of objects on the MIB for administration only purposes. This requires an SNMP manager to have intimate knowledge of the union of every possible device that could possibly be connected to a network. The second is that the SNMP has absolutely zero security. This has nothing to do with the User Defined Protocol (UDP) basis per RFC 867 as there are many very secure UDP based protocols. It has to do with the basic notion of the SNMP that any manager should be able to administer any device. There have been at least two revisions to the SNMP protocol. Both of these revisions have failed to provide any security, and many network devices have an agent that support only the first, and the RFC required, version of the SNMP protocol.

In order to provide some security, and a user friendly means of administration, nearly all network devices have adopted a console interface based on Transmission Control Protocol (TCP), using RFC 854 (or Telnet) encapsulation. An administrator can then use a Telnet client to connect to the device console to perform configuration and other administrative activities for that device. The connection is typically password protected. Since TCP is sequenced, another entity cannot piggyback on an administration connection. This solves the problem of network security, and the devices can still be monitored adequately by an SNMP manager. However, it creates the need to access a separate console for every entity on the

network. Sometimes, in the case of a host or application processor, multiple consoles simultaneously.

There is also a need to look at the output of various entities that are not network equipment. For example, a large network may have hundreds of server hosts. If an alarm is issued on one host, how is it monitored? Typically, these would come on a printer and an on-site administrator dealt with the issue. With the large number of servers, and a net workforce reduction, how is this to be done? The same can be said for Network Elements such as Recent Change Channels, and other “console” devices.

In the few paragraphs above, the concept of “console” has been defined in a rather round-about way. It is an endpoint that needs to be administered or monitored. A console may be on a network device, such as a router, or on a non-network device such as a SUN host, or even a #5ESS voice switch. Further, non-data network devices such as Digital Cross-connects may also be monitored and administered in the same manner. It does not matter that the native protocol to these devices is (B)X.25, Asynchronous, or any other protocol. They are all consoles, and their interface can be suitably manipulated in a straightforward manner.

The above should have answered the second question. There is a need to deal with the multitude of *consoles* in modern networks. There is a need to monitor those consoles autonomously. There is a need for a means to do this in a non-intrusive manner to the administrator. That need is addressed by the ***Virtual Console*** application, **IP-VCON**.

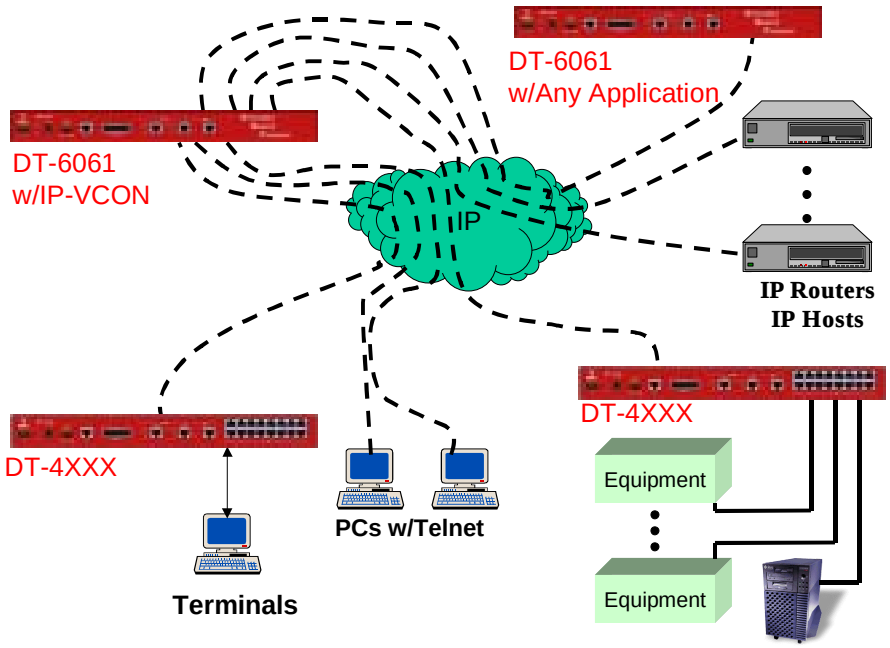
The **Virtual Console** application attaches, via TCP connections, to many *device consoles*, whether network based or physical (via a DT-4000 or equivalent). It will monitor those consoles, take action on exceptions as needed, and provide a single means to access every console on and off the network. The **Virtual Console** application is eternally vigilant and will notify when a device console indicates a problem situation. The **Virtual Console** can even determine outage of equipment.

Some of the problems mentioned in this introduction are easily handled by the **Virtual Console** application. For example, security is handled by defining multiple administrators, and giving each administrator their own identification and password. Each administrator can then be restricted to a list of *device console*s for which they have responsibility. This is more than a security issue, since there is no need to route alarm data to someone that cannot deal with the problem. Furthermore, the **Virtual Console** can be configured with multiple tiers of access to each console, allowing some administrators full access, while allowing others a limited command set, as might be necessary to display status or examine configuration.

2 TYPICAL DEPLOYMENT

In this section, a typical deployment of the **Virtual Console** application is discussed. Since the **Virtual Console** application can handle a large number of *device console* endpoints, there may not be a need to have more than one copy running in any given network.

Consider the Following Diagram:



In the diagram above, all of the network devices (on the right) have their consoles connected to the user's IP network, so that they are accessible via *telnet* over TCP.

- A device whose *console* appears only as a physical port can gain presence on the user's IP network by connecting the console port to a DT-4000 serial port.
- A *console* may be a BX.25 circuit by connecting to a LCN on an IP-PAD application of a DT-6061.
- The console may be an application console on a DT-6061.
- The *console* may be a router console, a Digital Cross-Connect, or any other device that can be connected to the network logically or physically.

Each person authorized to administer any of the network devices may connect to the **Virtual Console** application via Telnet, using a physical terminal or a Telnet application on a PC, as illustrated on the left of the diagram. After a login procedure to identify the administrator, the **Virtual Console** application gives access to all of

the device consoles to which the administrator is *allowed* access. Device consoles may be addressed by name or number. An administrator may patch through to any *allowed* console, and thus enter commands on that console or *watch* the console, receiving output or exception messages from the console. An administrator may also access alarm logs and configuration information about all *allowed* consoles. In addition to *allowed* consoles, administrators may be configured to have restricted access to consoles by way of administratively-configured *monitor groups*.

3 IP-VCON FEATURES

This section defines the features of the **Virtual Console** application. This is presented as a list, but some features require further elaboration.

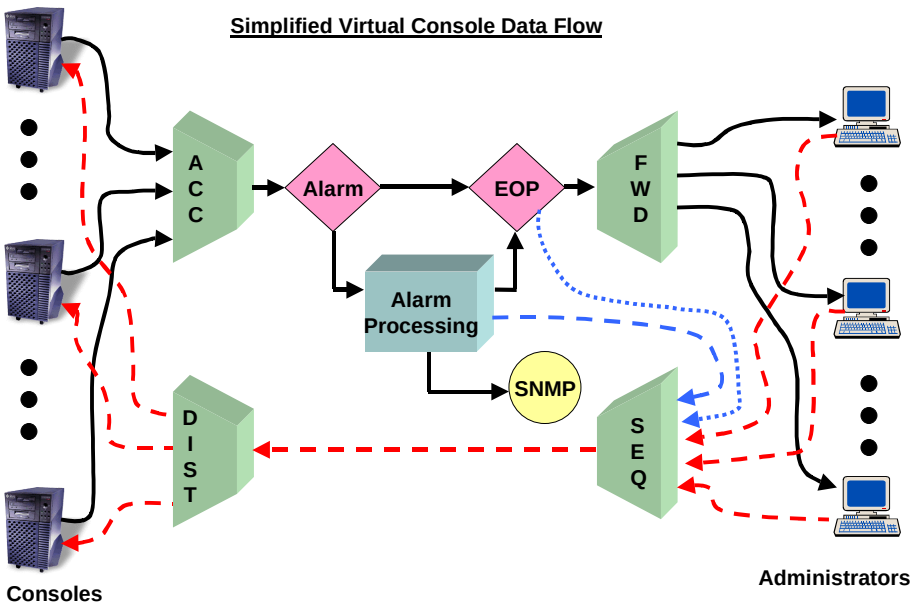
- ✓ 768 *Device Console* ports may be monitored and administered. Each device console may be monitored and administered by multiple administrators. Each console has a unique mnemonic name.
- ✓ 27 Simultaneous Administrators may be using the **Virtual Console**, through the *Administrator Console* interface.
- ✓ 64 Administrators may be defined to the **Virtual Console**. Each administrator definition has its own ID, Password, and list of allowed device consoles. An administrator has full administrative access to its individual list of allowed consoles, and can have limited access for monitoring other groups of consoles.
- ✓ 1 *Configuration (OA&M) Console* is available to be used by the **Virtual Console** administrator to set up the configuration of device console connections and individual administrator IDs and their capabilities.
- ✓ 32 Logger connections are available to selectively record device console activity monitored by the **Virtual Console**.
- ✓ Any type of device console may be monitored. Telnet, Physical, BX.25, LAPB, etc. Some protocols may require pre-processing by other DT-6061 applications (e.g. IP-PAD for X.25 or BX.25 circuits).
- ✓ Different access levels may be configured for up to 64 *monitor groups*. A monitor group defines a set of consoles and a limited command set that may be issued to those consoles. Assigning a monitor group to an administrator gives that administrator the ability to monitor those consoles using the limited command set.
- ✓ Any console may be monitored automatically for Alarms. The Alarm format may be standard, or user defined. This allows any console format to be used.
- ✓ An Alarm on a console may be programmed to forward a SNMP TRAP to a selected SNMP Manager whether or not an interested administrator is currently logged into the **Virtual Console**.
- ✓ If a device console has a login & password sequence, the operation is completed automatically by the **Virtual Console** application upon connection.
- ✓ Up to 16 automatic action scripts may be defined. These scripts may be individually associated with a console connection. When associated with a console, the console is monitored for a series of characters to trigger the script. Once triggered, the script will generate commands to that console or to another console configured for this script. For example, suppose we had a router that had a “BOOM” alarm. When it emits the “BOOM” alarm, the **Virtual Console** could tell the router to “restart” without intervention.

- ✓ Each console connection to the **Virtual Console** may be “Removed” from service, or “Restored” to service. When a console connection is “Removed” from service, the IP-VCON application will not attempt to make a connection to the console.
- ✓ Each administrator ID defined may be “Removed” from service, or “Restored” to service. When an administrator ID is “Removed” from service, access will not be allowed for that ID but all of the configured parameters are maintained.
- ✓ An alarm log is maintained for each console by severity. Each alarm severity maintains the last 30 alarms with a stamp relative to the current time. One use for this feature is logging the last 30 times a device has rebooted. For example, by defining a custom alarm (of any severity) for a reboot string, the **Virtual Console** will log every time a device restarts. This can be useful in the management of larger networks. Informational custom alarm text may be used for this purpose. The alarm log for a console may be cleared by an administrator.

4 IP-VCON DATA FLOW

It is important to understand how the *device console* endpoints are managed by the **Virtual Console** application. This allows effective use of all of the **Virtual Console** features.

The following diagram is a greatly simplified data flow through the **Virtual Console** application.



The flow of messages from a device *console* to an administrator begins with data arriving at the device console's TCP connection with the **Virtual Console**. The data may have originated from a physical connection via a DT-4000 or similar device. For alarm processing, the data is accumulated so that a check for the presence of an alarm can be performed. At the same time, the data is forwarded to any administrators that may be connected, so this accumulation function does not induce any noticeable delay in the data flow. Should alarm processing be necessary, it is done without affecting the traversal of the message to any connected administrators. The data is also subjected to an *end of prompt* test. An internal command sequencer needs to know when *end of prompt* is detected before sending automated commands to the device console. An *end of prompt* also triggers the **Virtual Console** application to display an appropriate *prompt* to all of the administrators that

are currently connected to the device console. The prompt for an administrator connection is described in section 10.

The data to a particular device console from one or many administrators flows through unmonitored by the **Virtual Console** application. If multiple administrators are connected simultaneously to the same device console, they will see the results of each other's input at each keystroke.

There may come a time when there are no *interested administrators* connected to the **Virtual Console** application during a significant event on a *console*. When that situation occurs, the **Virtual Console** shall still process the alarm conditions specified for that console. At the next connection, an *interested administrator* would see the alarm status in their prompt. If an action was specified for a particular alarm, it is autonomously initiated by the **Virtual Console** application.

In addition to the flow of data illustrated by the figure, there is an additional outlet for alarm data collected by the **Virtual Console**: logger ports. There can be up to 32 separate connections to logger ports. These connections allow any other data collection system to collect and process or permanently store alarms from the consoles under surveillance by the **Virtual Console**.

5 SUGGESTED REFERENCES

The following documents are resident at ***http://www.datatekcorp.com*** under the documentation button.

Document	**Scope**
DT-6061 Platform User's Manual.	Describes the DT-6061 Embedded Network Processor infrastructure and command set. This includes configuration information, hardware specifications, and SNMP MIB support. The DT-6061 is the infrastructure on which the Virtual Console application shall reside.
DT-4000 User's Manual.	Describes the DT-4000 multi-protocol access device. The DT-4000 is used as the interface for physical console connections, or for (B)X.25 links to the console circuits.
DT-6061 Redundant Operation White Paper.	Describes the method of operating the DT-6061 in a 1+1 sparing configuration. **Note**: *This paper is not posted on the above site. Contact the author for a copy via email.*

6 IP-VCON INTERFACES

The TCP port numbers associated with a DT-6061 application are normally determined by the **instance** number in which the application is installed. The IP-VCON is different in that premise because of the large number of distinct TCP port numbers, the need to simplify administration, and engineering configurations. The IP-VCON application may only be configured in **instance #1** of the DT-6061. Configuring it in any other instance will result in a periodic alarm after which the IP-VCON will proceed to sleep perpetually. Consult the DT-6061 infrastructure manual for information on how to install an application. It is strongly recommended that IP-VCON be the only application running on a DT-6061. Other applications can be installed and run, but only if they use a small number of TCP ports; furthermore, the resulting performance may be less than desirable.

The **Virtual Console** application is packaged in the file **`ip_vcon.dt6`**, and is installed on the DT-6061 by the name **`ip_vcon`**.

The TCP Numbers associated with the IP-VCON application (using **instance #1** as required) are as follows:

Set	#Channels	TCP Port#	Usage
OA&M console connection	1	10001	Configuration and administration of the IP-VCON application. This is the standard DT-6061 configuration TCP port number for instance #1. The general formula is 10,000 + instance#. Connections to this TCP port are made via a Telnet client.
Adm console connection	27	23	This is the administrator connection port. Up to 27 administrators use the same TCP port number to connect. The value is the standard telnet TCP port.
Logger connections	32	30000-30032	These are TCP ports that are used by loggers. A logger is either a PC or UNIX system, which is recording the output of the IP-VCON to a disk.

Set	#Channels	TCP Port#	Usage
Device Console connections	768	Dynamic	There are a total of 768 TCP ports reserved for Console connections. These are dynamically assigned at the time the IP-VCON originates the connections to consoles.

An example of how these circuits are used follows:

The IP-VCON is assigned to instance #1 on the DT-6061. The IP-VCON listens for inbound TCP connections on the OA&M console port, administrator port, and logger TCP port numbers specified above. Configuration for device consoles and for the users that may administer those device consoles is done through the IP-VCON OA&M port. Once configured, the IP-VCON initiates connections to all *console* endpoints that are “In Service”, and allows logins from all *administrators* that are "In Service".

7 INPUT CONVENTIONS

7.1 CONSOLE PORTS

There are two different console ports available to IP-VCON users: the OA&M port and the administrator port.

The OA&M console port has a login command with a password, like any other DT-6061 application (see the login command, 8.1). The OA&M console port is used to configure the IP-VCON application and to display measurements and status of all the connections managed by IP-VCON.

The administrator port is used by up to 27 network administrators to connect through IP-VCON to the device consoles monitored by IP-VCON, and also to display IP-VCON's status, configuration settings, and alarm logs for those console connections. Upon connecting to the administrator port a user must enter a login ID and a password that matches one of the configured *administrators*. Then the user has access to the set of device consoles that has been specified for that administrator.

7.2 COMMAND LINE INPUT

There are two different **Virtual Console** command sets, one for the *OA&M* console and another for the *administrator* consoles. For administrator consoles, there is a further distinction: user input at an administrator console may be directed either to a device console or to the **Virtual Console** application itself. Administrator consoles are frequently patched through to the device consoles that are managed by the **Virtual Console**, and user input is forwarded directly to a device console, to be interpreted by the device's own command handler. In order to divert input to the **Virtual Console** administrator command handler, the user must begin the line of input with **!**. Thus commands at the administrator consoles all begin with **!**.

All **Virtual Console** command parameters are given on the command line. Parameters of the form **<name>=<value>** may be given in any order.

Commands may be entered in upper or lower case. Parameters of the form **<name>=<value>** may use upper or lower case for **<name>**. Case is preserved for values. Backspace erases one character. The whole line is erased by typing **ctrl-X**. Some commands run in steps delimited by a **[NEXT]** prompt; those commands may be stopped by typing the **del** character (hex 7f).

7.3 COMMAND HISTORY

Both the OA&M console and the administrator consoles provide a simple *command history* capability. At the OA&M console, typing the *escape* key erases the current input and replaces it with the previous command in the history. Up to five command

lines are saved in the history, and may be accessed by typing *escape* repeatedly. At the administrator console, *history* is accessed by typing the *escape* key after first beginning a line with **!**.

7.4 PATTERN MATCHING

The **Virtual Console** uses text *patterns* as a means to match user input against database or console output text in several operations described later. The form of pattern matching is a simple variation of the *filename* matching that is used in PC and host operating systems. The elements are:

- Normal characters match themselves.
- ∗ matches any string of 0 or more characters
- **?** matches any single character
- **[chars]** matches any one of the enclosed characters.
- **[!chars]** matches any character that is *not* one of the enclosed characters**.**

Some examples:

a∗	matches any string beginning with the letter **a**.
∗[tT][nN]∗	matches any string that contains the letters **TN** together in upper or lower case.

Note the use of ∗ at the beginning and end of the pattern when it is desired that the pattern be found *anywhere* within the string.

7.5 SPECIAL CHARACTERS

In some commands, the user must configure text that matches the output from consoles being managed by the **Virtual Console** application. In those cases, a special input convention is used to enter special characters, because the **Virtual Console** will not accept special characters entered directly at the OA&M or administrator consoles. The special character conventions are:

\SOM	The "start of message" character, hex **01**.
\EOM	The "end of message" character, hex **1F**.
\EOP	The "end of prompt" character, hex **19**.
\r	The "carriage return" character, hex **0D**.
\n	The "new line" character, hex **0A**.
\b	The "backspace" character, hex **08**.
\t	The "tab" character, hex **09**.
\q	The double quote character **"**, hex **22**.
\p	A special "parameter" placeholder, used for actions (section 8.15).
****	The **** (backslash) character, hex **5C**.
\0ooo	The octal representation of any character, where "**ooo**" is from one to

three octal digits.

7.6 DOUBLE-QUOTED STRINGS

In some commands, the user must configure some parameters with text that may contain multiple words (sets of characters separated by spaces). For those parameters, the input text must always be enclosed between double quotes: **"**. For example: **majtext="* some words *"**.

There may be some cases where the user wishes to enter **"** (double quote) as a character within one of the text parameters. Since the double-quote character cannot be entered between quotes, the **\q** special sequence can be used, if necessary, to represent **"** within the input text.

7.7 CONSOLE IDENTIFICATION

The application command set of the **Virtual Console** offers special powers for addressing multiple consoles in each command. There are four main mechanisms for identifying consoles in a command:

- a single console name.
- a number or a range of numbers, e.g. **224-400**.
- identifying parameters with the form <*name*>=<*value*>, e.g. **name=[a-c]*** or **dest=132.64.37.100**.
- the current selection, denoted **[ ]**; the current selection is constructed using the "**select**" command.

Certain combinations are also allowed, such as a range or a selection along with identifying parameters, e.g. **[ ] name=*tn***.

7.7.1 CONSOLE NAMES AND RANGES

There are 768 console ports, numbered **1** to **768** that may be configured for management by the **Virtual Console**. Each console may be configured with a **name**. In any of the commands that address consoles that have already been configured, the consoles to be addressed in those commands may be identified by entering a console's name or number, or a range of numbers, e.g. **2-10**. The word ***all*** is a synonym for the range **1-768**.

7.7.2 IDENTIFYING PARAMETERS

A console or group of consoles may be addressed by matching with *identifying parameters*. The identifying parameters fall into two classes: configured properties and current status. The parameters for *configured properties* are:

- **name=<Pattern>**

- **comment="<Pattern>"**
- **dest=<IP address>**
- **dport=<TCP Port>**
- **admin=<Administrator ID>**
- **mon=<Monitor Group>**

There is only one *current status* parameter:

- **alarm=<Alarm Level>**. The alarm level can be **major**, **minor**, **info**, or **any**.

Identifying parameters work by filtering the given range or the selection in a command so that only consoles matching the parameters are used in the command. For example, the command **list 100-200 name=∗UMI∗** would list only those consoles in the range 100-200 that contain the characters "**UMI**" in their names. The **alarm** parameter matches consoles that have alarms at or above the given value of the alarm level. The distinction between *configured properties* and *current status* parameters is that *current status* is a moving target. In the command set, the only difference between *configured properties* and *current status* parameters is that only *configured properties* may be used to construct a *selection*.

Examples:

dest=123.21.56.78	This would identify consoles with the given IP address. This could rapidly select all instances on a DT-6061 or all endpoints on a DT-4000, etc.
comment="∗Seattle DT-4000∗"	This would identify consoles that had been configured with the string Seattle DT-4000 as part of the comment.
alarm=info	This would identify consoles that have alarms logged at the info level or higher.
admin=george*	This would identify all consoles that are *allowed* for use by all administrators whose login begins with ***george***
mon=1-2	This would identify all the consoles listed in monitor groups **1** and **2**.

When multiple identifying parameters are given, the resulting console list is the intersection of the consoles that match the parameters, i.e. the set of consoles that matches *all* of the parameters.

The design of console names and comments might well be guided by the convenience of identifying sets of consoles by pattern matching.

7.7.3 COMBINING IDENTIFIERS

The current selection or a range may be combined with *identifying parameters* to create a refined list. When *identifying parameters* are given without a range or **[ ]**, the implied range is **1-768**, and the *identifying parameters* are matched against all configured consoles.

Examples:

23-63 name=AZ*	this identifies those consoles in the range from 23 to 63 whose names begin with **AZ**
[ ] dest=123.45.67.89	this identifies consoles in the current selection that have the indicated IP address.
dest=111.22.34.56 dport=10001	this identifies the console with the given IP address and TCP port.

7.7.4 CONSOLE SELECTION

A *selection* of consoles may be constructed using the **select** command (described in section 8.12). The selection can contain any set of configured consoles. The selection may be invoked in any command by entering **[ ]** for the console identification. For example, if the selection contains consoles 5, 10, 23-50, and 101, the command **list [ ]** would print the *list* output for consoles 5, 10, 23-50 and 101. Each logged in administrator has a current *selection* (initially empty); the *selection* is temporary and is not stored in the database. Invocation of the current selection is only by the **[ ]** notation; it is never invoked by default.

As a shortcut in the **vfy**, **rs**, **rm**, and **dc** commands, which normally require entering a type of object as the first parameter, entering the first parameter **[ ]** implies the **con** object, so that, e.g., **vfy [ ]** may be used as a shortcut for **vfy con [ ]**.

8 IP-VCON OA&M COMMAND SET

8.1 LOGIN

Syntax: login PASSWD=<password>

The login command is used to allow access to the other configuration commands.

The login command is only visible when the application is in the *logged out* (i.e. secure) mode. The unit enters this mode whenever a *logout* command is issued or when the Telnet to the application instance OA&M TCP port is interrupted for any reason.

The password consists of up to seven alphanumeric characters. Special characters are not allowed. The password given on the command line is not echo-suppressed. If the **login** command is given with no parameters, the user is prompted for the password, and in this case, the password *is* echo-suppressed.

The default password is “initial”.

8.2 LOGOUT

Syntax: logout

The logout command is only allowed if the console user is logged *in*. It uses no arguments. It will set the console to the logged *out* mode.

8.3 CHANGE PASSWORD - CHGPASS

Syntax: chgpass OLD=<old> NEW=<new> CONFIRM=<new>

The **chgpass** command is used to change a user password on the system console. The command is only allowed if the user is logged *in*.

All three parameters must be given on the same line as the command. None of those entries are echo-suppressed.

If the current password is valid, and the two entries for the new password match, the password is changed to the new value.

8.4 HELP

Syntax: help | ? [Command]

The **help** command is always visible. The help command displays the currently allowed commands for the mode that the unit is currently entered. The alternate command for help is a question mark.

8.5 LABEL

Syntax: label [word (no spaces) | none]

The **label** command is used to give the command console a unique prompt of up to 31 characters or less. The command is visible only when logged into the IP-VCON OA&M console. If the **label** command is invoked without arguments, the current configuration of the label is displayed. If the argument to the **label** command is the word **none**, any current label is erased. If the argument to the **label** command is any other word, that word becomes the application console prompt label.

8.6 VERSION - VER

Syntax: ver

The **version** command is only visible when the application is *logged in*. The command has no arguments. It displays the current software and database revisions of the application.

8.7 OA&M SESSION TIMER CONFIGURATION - TIMEOUT

Syntax: timeout [OFF | <Number of Seconds>]
Default: OFF
Range: 15-255 seconds

The **timeout** command configures the OA&M session timer. When configured, the session timer will automatically **logout** a user from the IP-VCON OA&M console after the specified period expires with no user input.

8.8 APPLICATION COMMENTS - COMMENT

Syntax: comment [L1="Comment Text #1"]
[L2="Comment Text #2"]
[L3="Comment Text #3"]

The **comment** command is only visible when the application is *logged in*. The command is used to configure a comment field applicable to this instance of this application. The comments are visible on a *verify vcon* command. The use of this command is not required if no comments are desired.

The **L1, L2,** and **L3** comment fields allow the **Virtual Console** administrator to appropriately label this application should multiple installations be used. The text is a maximum of 60 characters in length for each field.

8.9 BANNER MESSAGE - BANNER

Syntax: banner [L<#>="System Banner Line <#>"]

The **banner** command is only visible when the application is *logged in*. The command is used to configure a system message banner that is displayed to the administrators upon connection. There are currently eight (8) lines of banner. Each line may be configured independently of each other with its own tag (i.e. L1, L2, …, L8). Each banner line may contain a maximum of 60 characters of text including spaces. The banner lines are visible on a *verify vcon* command. The use of this command is not required if no banner is desired.

8.10 ADMINISTRATOR SETUP - ADM

Syntax: adm <Admin #> [cnt=<Repetition #>]
[login=<login id>] [passwd=<password>
[allow=[+|-]<Range> | <Select Notation> | NONE]
[monitor=[+|-]<Range> | NONE]
[timeout=<#Minutes> | NONE]
[comment="Text" | NONE]
[katime=<#Minutes> | NONE]
[[+|-]ONETIME]
[like=<Admin #>
[DELETE]

The **adm** command is only visible when the application is *logged in*. The command is used to configure an administrator ID. The **cnt** allows a repetition of the same command on multiple administrators. The **<Admin #>** is a designator for an administrator ID in the range of 1 through the number of administrators allowed (see features description). The **<login id>** is up to 8 characters (case-sensitive) and may contain letters and digits. The **<password>** is also up to 8 characters (case-sensitive) and may contain letters and digits. The **allow** parameter will make the administrator authorized to use this **<Range>** of *console* endpoints. If an endpoint is to not be allowed, a *minus* is placed before it. Multiple administrators may be *allow*ed for the same consoles.

The **allow** parameter assigns the list of consoles that can be administered by this administrator. The value given with **allow** may be a range of console numbers or one of the selection notations: **[ ]**, **[+]**, or **[-]** (see section 8.12), or **none**. The **+** or **–** preceding the value indicates adding or deleting the indicated consoles to/from the allowed set of consoles. The **[+]** and **[-]** notation is equivalent to **+[ ]** or **–[ ]**, adding or deleting the current *console selection* to/from the list. The **allow** parameter is cumulative. For example, suppose administrator #6 were already configured with a login and password. It is now desired to allow the administrator full access to all *console* connections except #357. Then the command would be:

adm 6 allow=1-768 allow=-357

The **monitor** parameter (a.k.a **mon**) allows this administrator to access the consoles in the given monitor groups using the limited command sets configured for the monitor groups. Monitor group numbers range from **1** to **64.** The **monitor** parameter is cumulative. For example, if this administrator is already configured with access to monitor groups 1 and 2, group 3 may be added and group 2 removed as follows:

adm 12 mon=+3 mon=-2

Both the **allow** and the **monitor** parameters also accept the keyword **none** to remove all consoles or monitor groups from the administrator.

An administrator definition does not automatically allow the login procedure to take place. The administrator ID (in this example #6) needs to be placed into service using the **restore** command before a login is allowed.

The **timeout** tag allows an idle timeout for a particular administrator ID. If not desired, the value may be set to **NONE**. By doing so, no timeout is implemented for the administrator ID. By setting the **<#Minutes>**, an idle administrator session is disconnected after that number of minutes has elapsed.

The **comment** field allows the **Virtual Console** administrator to appropriately label this administrator should it become necessary. The text field is a maximum of 60 bytes in length.

The **ONETIME** option allows the administrator id to be used exactly once. After the one usage, the password is changed to an invalid value. In order to re-use the ID, this command must again be used to set the password. An optional minus may precede this option to remove the designation.

The **KATIME** option allows a TCP keep-alive to be sent to periodically check the status of the connection. The recommended interval is 2 minutes. The default is that no keep-alive transactions are to be sent.

If **like=<Admin #>** is given, the configuration values (except **login** & **passwd**) of an existing administrator are used to initialize a new one. The **login** and **passwd** values must be set explicitly.

The **DELETE** option shall remove the administrator id, and all of its associated configuration, from the IP-VCON application database.

8.11 CONSOLE CONNECTION SETUP - CON

Syntax: con <Console #> [cnt=<Repetition #>]
[dest=<Destination IP Address>]
[dport=<Destination TCP Port>]
[incr=<TCP Port Increment on Repetition>]
[type=< DT | UNK >]
[eop=<DT | "string" | NONE>]
[login=<Login ID>] [passwd=<Password>]
[alarm=< STD | <Alarm #> | NONE >]
[log=< major | minor | info | any | NONE >]
[majact=< script# > | NONE]
[minact=< script# > | NONE]
[infoact=< script# > | NONE]
[name=<Mnemonic Name>]
[comment="Text" | NONE]
[katime=< <#Minutes> |SYS| NONE >]
[like=<Console #>]
[DELETE]

The **con** command is only visible when the application is *logged in*. The command is used to configure a console connection. The **<Console #>** is a designator for the *console* in the range of **1** through **768**, the number of *consoles* supported (see features description). The **cnt** allows a repetition of the same command on multiple consoles. The **con** command also accepts a **<Range>** or the current selection, denoted by **[ ]**, in place of **<Console #>** or **cnt=<Repetition #>** for configuring multiple consoles with a single command.

The **dest=<Destination IP Address>** specifies the IP address on which the console resides. If the console is physical, this is the IP address of the DT-4000 or similar device on which the console is connected.

The **dport=<Destination TCP Port>** gives the TCP port on the device that defines the console of that device. If the console is physically connected, this is the TCP port of the physical port on which the connection is made.

The **incr=<TCP Port Increment on Repetition>** allows for multiple consoles residing on the same IP device to be configured simultaneously. This is the amount by which the **<Destination TCP Port>** is incremented on each successive console.

The **type** indicates the type of procedure to be performed on the console upon connection. As of the time of this writing, there are only two types. A **type** of **DT** will attempt automatic login using the standard procedure defined in all of the *Data*

Transport devices (e.g. UMI, DT-6061, etc.), using the value provided with the **passwd** tag. A **type** of **UNK** will skip the automatic login procedure.

The **passwd** tag is currently used only for consoles with a **type** of **DT**. The value of the **passwd** tag is used in the standard *Data Transport* device login sequence. The **login** tag is not currently used. **login** and **passwd** tags are not currently required with a **type** of **UNK.**

The **eop** is the "end of prompt" sequence. It can be a standard sequence used by *Data Transport* devices (**DT)**, it can be a custom string double quoted, or there could be no sequence **(NONE)**. The custom string accepts special character notation, as described in section 7.5.

The **alarm** tag indicates the alarm decoding to be used. An **alarm** tag of **STD** uses the Major, Minor, and Info alarm decoding found in many devices. Otherwise, an **<Alarm #>** may be specified to declare custom message matching. The **alarm** command (section 8.14) allows specific text patterns to be treated as alarms. Alarms qualify to be considered for *alarm actions*, as described below (and in sections 8.14 and 8.15). An **alarm** tag of **none** means that *no* alarms will be detected for this console.

The **log** tag specifies the severity level for retaining alarms detected by the **Virtual Console**. The possible values for **log** are: **any**, **info**, **minor**, or **major**. Alarms at or above the severity level specified by **log** are retained (a value of **any** retains all output) for later viewing by the **dlog** command. Only the most recent 30 or so lines are retained.

The **majact** tag indicates an action script to be executed when a major alarm is detected on this console. If no action is to be taken, it can be set to **NONE**. Otherwise, it is set to a script number entered with the **action** command (section 8.15).

The **minact** tag indicates an action script to be executed when a minor alarm is detected on this console. If no action is to be taken, it can be set to **NONE**. Otherwise, it is set to a script number entered with the **action** command.

The **infoact** tag indicates an action script to be executed when an information alarm is detected on this console. If no action is to be taken, it can be set to **NONE**. Otherwise, it is set to a script number entered with the **action** command.

The **name** tag indicates the *mnemonic name* used to identify the *console* by an administrator. The *mnemonic name* may contain letters or numbers. The **name** tag is up to 31 characters in length. When multiple consoles are configured in a single command, the presence of "**%d**" or "**%**<*num*>*"* within the **name** allows for an automatic sequence number to be generated as part of the name. With "**%d**", the

number begins with '**1**', and with "**%**<*num*>", the number begins with the given <*num*>; the number increases by one for each console.

The **comment** field allows the **Virtual Console** administrator to appropriately label this *console* should it become necessary. The text field is a maximum of 60 bytes in length. The presence of "**%d**" or "**%**<num>" within the comment allows for an automatic sequence number to be generated, as with the **name** tag.

The **name** and **comment** can be useful for console identification in other commands (section 7.7) or for constructing alarm actions (section 8.15). Careful study of those two features may be important in designing a strategy for configuring console names and comments.

The **katime** parameter is used to configure the keep-alive interval between the IP-VCON and a console. There are two keep-alive types. If a value for **<#minutes>** is specified, the older keep-alive algorithm is invoked. The range of **minutes** is 1-250. This algorithm uses the application level telnet protocol that requires the other end to deal with it. The application either detects no response after a prearranged interval or receives an error when trying to read or write on a socket that has disconnected.

If the value **sys** is specified for **katime**, a low-level TCP protocol, not available at the user/application level, is invoked. The operating system initiates the protocol after notification by the application when the application establishes a connection, i.e. opens a socket. The operating system executes the low-level protocol and will terminate a socket when no response is received to the **katime** probes. A connection must be idle for 200 seconds (The value is not settable by the user or application), at which time the operating system will retry sending a probe 8 more times at 12 seconds intervals before it terminates the socket with an RST+ACK. The application will react as though the other side did a normal TCP disconnect. The keep-alive is transparent to the console endpoint as it is hidden in the underlying protocol layers. The connection keep-alive testing may be disabled by setting the value of the **katime** option to **NONE**.

If **like=<Console #>** is entered, configuration values of an existing console are used as the starting point for configuring a new console. The **dport** parameter is *not* taken from the existing console, and the **name** parameter is modified.

The **DELETE** option shall remove the console entry, and all of its associated configuration, from the IP-VCON application database.

Example: con 101-130 name=AZ-DT-6061-%d dest=123.45.67.89 dport=10001 incr=1 comment="arizona DT-6061 instance %d console"

The above example would quickly configure uniquely named consoles with unique comments for all the instances on a DT-6061 at the given IP address.

8.12 CONSOLE SELECTION - SELECT

Syntax: select [[+] | [-]] <Range> | <Mnemonic Name> | [ ]
<Identifying Parameters>
select [[+] | [-]] <Range> <Range> …

The **select** command is only visible when the application is *logged in*. The command is used to establish a *console selection* that may be used in subsequent commands by entering **[ ]** as a console identifier. Following the optional **[+]** or **[-]**, there are two forms of the select parameter list: **<Identifying Parameters>** (section 7.7.2) or a **<Range>** (section 7.7.1) list. With a leading **[ ]**, **[+]** or **[-]**, the current selection is modified, as described below. Unless one of **[ ]**, **[+]** or **[-]** is given, the previous selection is discarded and a new selection is developed from the remaining parameters.

Using the **<Identifying Parameters>** form, the user may begin with a **<Range>** or **[ ]** as an initial list and follow it with parameters of the form **<name>=<value>**, taken from the set of *configured properties* described in section 7.7.2. The *configured properties* serve to limit the initial **<Range>** or *selection* to just those consoles matching all the given *configured properties*. Using **select all** selects all configured consoles.

Using the **<Range>** list form, the user enters a sequence of numbers and ranges of numbers, e.g. **4 10-15 20-22**, etc. The result is the combination of all consoles within the given ranges.

Using either form it is possible to add to (**[+]**) or subtract from (**[-]**) the previously established selection, by starting with the **[+]** or **[-]** notation. For example, if the selection were previously created with **select name=∗UMI∗**, the selection could be augmented by: **select [+] name=∗DT-4000∗**, yielding a selection that contains all consoles whose names include either **UMI** or **DT-4000**.

With no parameters, **select** shows the list of console numbers that are currently selected.

8.13 MONITOR GROUP SETUP - MON

Syntax: mon <Monitor Group Range>
[comment=”Text”]
[cmds=[+|-]”<Command List>”
[con=[+|-]<Range>|<Select Notation>|NONE]
[like=<Monitor Group #>]
[DELETE]

This configures a monitor group or a range of monitor groups. A monitor group is a list of consoles and commands that can be used to assign limited console access to an administrator. Consoles may belong to multiple monitor groups, and

administrators may be configured with access to multiple monitor groups. If an administrator has access to a console through two or more monitor groups, the administrator may use all the commands of said monitor groups on that console.

The monitor group numbers range from **1** to **64**. The **comment** parameter assigns the *text* as a label associated with this monitor group, up to 60 characters. The **like** parameter assigns the values of a different monitor group to this monitor group. The **delete** parameter erases configuration for this monitor group.

The **con** parameter, a.k.a. **console**, assigns a list of consoles to this monitor group. The value given with **con** may be a range of console numbers or one of the selection notations: **[ ]**, **[+]**, or **[-]** (see section 8.12), or **none**. The **+** or **–** preceding the value indicates adding or deleting the indicated consoles to/from the monitor group. The **con=[+]** and **con=[-]** notation is equivalent to **con=+[ ]** or **con=–[ ]**, adding or deleting the current *console selection* to/from the list. The **con** parameter is cumulative, similar to the **allow** parameter of the **adm** command (Section 8.10).

The **cmds** parameter assigns or edits a list of commands and parameters that may be sent to the consoles in the group by administrators assigned to this monitor group. Up to **20** commands may be assigned to each monitor group, occupying a total of up to **99** characters, including commas. Unless precautions are taken (described below) *only non-prompting commands should be assigned to monitor groups,* because there is no special way to specify legal responses for prompts. The command list is a comma-separated list of command names with space-separated parameters (tokens), allowing pattern-matching notation for each token. For example:

mon 6 cmds=”vfy,ver,dc,dconn,dm,dmeas,help”

permits the listed commands to be sent along with any parameters entered by the administrator.

mon 7 cmds=”rs sam,rm sam” or **cmds=”r[ms] sam”**

permits the **rm** or **rs** commands to be sent when accompanied by the **sam** parameter. The more parameters that are configured with a command, the more restricted it is, since all the parameters must match before the command is allowed.

The **cmds** parameter may be used to edit the assigned command list by using **+** or **–** in front of the quoted string to add or delete the indicated commands to/from the list. Also, multiple **cmds** editing parameters may be given in one command. Thus:

mon 6 cmds=-”vfy” cmds=+“vfy sam”

deletes the more permissive form of the **vfy** command and adds the more restricted form **vfy sam**.

A special notation using **$** is used to distinguish between a permissive form of a command and a restricted form. The permissive form allows the user to enter the

configured command and parameters followed by any other parameters. The restricted form uses a final parameter of **$** to limit the user's command to the parameters described in the configuration. For example, configuring **cmds=**"**dlog**" allows the **dlog** command to be sent with any parameters, including **dlog clr**. If parameters are not to be allowed, configuring **cmds="dlog $"** allows **dlog** to be sent only when it is given with no parameters. Note: avoid including a more permissive form along with a restricted form (e.g. **cmds="dlog,dlog prompt=* $"**), because the permissive form makes the restriction ineffective. The following corrected example allows only the two intended restricted forms of **dlog**, one with no parameters, and one with just the **prompt=**<anything> parameter:

mon 8 cmds="dlog $,dlog prompt=* $"

Console commands with prompts: There may be special circumstances where prompting commands can be allowed in monitor groups. Difficulties arise from the fact that IP-VCON cannot distinguish between the prompt issued by the console's command interpreter and prompts issued by individual commands. Since IP-VCON processes all user input to monitored consoles through the command pattern-matching process, all allowed responses to all prompts must be configured as if they were commands. It may be possible to configure prompted commands for certain consoles with attention to the principle that no pattern configured for a response to prompting should match any command that should not be allowed on the console. As an example of a safe configuration, if a command emits a series of prompts that request numeric input, the monitor group could be configured to permit that command and also permit input of numbers, because no number is a command. Other than numbers or certain canned responses that don't match console commands, it is unlikely to be able to configure a monitor group for use with prompting commands.

Special Input Sequences: When an administrator is connected to a monitored console, the only input forwarded to the console is that which matches a configured command. Normally, empty lines are *not* sent, nor is the *DEL* (hex 7f) character, because these can be disruptive to other users. For special needs, two special sequences are defined for configuring a monitor group command list so that empty lines or the *DEL* character may be sent to the console. To allow empty lines to be sent, add the string **$** to the command list, e.g. **cmd=+"$"**. To allow the *DEL* character to be sent, add the string **$DEL**, e.g. **cmd=+"$DEL"**. Both of these sequences may be useful where prompted commands have been configured. The empty string can be used to respond to prompts that have defaults, and the *DEL* character can be used to kill a command that has prompted for input that the user doesn't have permission to enter.

Example: The following sequence configures commands and prompts for a partially prompted usage of the **rm** or **rs** commands for *sam* ports:

cmds="r[ms] sam port,[0-9],[0-9][0-9],[0-9][0-9][0-9][0-9],$del"

This configuration allows the **rm** and **rs** commands for *sam* ports to be typed with or without the numbers for module, board, and port. If the numbers are not entered as part of the command, they can be entered as responses to prompts. The *DEL* key is explicitly allowed here, allowing the user to break out of a prompting sequence (or to interrupt any command).

8.14 DEFINING CUSTOM ALARMS - ALARM

Syntax: alarm <Alarm #> [majtext="Maj Alarm Text Pattern" | NONE]
[mintext="Min Alarm Text Pattern" | NONE]
[infotext="Info Alarm Text Pattern" | NONE]
[like=<Alarm #>]

The **alarm** command is only visible when the application is *logged in*. The command is used to configure a custom interface for decoding alarms. Each **<Alarm #>** defines three types of alarms, a major category, a minor category, and an informational category. The **<Alarm #>** can range from **1** to **16**. When a console is configured with an **<Alarm** #>, all data received from that console is matched against the three patterns. If one of the patterns is matched, the corresponding alarm is declared. For each of the parameters, a value of **NONE** without quotes will declare that the field should be not used in matching. In addition to the pattern-matching characters described in section 7.4, the special characters in section 7.5 may be used. If **like=<Alarm #>** is given, an existing alarm configuration is used as a model to initialize a new one.

8.15 DEFINING ACTION SEQUENCES - ACTION

Syntax: action <Action #> [match="<Pattern>"]
[send="Text to send to Console"]
[param="<Parameter specifiers>"
[console=<Console #>]
[trap="<TRAP INFO LIST>" | NONE]
[like=<Action #>]

The **action** command is only visible when the application is *logged in*. The command is used to configure an action to be performed when a particular alarm is recognized. An action consists of a command to be sent to a console or a trap to be sent to an SNMP manager. The **<Action #>** ranges from **1** to **32**. If **like=<Action #>** is given, an existing action is used as a model to initialize a new one.

In order to be effective, an action number must be assigned to one or more consoles (as a **majact**, **minact**, or **infoact** value as described in section 8.11). The alarms for those consoles (as configured by the **con** and **alarm** commands) then qualify to be tested to trigger the actions. The **match** parameter, which may include the special

characters described in 7.5, is pattern-matched against qualifying alarm text to determine whether the action shall be performed. There are two types of actions currently defined, **send**, and **trap**.

The **send** option is to send a text string to the *console*. The text string represents a command to be performed on the alarming device or on a different device. This can be used for immediate recovery from a condition reported in device console output. The **send** text must be enclosed between **"** (double quote) characters, and may use the special character sequences described in section 7.5. The **send** text may be directed to a different console from the one triggering the action by configuring the **console** parameter with the number of a different console. Especially for this purpose, it is possible to modify the **send** text at the time of the action to include some information about the console triggering the action.

One or more parameters may be embedded in the **send** text by using the special character sequence **\p** as a placeholder. The data actually inserted at those placeholders is determined by *parameter specifiers* configured by the **param** option. The *parameter specifiers* isolate portions of the **name** or **comment** of the console that has triggered the action. These substrings are then substituted for **\p** in the **send** text at the time of the action to form a command to send to the alternate console. The *parameter specifiers* in **param** are composed of a letter '**N**' or '**C**', followed by a character to be used as a separator, then two more strings terminated by the separator character. The letter '**N**' chooses the **name** field, and '**C**' chooses the comment field. The two strings are matched in the **name** or **comment**, and the characters found between the strings form the resulting parameter.

Example: Suppose a DT-6061 system console is managed by IP_VCON as console *10*, and 20 application *instance* consoles on the same DT-6061 are managed as consoles *11* through *30*. The objective is to configure an alarm action for the *instance* consoles that will be carried out on the DT-6061 system console. Configuration would be as follows:

The consoles are configured with **alarm=std**.

Each **comment** of the 20 *instance* consoles is configured with text as:

phoenix4 DT-6061 instance 1 console
phoenix4 DT-6061 instance 2 console
…

The above configuration is easily accomplished using:

con 11-30 comment="phoenix4 DT-6061 instance %1 console"

Each instance console is configured with major action script **6**, e.g.:

con 11-30 maject=6

and **action 6** is configured with:

match="∗something bad∗"
send="restart \p"
param="C/instance / console/"
console=10

Any major alarm containing the string "something bad" from one of the 20 instance consoles triggers the following process. As an example, let it be console *29*, which corresponds to the console for *instance 19* on the DT-6061.

1. Action 6 matches "**something bad**" in a major alarm for console 29.
2. The substitution process for the string "**restart \p**" uses the specifier **C/instance / console/** to isolate the instance number, **19**, lying between "instance " and " console" in console 29's comment.
3. The instance number is then substituted for **\p** in the **restart** command.
4. The resulting command **restart 19** is sent to console **10**, the DT-6061 system console.

Variations:

- The **send** text may contain multiple **\p** sequences and the **param** string can contain corresponding *parameter specifiers* (concatenated, no spaces). The substitutions are carried out in the order they appear. (If an action is configured with a number of **\p** sequences different from the number of *parameter specifiers*, the user is warned.)
- Any character may be used as the delimiter (following '**N**' or '**C**'). The '**/**' character is suggested, but if the *before* or *after* string were to contain '**/**', it would be necessary to choose a different delimiter character, e.g. the '**-**' in **N-be/fore-after-**.
- An empty "before" string matches the beginning of the comment or name; an empty "after" string matches the end of the comment or name. So **N///** matches the entire name.

For the consoles of devices (or applications) that are under the control of another device, the design of console names and comments might take into consideration the ability described above to extract parameters from the name or comment for use in sending alarm actions to the controlling device.
The **trap** option configures an IP address and trap community name to send an SNMP trap to a trap manager. The **<TRAP INFO LIST>** is a quoted string containing the IP address of the trap manager and the trap community name, separated by a comma. Example: **trap="123.45.6.7,privateTraps"**.

IP-VCON sends an enterprise-specific trap with specific trap type of 1, containing two variables in the IP-VCON name space:

Object Identifiers	
IP-VCON	enterprises.3791.3.7.2
<console name>	IP-VCON.1
<alarm text>	IP-VCON.2

Example: Using the consoles configured as above to recognize standard alarms and to use **action 6** as the major action, SNMP traps are enabled by adding a **trap** configuration to **action 6**, such as **trap="192.168.9.88,privateTraps"**. Then when a major alarm containing "something bad" is detected on any of those consoles, an enterprise-specific trap is sent to the trap manager at **192.168.9.88** with the community name **privateTraps**. The trap contains the name of the console and the full text of the console output that IP-VCON determined to be an alarm. (A MIB file is available from Datatek (**ip_vcon.mib**) which can be compiled with other MIB files and will enable the trap information to be displayed by an SNMP manager in a more readable form.)

Alarm handling summary: automatic alarm handling is configured in the parameters of these three or four configuration commands:

1. In the **con** command, the **alarm** type (**none**, **std**, or a custom alarm number declared by **<Alarm #>**), the **log** level, and the **majact**, **minact**, and **infoact** of each console. Optionally, the **name** or **comment** options may be relevant.
2. If the **alarm** type of the console is a custom alarm, the **majtext**, **mintext**, or **infotext** of the selected custom alarm configured by the **alarm** command.
3. In the **action** command, the **match**, **send, param, console**, and **trap** parameters of an action selected by the **majact**, **minact**, or **infoact** of the console.
4. In the **logger** command, the monitor groups and administrators selected for forwarding alarms to logger channels.

Output generated by each device console is continuously scanned for alarm content (as configured by items 1 and 2). If console output qualifies as an alarm and meets or exceeds the **log** severity level, it is transmitted to each active *logger* connection that has been configured (by the **logger** command) to receive alarms for this console. It is also saved inside IP_VCON for later viewing by the **dlog** command, possibly displacing a previously saved alarm. Regardless of log severity, the corresponding **action** is determined, and the alarm text is further scanned to attempt a match with the alarm action **match** pattern (as configured by item 3). If the match is made, the corresponding **send** or **trap** action is performed.

If the **send** option is configured, the text in the **send** option is examined for the special **\p** sequence. For each **\p** sequence, a *parameter specifier* from the **param** option is used to locate a substring of the **name** or **comment** of the triggering console, and that substring is inserted in place of **\p**. The resulting text is then sent to the triggering console or to a different console, if specified by the action's **console** parameter.

If the **trap** option is configured, an enterprise-specific trap PDU is sent to the trap port at the configured IP address with the configured community string. The trap contains values for two variables: the console name and the alarm text.

8.16 LOGGER CHANNEL SETUP - LOGGER

Syntax: logger <Range> [monitor=[+|-]<Range> | NONE]
[admin=[+|-]<Range> | NONE]
[like=<Logger Channel#>] [DELETE]

The **logger** (or **log**) command is only visible when the application is *logged in*. The command is used to configure a logger channel (in the range **1** to **32**) to forward the alarms for a subset of the consoles. The set of consoles for a **logger channel** is taken from the *allowed* and *monitored* console sets configured for sets of **administrator** and **monitor group** objects, respectively. When a logger channel is configured with an administrator index, all qualifying alarms for that administrator's *allowed* consoles are sent to any current connection to that logger channel. Likewise, when a logger channel is configured with a monitor group index, all qualifying alarms for that monitor group's consoles are sent to that logger channel. Any logger channel may be configured with any set of administrators and monitor groups.

The **admin=** and **monitor=** parameters may be entered multiple times, and have a cumulative effect. A **<Range>** preceded by "**–**" is deleted from the configuration. The term **none** may be used in place of a **<Range>**. The **like=** option duplicates another logger channel. The **delete** option erases the configuration for the logger channel.

Logger channels **1** to **32** are accessed by making a **telnet** call to TCP ports 30001 to 30032, respectively. In addition to the configurable logger channels **1** to **32**, TCP port 30000 (a.k.a. channel **0**) carries qualifying alarms for **all** consoles.

8.17 VERIFY OF CONFIGURATION - VFY

Syntax: vfy ADM <Range> | login=<Pattern>
vfy CON <Console Identification> [prompt=<Yes/No>]
vfy MON <Range> | comment="<Pattern>"
vfy ALARM <Range> | ACTION <Range> |
VCON | ALL]

vfy LOG <Range>

The **vfy** command is only visible when the application is *logged in*. The command is used to verify the configuration for all of the objects of the **Virtual Console**. Range checking is done on each object per the feature specification.

The general parameters are displayed with the object of **VCON**. The administrator configurations are displayed with the object of **ADM**. The *console* connection configurations are displayed with an object of **CON**. The configurations identifying custom alarm indications are displayed with an object of **ALARM**. Actions to be made on alarms are displayed with an object of **ACTION**. The reports for the **ALARM** and **ACTION** objects give derived lists of the console numbers that refer to the given objects. The display for **MON** shows the console lists and allowed commands, and includes a derived list of administrators assigned to monitor groups. The display for **LOG** shows the configured lists of administrators and monitor groups assigned to logger channels, and shows derived lists of consoles whose qualifying alarms will be carried on those logger channels.

When using **vfy con**, the **<Console Identification>** parameters may be given, as described in section 7.7.2. As a shortcut, **vfy [ ]** is the same as **vfy con [ ]**, which verifies the currently selected consoles. When using **vfy adm**, the **login=** parameter (with a match pattern) may be given instead of a range. When using **vfy mon**, the parameter **comment="<Pattern>"** can be used to identify monitor groups in place of a monitor group number range.

The **vfy con** report can be rather lengthy for a large number of consoles. When the number of consoles in the report is greater than *4*, the **Virtual Console** pauses the report after each console, shows a prompt of **[NEXT]**, and waits for the user to enter any character before proceeding to the next report. The prompting behavior is suppressed by entering the **prompt=no** parameter on the command line. The command in progress can be cancelled by the **del** key.

8.18 PLACING COMPONENTS IN SERVICE - RS

Syntax: rs < ADM <Range> | CON <Console Identification> >

The restore command is only visible when the application is *logged in*. The command is used to place a console connection, or an administrator ID into service. In the case of a console connection, a TCP connection is not attempted until the component is placed into service.

The object to be restored to service and its modifier are both required. When the object is **ADM**, one or more administrator IDs are to be placed into service. Only those IDs that have a login & password defined as not being NULL will actually be placed into service. When the object is **CON**, one or more console connections are

placed into service, as identified by console identification parameters described in section 7.7.2. Only those console connections that have an IP address and destination TCP port will actually be placed into service.

As a shortcut, **rs [ ]** places the currently *selected* console connections into service.

8.19 TAKING COMPONENTS OUT OF SERVICE - RM

Syntax: rm < ADM <Range> | CON <Console Identification> >

The remove command is only visible when the application is *logged in*. The command is used to take console connections, or administrator IDs out of service. When a console connection is taken out of service, the TCP connection made to that console is disconnected. When an administrator ID is taken out of service, a login for that ID will no longer be allowed. If an administrator is currently established, the **Virtual Console** will display a suitable message and disconnect the administrator.

8.20 LIST CONSOLES - LIST

Syntax: list <Console Identification>

The **list** command is only visible when the application is *logged in*. The command is used to show a one-line report of the name, ID#, alarm condition indicator, service state, and IP address of each *console* matched by the given **<Console Identification>** (as described in section 7.7). The alarm condition indicator in the report shows as follows: "**" means the console has reported a major alarm, "* " means a minor alarm.

8.21 DISPLAYING ALARM LOGS - DLOG

Syntax: dlog <Console Identification> [clear=yes] [prompt=no]

The **dlog** command is used to display the alarm log history for one or more *console* connections, matched by the given parameters. (See section 7.7 for **<Console Identification>**.) The alarm history shows the alarm level and a timestamp relative to the current time when displayed. Up to 30 lines of alarms are kept in the alarm history, based on alarm level and age. When the optional **clear=yes** parameter is given, each console's alarm history is cleared after being printed. By default, the **Virtual Console** pauses after displaying each alarm history and waits for the user to type something. Typing **del** stops the command. Typing any other key continues to the next console alarm history. When the optional **prompt=no** parameter is given, the histories are displayed with no pause.

Example: **dlog name=*UMI* clear=yes alarm=major** This example displays and clears all logged alarms for consoles that contain **UMI** in their name and have **major** alarms in their alarm histories.

8.22 CLEARING THE ALARM LOGS - CLOG

Syntax: clog <Console Identification>

The **clog** command is used to clear the alarm history for one or more *console* connections, matched by the given parameters. (See section 7.7 for **<Console Identification>**.)

8.23 DISPLAYING CURRENT CONNECTIONS - DC

Syntax: dc [adm [<Range>] [login=<Pattern>]]
dc con [<Console Identification>]
dc logger [<Range>]

The **dconn** command is used to display the current connections into the IP-VCON application. This includes all of the console, administrator, and logger connections. The command reports the connection peer for each active connection, or an error code if IP-VCON has tried, but failed to make the connection.

Please note that the command does not *require* any arguments. When supplied with parameters, **dc adm** limits the report to the **<Range>** of administrator IDs, from **1** to **64**, or to the administrator logins matching the **<Pattern**>. With no parameters, **dc adm** lists all administrators currently logged in as well as administrator connections that have not completed login. The **dc con** command accepts **<Console Identification>** parameters. As a shortcut, **dc [ ]** means **dc con [ ]** (display console connections for the current selection).

8.24 DISPLAY OF MEASUREMENTS - DMEAS

Syntax: dmeas [ALL | ADM <Range> |
CON < Console Identification> |
LOGGER <Range>]

The **dmeas** (or **dm**) command is only visible when the application is logged in. The command is used to display the current measurements on any of the interfaces.

The **dmeas** command may display the measurements for a single interface, or all of the interfaces. Where the value of **<Range>** is specified, the identifier may be a single numeric (e.g. the number '3'), a numeric range (e.g. '1-3'), or the value '**ALL**' to indicate the entire allowed numeric range.

The **CON <Console Identification>** option will display the measurement information for the *console* connections to the IP-VCON that match the given *console identification* parameters.

The **ADM <Range>** option will display the measurement information for the IP-VCON administrator connections. The values in the **<Range>** are 1 through 27 inclusive.

The **LOGGER <Range>** option will display the measurement information for the logger connections. The values in the **<Range>** are 1 through 32 inclusive.

When measurements are displayed via the **dmeas** command, and more than a single entity has been specified; only non-zero entries are actually displayed.

8.25 CLEAR MEASUREMENTS - CLR

Syntax: clr

The measurements displayed with the **dmeas** command are aggregated until cleared. The **clear** command will set all measurements to zero. The command has no arguments.

8.26 SNOOPING ON TRAFFIC - SNOOP - [NOT IMPLEMENTED YET]

Syntax: snoop [OFF | ADM <#> | CON <#>]

The IP-VCON application has a diagnostic ability to snoop on any of the interfaces that carry user data. This is done with the **snoop** command. All output is directed to the OA&M connection.

If the command is invoked with no arguments, it produces a report of all active snooper configurations.

If the command is invoked with the **OFF** option, all of the active snooper configurations are disabled.

If the command is invoked with the **CON <#>** option, the *console* interface specified is snooped. The **<#>** is in the range of 1 to 768. Output is displayed on the OA&M session. Please note that this could be extensive for a moderately busy *console* connection.

If the command is invoked with the **ADM <#>** option, the administrator interface specified is snooped. The **<#>** is in the range of 1 to 27 inclusive and corresponds to administrative connection numbers as displayed on the display connection report.

The primary intent of this command is in the aid of creating automatic operation of the IP-VCON application.

9 IP-VCON MEASUREMENTS

This section itemizes the measurements available using the *display measurements* (**dmeas**) command.

The base measurements are always displayed, and the error and exception counters are only displayed if nonzero.

The measurements are per console, and per administrator.

The measurements available are as follows:

Measurement Description	**Type**
Number of Bytes Received	Base
Number of Bytes Transmitted	Base
Number of Bytes Dropped on Transmit	Base

10 ADMINISTRATOR CONSOLE INTERFACE

The administrator interface is presented on any of up to **27** simultaneous connections to the administrator port. Upon connection, a user is prompted to login with an ID and password. If the telnet session remains idle for two minutes without completing the login and password, the session is disconnected. Successfully entering ID and password grants access to the administrator console interface.

Once logged in, the administrator will receive a prompt as follows:

< <ALM_COND>: <Console_Name> > *EOP*

The **<ALM_COND>** displays the highest current alarm condition for any of the device consoles that are allowed to be accessed by this administrator ID. The **<ALM_COND>** has three possible values. A space (“ “) indicates that there are no alarms, or that the highest severity alarm for all of the associated *console* connections was informational. A single asterisk (“∗”) indicates that the highest severity alarm for all of the associated *console* connections was a minor alarm. A double asterisk (“∗∗”) indicates that the highest severity alarm for all of the associated *console* connections was a major alarm.

The **<Console_Name>** is the mnemonic name of the currently selected device console. When no console is selected, the **<Console_Name>** will indicate “—ANY—“ (without the double quotes).

The **EOP** (hex **19**) is the standard end of prompt sequence for a *Data Transport* device.

The commands on the administrator interface begin with the **“!”** character to differentiate them from user input intended for the console to which the administrator is patched through.

10.1 INPUT CONVENTIONS

When an administrator is not connected to a console by the **!set** command, input is processed by IP-VCON, using the typing conventions described in section 7. The administrator is expected to enter IP-VCON commands, all of which begin with ‘**!**’.

When the administrator is connected to an **allowed** console, IP-VCON watches for input line boundaries (i.e. the *enter* key), and checks for ‘**!**’ at the start of each line to allow switching to IP-VCON command mode. Otherwise, input is passed through, without processing, to the connected console, which processes input according to its own rules, usually echoing each character. Output from the console, including input echo from the console, is sent by IP-VCON back to the administrator (and to all administrators connected to the same console). IP-VCON watches for prompts from

the console (as configured by the **con** command) and substitutes the IP-VCON prompt that is seen by administrators.

When an administrator is connected by **!set** to a console that is not **allowed** but is available in restricted mode through one of its monitor groups, input is processed and echoed by IP-VCON until the line is terminated (i.e. with the *enter* key). When the administrator enters a command that meets the monitor group's restrictions, the command is then sent to the console. The target console processes that input according to its own rules, and usually echoes that input, which is sent back to the administrator. Thus the administrator will see each command twice: once as it is typed and once again when the target console echoes it. When the user enters a command that doesn't meet the monitor group's restrictions, an error message is given and nothing is sent to the target console.

When the administrator begins a line with '**!**', the rest of the input line is processed by IP-VCON, and the completed line is processed as an IP-VCON command. Nothing is sent to the connected console.

10.2 ADMINISTRATOR COMMAND SET

The available commands are as follows:

10.2.1 COMMAND	10.2.2 DESCRIPTION
!help [<!command>] !?	The "!help" command displays usage for the commands in this table, or for just the given <!command>
!set <Mnemonic Name> !set <Console #>	The "!set" command directs the administrator console to communicate with the <Mnemonic Name> console. While this condition exists, communication to the administrator shall arrive only from the <Mnemonic Name> console. Allowed consoles for an administrator may update the <ALM_COND> during this time. A second "!set" command clears the prior association before making a new one. If the chosen console is restricted, the administrator is notified.
!clr	The "!clr" command eliminates the association to the <Mnemonic Name> in the previous "!set" command for this administrator.
!select <Parameters>	The "!select" command is exactly like the OA&M "select" command (see section 8.12.),

10.2.1 COMMAND	10.2.2 DESCRIPTION
	with the exception that only the consoles associated (allowed or monitored) with this administrator will be selected.
!list <Console Identification>	The "!list" command will list the Mnemonic Names, ID, and other selected information about the consoles associated with the administrator ID. The report is limited to consoles that match the parameters. (See section 7.7 for <Console Identification>.)
!dlog <Console Identification> [clear=yes] [prompt=no]	The "!dlog" command displays the current alarm log for the console devices selected. (See section 7.7 for <Console Identification>.) If no <Console Identification> is given, !dlog operates on the currently "set" console.
!clog <Console Identification>	The "!clog" command clears the current alarm log for the console devices selected. (See section 7.7 for <Console Identification>.) If no <Console Identification> is given, !clog operates on the currently "set" console.
!dc <Console Identification>	The "!dc" command displays a list of connections to the console specified. If no <Console Identification> is given, all of the console connections associated with this administrator ID are displayed. The display shows the numeric ID, Mnemonic ID, and the connection status for every console associated with this administrator ID. When <Mnemonic Name> is used, the console connection matching is presented.
!mon [<Console #>]	The "!mon" command with no arguments lists the consoles that this administrator may access using restrictions configured in the monitor groups assigned to this administrator. With a console number, or if this administrator is currently connected to a restricted console, the list of commands available to this administrator on the restricted console is reported.

10.2.1 COMMAND	10.2.2 DESCRIPTION
!send <Console Identification> cmd="<Text>"	The "!send" command sends the <Text> defined by cmd= as input to the consoles matching the <Console Identification> (See section 7.7 for <Console Identification>.).The <text> may use any of the special characters except \p. (See section 7.5) The consoles are selected one at a time: the input is sent, and output is returned to the administrator; a prompt is indicated with “[START …]”, "[NEXT …]", or “[DONE …]”. When the administrator types a key at the prompt, the next matching console is selected except when the “[DONE…]” prompt appears. This indicates that the !send command is completed and the prompt returns to the normal command-mode prompt. Alternately, If the '!' or del key is typed, the !send command stops and returns to the normal command-mode prompt.
!vfy <Console Identification>	The “!vfy” command displays the details of the console connection. This command is useful for showing the comment field associated with a console. If no <Console Identification> is given, the currently "set" console details are displayed.
!who	The “!who” command displays all of the administrator IDs logged into the Virtual Console application. The administrator connection that issued the command will be flagged with an asterisk. The report will also show the console connection that was “set” for each administrator.
!logout	The !logout command logs out the administrator and brings up the Login: prompt.
!chgpass	The “!chgpass” command allows the administrator to change their current password to a new value. The current password will not automatically age, nor is it checked against prior usage history by this command.

Notes:

- An administrative login may be used for multiple simultaneous sessions in IP-VCON.
- When an administrator is connected to a console via the “!set” command, output from that console is displayed. If the console is *allowed* for this administrator, the administrator's input is sent to that console. If the console is *restricted*, input is collected first by IP-VCON until the whole line has been entered.
- When more than one administrator is connected to the same console, each administrator shall each see the output from that console. That output will include the *echo* of the peer administrators' input to the console.
- If an administrator connects via "!set" to a console to which one or more administrators are already connected, each of those administrators will be informed of the new connection.
- If multiple administrators are connected to a single console via the “!set” command, and an administrator disconnects via the “!clr” command; then the remaining administrators shall receive a message indicating the disconnection.
- If connectivity to a monitored connection is lost while an administrator is actively connected, a message shall be produced by IP-VCON to the administrator. At that point, the selection of the console is disconnected, and the prompt will indicate “—ANY—“ (without the double quotes).
- If an automated command, such as an alarm **action** is sent to a device console while one or more administrators are connected to it, the administrators shall first receive a message from IP-VCON indicating that an automated command is being sent. Then they will see the device's response upon receiving the command.

11 LOGGER INTERFACE

There are 32 TCP telnet connections to the IP-VCON available to be used by systems that log activity. Multiple connections are provided so that there is no requirement of the user's logging system to distribute the activity further. All device console alarms meeting or exceeding the console's configured alarm log severity level are forwarded to all (or some) of the logger connections. What qualifies as an alarm from a device console is described in sections 8.14 and 8.11. In addition to alarms from device consoles, there are a few alarms from IP-VCON that refer to actions delivered to consoles.

Connections to the logger interface may be made to any of the logger channels **0** through **32** (TCP ports 30000 through 30032). Logger channel **0** forwards all qualifying alarms. Logger channels (**1** through **32**) must be individually configured (using the **logger** command) to forward alarms for subsets of the consoles. Multiple connections may be made to any logger channel, as long as the total number of connections does not exceed **32**.

A connection to the logger interface may be made on any system, UNIX or PC based, by using the **telnet** command. For example, suppose that the IP-VCON resided on a DT-6061 at IP address 192.148.7.200. A connection to the logger interface from a UNIX (or LINUX) system would be as follows:

telnet 192.148.7.200 30000 | tee –i logfile

A connection from a PC would be:

telnet 192.148.7.200 30000

It will be necessary on the PC to manually start the file logging on the telnet client by pulling down the “terminal” menu and then “Start Logging”.

The IP-VCON logger interfaces do not accept any input. If any data is presented on these connections to the IP-VCON, it is silently discarded.

12 HARDWARE WARRANTY

The warranty period for the DT-6XXX hardware on which this application runs shall be ninety (90) days from the date of shipment of the hardware from Datatek Applications, Inc. Replacements and repairs are guaranteed for the longer of the remaining original warranty period or 30 days whichever is longer.

13 END-USER LICENSE AGREEMENT FOR SOFTWARE

This License Agreement ("License") is a legal contract between you and the manufacturer ("Manufacturer") of the system ("HARDWARE") with which you acquired software product(s) identified above ("SOFTWARE"). The SOFTWARE may include printed materials that accompany the SOFTWARE. Any software provided along with the SOFTWARE that is associated with a separate end-user license agreement is licensed to you under the terms of that license agreement. By installing, copying, downloading, accessing or otherwise using the SOFTWARE, you agree to be bound by the terms of this LICENSE. If you do not agree to the terms of this LICENSE, Manufacturer is unwilling to license the SOFTWARE to you. In such event, you may not use or copy the SOFTWARE, and you should promptly contact Manufacturer for instructions on return of the unused product(s) for a refund.

13.1 SOFTWARE LICENSE

You may only install and use one copy of the SOFTWARE on the HARDWARE (unless otherwise licensed by Manufacturer). The SOFTWARE may not be installed, accessed, displayed, run, shared or used concurrently on or from different computers, including a workstation, terminal or other digital electronic device (“Devices”). Notwithstanding the foregoing and except as otherwise provided below, any number of Devices may access or otherwise utilize the services of the SOFTWARE. You may not reverse engineer, decompile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. The SOFTWARE is licensed as a single product. Its component parts may not be separated for use on more than one HARDWARE. The SOFTWARE is licensed with the HARDWARE as a single integrated product. The SOFTWARE may only be used with the HARDWARE as set forth in this LICENSE. You may not rent, lease or lend the SOFTWARE in any manner. You may permanently transfer all of your rights under this LICENSE only as part of a permanent sale or transfer of the HARDWARE, provided you retain no copies, you transfer all of the SOFTWARE (including all component parts, the media and printed materials, any upgrades, this LICENSE and, if applicable, the Certificate(s) of Authenticity), and the recipient agrees to the terms of this LICENSE. If the SOFTWARE is an upgrade, any transfer must also include all prior versions of the SOFTWARE. Without prejudice to any other rights, Manufacturer may terminate this LICENSE if you fail to comply with the terms and conditions of this LICENSE. In such event, you must destroy all copies of the SOFTWARE and all of its component parts.

13.2 INTELLECTUAL PROPERTY RIGHTS

The SOFTWARE is licensed, not sold to you. The SOFTWARE is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. You may not copy the printed materials accompanying the SOFTWARE. All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE is the property of the respective content owner and may be protected by applicable copyright or other intellectual property

laws and treaties. This LICENSE grants you no rights to use such content. All rights not expressly granted under this LICENSE are reserved Manufacturer and its licensors (if any).

13.3 SOFTWARE SUPPORT

SOFTWARE support is not provided by Manufacturer, or its affiliates or subsidiaries separate from the HARDWARE. For SOFTWARE support, please contact your supplier of the HARDWARE. SOFTWARE support is limited to the warranty period stated below unless either a separate contract has been consummated between you and the manufacturer or the manufacturer has agreed in writing at the time of purchase by you of the software to an extension of the warranty. Should you have any questions concerning this LICENSE, or if you desire to contact Manufacturer for any other reason, please refer to the address provided in the documentation for the HARDWARE.

13.4 EXPORT RESTRICTIONS

You agree that you will not export or re-export the SOFTWARE to any country, person, or entity subject to U.S. export restrictions. You specifically agree not to export or re-export the SOFTWARE: (i) to any country to which the U.S. has embargoed or restricted the export of goods or services, which as of March 1998 include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria, or to any national of any such country, wherever located, who intends to transmit or transport the products back to such country; (ii) to any person or entity who you know or have reason to know will utilize the SOFTWARE or portion thereof in the design, development or production of nuclear, chemical or biological weapons; or (iii) to any person or entity who has been prohibited from participating in U.S. export transactions by any federal agency of the U.S. government.

13.5 LIMITED WARRANTY

Manufacturer warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of shipment from Datatek Applications, Inc. Software support is limited to the hours of 9 AM to 5 PM ET Monday through Friday excluding Datatek-observed holidays. Other coverage and extended warranty may be purchased at additional cost. Any implied warranties on the SOFTWARE are limited to ninety (90) days. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Manufacturer's and its suppliers' entire liability and your exclusive remedy shall be, at Manufacturer's option, either (a) return of the price paid, or (b) repair or replacement of the SOFTWARE that does not meet this Limited Warranty and which is returned to Manufacturer with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

13.6 NO OTHER WARRANTIES

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, MANUFACTURER AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT, WITH REGARD TO THE SOFTWARE AND THE ACCOMPANYING WRITTEN MATERIALS. THIS LIMITED WARRANTY GIVES YOU

SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

13.7 SPECIAL PROVISIONS

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and HARDWARE Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial HARDWARE Software-Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is Datatek Applications, Inc., 379 Campus Drive, Suite 100, Somerset, NJ 08873.

If you acquired the SOFTWARE in the United States of America, this Software License are governed by the laws of the State of New Jersey, excluding its choice of laws provisions. If you acquired the SOFTWARE outside the United States of America, local law may apply. This LICENSE constitutes the entire understanding and agreement between you and the Manufacturer in relation to the SOFTWARE and supercedes any and all prior or other communications, statements, documents, agreements or other information between the parties with respect to the subject matter hereof.

14 LIMITATION OF LIABILITY

To the maximum extent permitted by applicable law, in no event shall Manufacturer or its suppliers be liable for any damages whatsoever (including without limitation, special, incidental, consequential, or indirect damages for personal injury, loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if Manufacturer has been advised of the possibility of such damages. In any case, Manufacturer's and its suppliers' entire liability under any provision of this License shall be limited to the amount actually paid by you for the SOFTWARE and/or the HARDWARE. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.