Registry Key Assignments

As of JANOS v1.6.1

INTEG Process Group, Inc.
2919 E. Hardies Rd, First Floor
Gibsonia, PA 15044

www.integpg.com

JNIORsales@integpg.com

PH (724) 933-9350
FAX (724) 443-3553

Contents

1. About the Registry

The JNIOR Automation Network Operating System (JANOS) and its applications can be configured to suit your needs. Configuration involves choices, and those settings may be stored in a variety of ways. JANOS relies on its Registry system for all operating system configuration. The Registry can also be easily used by applications and web pages for the storage of custom configuration settings. The Registry may also be used to store and share data dynamically.

The JANOS Registry is non-volatile. Its content remains in place even when power is removed. Information is stored as a set of *name-value* pairs. Each entry is referenced by a unique *Registry Key* or name. Each entry contains information formatted as a character string representing its value. The content is available to JANOS directly, to external applications and web pages through protocols, and to local application programs through the Java runtime library.

JANOS maintains a backup copy of the Registry in the `/flash/jnior.ini` file. When content in the Registry is changed this INI file will later be updated to reflect the changes. This backup file is automatically generated and should not be overwritten or modified. JANOS performs this *backup* every several minutes as needed. The `/flash/jnior.ini` file may be read and saved as a representation of Registry content. This INI file will reflect changes only after the backup occurs. The backup is autmoatically performed on reboot.

A copy of the `/flash/jnior.ini` may be editted and saved under a different filename. This then may be injested using the REGISTRY -I command as means a performing bulk configuration.

All actions are logged to the `/jniorsys.log` file providing an audit trail for configuration management.

2. Accessing the Registry

Configuration is likely best performed using the Dynamic Configuration Pages (DCP). Once the JNIOR is connected to the network the browser can be used to open the DCP website using the unit's IP address. The JNIOR is configured by default to open the DCP. An administrator login is required to access the registry. The default username is 'jnior' and the default password for that account is also 'jnior'.

The 'Configuration' tab of the DCP provides an organized form-oriented means for adjusting the various configuration settings. In this section you are provided with over a dozen different categories. These settings affect both how the JNIOR operates and how the DCP displays information. Not all of the valid Registry Keys are presented within this section but only the most common and appropriate settings for each category. Certain advanced settings will need to be made manually by a different means. The unit's 'Network' configuration, for instance, may be easily adjusted in this section.

The DCP also provides a 'Registry' tab. This section shows the raw content of the Registry Keys in a form similar to a file explorer. Only those keys with values are shown. You can add, remove or edit any Registry Key using this tab. Here you are required to know specifically what key or keys you want to change. This is most appropriate for advanced administrators. This does provide a graphical user interface for Registry Key management.

The 'Console' tab in the DCP provides access to the JANOS Command Line Console. This is the same command line facility that can be accessed using a Terminal or Telnet application to open the standard Telnet port (port 23) over the network. Even in the absence of a network connection you may open the console by making a serial connection (115.2 Kbaud, 8 data, 1 stop, no parity) to the RS-232/COM port located to the right of the Ethernet/LAN connection on the JNIOR. The COM port is a 3-wire connection using a male (or pin) type DB-9 connector. Pin 2 is data transmitted (Tx) from JNIOR (data source); Pin 3 is data to be received by the JNIOR (connected computer is the source); And, pin 5 is ground (GND). A standard USB-to-serial converter presenting a female (or socket) type DB-9 connector may be used directly. Once you have logged into the console you may use the REGISTRY command and many other commands to configure and administer your JNIOR. Type 'help' there for the list of commands. You can type 'help' followed by a space and a command to get details on the syntax for the specified command.

If you are working with a Windows based PC you may download and install the INTEG Support Tool. The installer is available from our website at http://www.integpg.com. Once the Support Tool is opened the 'Beacon' tab will display all of the JNIORs located on the current network segment. If you right-click any JNIOR the resulting context menu will provide access to the unit's web page (DCP), a Telnet application, and many other useful functions. The Support Tool also provides a 'Registry Editor' tab through which you can add, remove and edit content as needed for the selected JNIOR(s).

2.1. The Registry Editor

A Registry Editor is available for use in an interactive mode. The editor is started using the `registry` command entered at the command prompt. The command prompt is available for example through a Telnet (or COM RS-232 serial) connection. This `registry` command is available only to those logged in as administrator.

The Registry has a structure much like a file system. There are no “files” per say but there are *keys* at various levels each containing one or more *values*. Analogous to a file path, the “Registry Key” specifically defines a Registry entry and the ‘/’ separator breaks the Registry down into *sections* (like folders).

The Registry editor displays one section at a time and provides for options to navigate through the Registry. In a Telnet session (or through a direct serial connection) you must login as an administrator. Use the command “registry” or its shorter alias “reg” to enter the editor. Nothing else on the command line is required although the “help” option is available. User keystrokes are shown in **bold characters**.

```
JANOS /> help registry
registry [option]

Display and Modify JNIOR Registry settings.
 [-s]    Generate snapshot
 [-i]    Load specified INI file
Alias: reg

Command is available only to administrators.
Command line syntax:  registry [keyname [= [newcontent]]]
Valid at prompt:
   ##  - number from menu displays item, <previous> moves up
  ".." - moves up, same as <previous>
  "."  - redisplays current section
  "*"  - selects all entries for removal
  "<section>/.." - opens that section
  Type any key to Add/Edit that key (relative to current section)
  Type key prefixed by '/' to specify it from the Root
  Immediate blank line Exits the editor
```

When you start the Registry Editor the very "top" of the Registry or the *root* is displayed. You will see that the root contains the valid System Parameters as well as paths to one or more *sections*.

```
JNIOR /> registry
JNIOR Registry Editor
Copyright (C) 2005 INTEG process group, inc.  All Rights Reserved.
See Help for more information.


Content of /..
   1  $BootTime = Mon Apr 11 13:28:53 GMT 2005
   2  $SerialNumber = 4904004
   3  $Version = 2.01.219
   4  Device/..
   5  Email/..
   6  Events/..
   7  IO/..
   8  IpConfig/..
   9  JniorServer/..
  10  ModbusServer/..
  11  <exit>

Key (or ## selection) to Add/Edit/Remove?
```

At this prompt you may either enter the Registry Key string or a selection by number from the displayed menu. One feature of the very first prompt after the display of the menu is that a blank line will exit the Registry Editor and return you to the command prompt. You can always get out of the Registry Editor by hitting repeated Enter keystrokes. At the <Root> menu you might also use selection number 11 (in this particular Registry) to exit. The various Registry sections are shown listed by there initial *node* followed by "/.." to indicate that there is more below. Enter a numeric selection to descend into any section.

Key (or ## selection) to Add/Edit/Remove? **6**

```
Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = enabled
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

Here we have elected by entering #6 to view the `Events` section. In this particular JNIOR we see that the Power-On Reset and Alarm event features have been enabled (see *Configuration Examples*). We can see that there is the option shown presently as #3 to return to the previous menu and effectively moving you back up a level in the structure. Entering the number 3 at this point would in fact redisplay the Root menu as previous shown.

Say that we wish to disable the Power-On Reset (or Boot) Notification. We can simply select that Registry key and change its value. Here's how that would go:

```
Key (or ## selection) to Add/Edit/Remove? 2
  Events/OnBoot = enabled
[C]hange, [E]dit, [R]emove? c
New Value(s)? disabled
  Events/OnBoot = disabled
Accept [Y/N]? y

Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = disabled
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

Now in the background JNIOR is quietly updating the jnior.ini file to reflect this change. When JNIOR is rebooted (using the `reboot` command for instance) it will come up and load the Registry from the jnior.ini file. When JNIOR gets to the point where it must decide whether or not to send the Boot Notification it will see that it is not desired. No notification will be sent.

Note that the responses to the questions are case-insensitive. The lower case 'c' is as good as 'C' to elect to [C]hange a value. The [E]dit option differs only in that it displays the original content first allowing you to backspace and change it.

Since by default the Power-On Reset or Boot Notification is disabled, this Registry key being actually set to "disabled" is not really needed. We could actually have simply deleted it and achieved the goal of stopping the email notification. This removal is easily done:

```
Key (or ## selection) to Add/Edit/Remove? 2
  Events/OnBoot = disabled
[C]hange, [E]dit, [R]emove? r
Delete Key [Y/N]? y

Content of Events/..
   1  OnAlarm = enabled
   2  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

The key has been removed from the Registry. Now if we really do want these notifications we can redefine the key. To do so we enter the desired key name "relative" to the current section. We only have to enter the key name at this point and not the entire string. Note that Registry keys <u>are</u> case-sensitive.

```
Key (or ## selection) to Add/Edit/Remove? OnBoot
  Events/OnBoot = <undefined>
Add Key [Y/N]? y
New Value(s)? enabled
  Events/OnBoot = enabled
Accept [Y/N]? y

Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = enabled
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

In this fashion we can also define new keys within new substructure. This example shows the creation of a key in a different section of the Registry below the currently displayed section.

```
Key (or ## selection) to Add/Edit/Remove? OnBoot/Email
  Events/OnBoot/Email = <undefined>
Add Key [Y/N]? y
New Value(s)? enabled, AdminEmail
  Events/OnBoot/Email = enabled, AdminEmail
Accept [Y/N]? y

Content of Events/OnBoot/..
   1  Email = enabled, AdminEmail
   2  <previous>

Key (or ## selection) to Add/Edit/Remove? ..

Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = enabled
   3  OnBoot/..
   4  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

So in this case we had wanted to create the Registry key `Events/OnBoot/Email` which can be used to reference a unique email definition that we will arbitrarily call "AdminEmail". Since we were already displaying the `Events` section we merely need to enter the balance of the key to create it.

Once the new structure has been defined the Registry Editor will display the new section. We see here something new that we can do at the prompt. If you enter two periods ".." the Registry editor will move up to the previous level. In the example above we would have achieved the same result if we have selected the <previous> option. Similarly you can use a single period "." to redisplay the current section. These have been defined analogous to their use in file system directory navigation.

One other special character is available and it provides the ability to remove an entire Registry section including all of the subsections contained therein. The "*" asterisk command is valid in any section except the root. It selects all of the keys and subsections listed. It then confirms your intention to delete them. An affirmative response will remove the entire section of the Registry. Obviously this is very powerful and care should be exercised in its use. It is very helpful in cleaning house.

Look at the current content of the `Events` section. One thing to note is that a key name (`OnBoot` in this case) can specify both a value and further substructure.

One further example demonstrates the removal of a Registry section. To remove an entire section you must separately delete each key within the section. If there is a lot of this to do then perhaps external editing of the jnior.ini file would be more expeditious. The following removes the section just entered. Note that the Registry Editor removes the whole section and displays the first level above that with remaining content.

```
Key (or ## selection) to Add/Edit/Remove? 3

Content of Events/OnBoot/..
   1  Email = enabled, AdminEmail
   2  <previous>

Key (or ## selection) to Add/Edit/Remove? 1
  Events/OnBoot/Email = enabled, AdminEmail
[C]hange, [E]dit, [R]emove? r
Delete Key [Y/N]? y

Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = enabled
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

Finally it is worth noting that we can specify a new key relative to the Root no matter where we are in the Registry structure. This is done by prefixing the key with an additional '/' character. While the leading '/' is not normally used in referencing Registry keys it is appropriate here to insure that the entry is attached to the correct part of the structure.

```
Key (or ## selection) to Add/Edit/Remove? /Email/AdminEmail/ToAddress
  Email/AdminEmail/ToAddress = <undefined>
Add Key [Y/N]? y
New Value(s)? jdoe@integpg.com
  Email/AdminEmail/ToAddress = jdoe@integpg.com
Accept [Y/N]? n

Key (or ## selection) to Add/Edit/Remove? .

Content of Events/..
   1  OnAlarm = enabled
   2  OnBoot = enabled
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

And in this case we changed our mind. Had the key been created the Registry Editor would have displayed the related section. Instead we declined to "Accept" the new key and so we were prompted for additional action. The example used the single period to redisplay the original section.

Navigation about the Registry can be handled by moving down through the levels to different sections by selecting them from the content menu and moving back up using the <previous> selection or the ".." entry. You may also enter the desired section as if you were entering a key by using a trailing pair of periods. The Registry Editor will move to the section nearest the key or section that you have entered. For example:

```
Content of JniorServer/..
   1  Login = disabled
   2  Port = 9200
   3  <previous>

Key (or ## selection) to Add/Edit/Remove? /IO/Inputs/din1..

Content of IO/Inputs/din1/..
   1  Alarm1/..
   2  Count/..
   3  CountDisplay = enabled
   4  Desc = Generator
   5  <previous>

Key (or ## selection) to Add/Edit/Remove? Count/..

Content of IO/Inputs/din1/Count/..
   1  Alarm1 = enabled
   2  Alarm2 = enabled
   3  Limit1 = 5
   4  Limit2 = 10
   5  <previous>

Key (or ## selection) to Add/Edit/Remove? /Events/OnBoot/Email..

Content of Events/OnBoot/..
   1  Email = enabled, Boot
   2  <previous>

Key (or ## selection) to Add/Edit/Remove? /IO/Inputs/din12/WhoKnows..

Content of IO/Inputs/..
   1  din1/..
   2  din2/..
   3  <previous>

Key (or ## selection) to Add/Edit/Remove?
```

As can be seen from the last entry you do not have to specify a valid key or section. The Registry Editor will move to the nearest valid section.

The Registry Editor is best learned by doing. It is intended to provide access to the JNIOR's configuration through a minimal connection that does not support cursor movement, screen formatting, or even a mouse.

3. Command Line Usage

You may use the `registry` command at the command line prompt to view, change and remove individual Registry Keys. The general syntax is as follows where the `[]` indicate optional parameters.

```
registry [keyname [= [newcontent]]]
```

The `registry` command alone enters the interactive editing session as previously described. If the command is to be used on the command line to view, change or remove any Registry Key a valid `keyname` parameter must be specified. In order to change or remove a Registry Key the '=' must be supplied.

3.1. Querying a Registry Key

The following command retrieves a registry key value if one is defined.

```
JNIOR /> registry Device/Test
Device/Test is not defined.

JNIOR /> registry JniorServer/Port
JniorServer/Port = 9200

JNIOR /> reg $Version
$Version = v0.6.0-b1.4
```

3.2. Setting/Changing a Registry Key

A new value either defining a new Registry Key or changing the value of an existing Registry Key may be supplied at the command line. If the new value is to contain spaces you must enclose the value with double quotes. The following are some examples:

```
JNIOR /> registry Device/Test = TestingNewValue
JNIOR /> registry Device/Test
Device/Test = TestingNewValue

JNIOR /> registry Device/Test = "Testing New Value"
JNIOR /> registry Device/Test
Device/Test = Testing New Value

JNIOR /> reg Device/Test = "Parameter1, Parameter2"
JNIOR /> reg Device/Test
Device/Test = Parameter1, Parameter2
```

3.3. Removing a Registry Key

Finally you may remove a Registry Key by omitting any new value. The following demonstrates the usage:

```
JNIOR /> registry Device/Test =
JNIOR /> registry Device/Test
Device/Test is not defined.
```

Note that Registry Keys are case-sensitive and care must be used to correctly enter the key. A registry Key that may be correctly spelled but that does not contain the exact case for each character will not be recognized properly. These command line shortcuts are for those advanced users who are very familiar with the proper Registry Key formatting.

4. INI File Format

A copy of the Registry content is maintained in the `/flash/jnior.ini` file.

The INI file format is standard. It uses "section" headings enclosed by a pair of '[' and ']' brackets. Each section can contain any number of keys assigned values following the '=' sign. A key may have any number of values each being listed after the '=' and separated by a ',' comma.

Comments may appear in the INI file. Any text on any line following the ';' semicolon character is considered to be commentary (including the semicolon). In general, leading and trailing white space (one or more spaces or tabs) is ignored.

Any value that is intended to contain a ',' comma or ';' may be enclosed in "quotes". Thus the key defined here

```
Desc = "Relay, Main"
```

provides a description having just one value of *Relay, Main*. A value that is intended to contain a '"' double-quote character may be "escaped" by preceding it with a '\' back-slash.

4.1. Boolean Keys

Many keys are used to enable or disable some function. These have a logical on/off kind of purpose. JNIOR writes these values as "enabled" meaning on or "disabled" meaning off. If the INI file is edited externally the following content will all result in an enabled key:

```
enable
enabled
True
1
on
yes
```

Basically any content starting with the characters "enable" will qualify. All other content will be considered "Off" or disabled. These Boolean keys are case-insensitive.

5. Registry Key Syntax

A Registry *key* is a string specifying specific content much like a file specification. The key details a path to the configuration *value* within a desired Registry *section*. This is best illustrated by example. The following is a valid Registry Key:

```
IO/Inputs/din1/Desc
```

Again in a very analogous fashion to a file specification the part of the above key `IO/Inputs/din1` specifies a specific Registry section. In this case it is the section that configures Digital Input number 1. Similar to a file name `Desc` specifies a particular configuration setting or value defined in that section. In this case it is the description to be used with the particular input.

In the registry editor this key might appear as follows:

```
IO/Inputs/din1/Desc = Generator
```

This clearly defines the content for this key as the word “Generator”. Perhaps the JNIOR with such a configuration setting provides the status of a nearby generator through digital input 1. All Registry Values are themselves character strings. These strings are also used to represent Boolean conditions and numeric values at times. In those situations words like “enabled” and “disabled” could mean Boolean conditions True and False respectively or the string “0.0” might indicate the numeric value zero as might be appropriate for a configuration setting when a fractional value can be specified.

The corresponding entry in the INI file for the above Registry Key would appear as follows:

```
IO/Inputs/din1/Desc = Generator
```

In the Registry individual values are not always a single string as shown above but are in actuality arrays of strings. Multiple strings may be specified separated by commas. For example:

```
Email/ToAddress = jdoe@integpg.com, admin@integpg.com
```

This is a typical Registry key specifying a short list of email address to be used during Event Notification. There are actually two separate strings provided. The first being “jodoe@integpg.com” and the second “admin@integpg.com”. JNIOR in this case knows how to handle a list of address and will process the notification Emails appropriately sending one to each or the listed recipients.

An individual string in a Registry value may be optionally enclosed in double quotes. In particular this is important if the string is intended to include a comma character (or perhaps a semicolon) where it is important that JNIOR not be confused into thinking that separate strings are being listed (or an INI comment included). For instance a valid and somewhat formal output description might appear as:

```
IO/Outputs/rout1/Desc = “Relay, Main”
```

And this would appear in the jnior.ini file like this:

```
IO/Outputs/rout1/Desc = “Relay, Main”
```

One can see that there is a wide variety of ways in which to include detailed configuration information in this type of Registry system. This document describes those Registry Keys that are specifically used by JANOS. The Registry can be easily used to store unique and custom application specific configuration as well. Those details should be available to you in the application related documentation.

In the INI file every Registry value is located within a section (specified surrounded by the square brackets). Therefore it would appear that in every case when a Registry key is viewed through the registry editor (or

specified through one of the protocols) it would have to include at least one '/' separator as follows in this generic definition:

```
section/value = stringarray
```

This is for the most part true although there is a class of System Parameters which may be read from the Registry but that do not appear in the INI file. These are detailed in a following section of this document. These *System Parameters* have names that always begin with a '$' character are not contained within a specific section. System Parameters are intended to convey system related information such as the firmware version.

In fact, the '$' character may be used as a prefix with a Registry key name anywhere within the structure (any section). This indicates a *Dynamic Value* which will not appear in the INI file. One example of this occurs with the use of external sensors on the one-wire port. For example, a temperature sensor would report its value as follows:

```
OneWire/9600080010C84D10/$Value = 73.4, 23.0
```

This provides the current temperature of 73.4 degrees Fahrenheit or 23.0 degrees Celsius from the device on the Sensor Port whose address is "9600080010C84D10". This `$Value` Registry entry does not appear in the INI file and it does change dynamically as in this case it would track the temperature in the area of the sensor (updated about 4 times a second).

By the way, the `OneWire/Devices` Registry entry lists the addresses of all active sensors found on the Sensor Port. An application would query that key first to locate a specific device and using that information it may then read (and subscribe to) the related `$Value` key. If a subscription is used, the application would be notified whenever the temperature value changed.

6. Configuration Examples

6.1. Simple Email Notification on Power-On Reset

When a JNIOR is used in an application requiring continuous 24/7 operation instances of power loss or other interruption may be of some concern. You might want to be alerted when such events occur. The JNIOR may be easily configured to provide a notification by email when this happens. This feature can be enabled through the Registry.

```
Registry Events/OnBoot/Email = enabled
```

Using the Registry Editor make sure that the above key is defined. This enables the feature.

You must also make sure that the IP configuration of the JNIOR is correct. In particular JANOS must know how to find the email server (MailHost). Use the `ipconfig` command to specify a MailHost and the Sender's email address (FromAddress).

Note that some email servers may accept email only from valid members of the domain. The server may not recognize the FromAddress used by the JNIOR and reject the email posting. This error will be logged to the jniorsys.log file. You may need to supply a valid Username and Password. This must be done through the `ipconfig` command

With the IP configuration and the above keys set, a very simple email will be transmitted at the completion of the Boot (power-on reset process). Using additional Registry entries this email may be highly customized. You can specify multiple recipients and even attach files conveying required information.

6.2. Simple Alarm Notification

Certain situations may be set to trigger an Alarm condition. This might be when a digital input turns on, when an input counter reaches a certain count or when an input/output usage reaches a defined number of hours. When an Alarm condition occurs JANOS may be configured to send an email notification.

```
Registry Events/OnAlarm/Email = enabled
```

Using the Registry Editor make sure that the above key is defined. This enables the alarm notification. This feature may be applied to individual inputs and the operation is dependent on the presence of other Registry settings. The above enables the feature generically (all alarms).

You must also make sure that the IP configuration of the JNIOR is correct. In particular JNIOR must know how to find the email server (MailHost). Use the `ipconfig` command to specify the MailHost, a valid Sender's email address, and (if needed) the required login credentials.

One of the JNIOR digital inputs must then be configured to generate an alarm. With the following setting an alarm will be triggered when a voltage is applied to the Digital Input number 1.

```
Registry IO/Inputs/din1/Alarming = enabled
```

An alarm condition must exist for at least ½ second before it will trigger the notification. Input latching may be used to capture shorter pulses if those are intended to trigger the alarm. Once an alarm notification is sent the specific alarm condition must be removed for at least 5 minutes before any additional notification would be sent.

The resulting email messages may be highly customized on an input-by-input basis. These may be addressed to multiple recipients and include file attachments such as a jniorio.log file.

6.3. High Speed Relay Control Feature

The JNIOR provides for remote monitoring and it also supports eight relays which can be controlled remotely. Typically these are not applied in high speed or time critical applications. The JNIOR does support one feature that can be considered useful for high speed control.

Each of the relay outputs can be made to close upon reaching a defined trigger count. For instance if you want Relay 2 to close after counting 5,000 pulses on the associated Digital Input 2, you can define the following Registry Key:

```
IO/Outputs/rout2/$TriggerCount = 5000
```

Assuming that your application has cleared the counter on Digital Input 2, the relay will close after 5,000 input pulses. The relay closes immediately upon seeing the 5000^{th} low-to-high transition of the input. See the Registry Key definition for more information.

7. Dynamic Registry Keys

Any Registry Key or Path that begins with the '$' character is considered a *dynamic entry* which is temporary and present only for the current session. These keys are not saved to the .INI file and do not persist through a system reset or reboot.

Certain Dynamic Keys are defined during boot and therefore appear to always be present. These are typically System Parameters as documented in the following section.

Other Dynamic Keys are created by the system to provide access to frequently changing data. One example of this is the $HourMeter key providing Usage information such as the amount of time that a particular input is active.

Entire Registry Paths may be defined wherein all the keys are dynamic in nature. If the '$' begins any level of a Registry structure all of the keys contained therein are dynamic and will not be preserved. The individual entries themselves need not include the '$'.

Applications can use dynamic keys as a form of inter-process communications.

Only the system can define keys at the root level of the Registry. In all cases those defined are dynamic keys.

8. System Parameters

These '$' keys do not appear in the `/flash/jnior.ini` file and are OS related. These are special dynamic keys. Normally shortly after a Registry key is modified the system updates the `/flash/jnior.ini` file as a backup. The `/flash/jnior.ini` file is auto-generated and should not be overwritten or modified. When a key starting with '$' is created or modified it does not trigger a backup. The '$' key remains in the active Registry and for the Series 4 it is non-volatile. Applications can use '$' keys to provide external access to dynamic parameters.

The following are created by JANOS.

$BootTime

This returns a string representing the time according to the JNIOR clock at the completion of the latest power-up boot sequence.

$Model

This returns the product Model number. For example: "410"

$SerialNumber

This returns the product serial number as a String. For example: "612080001"

$Version

This returns the current Version string for the product release. For example: "v0.6.0-b1.4"

$LastNtpSuccess

This returns the last time the system clock was successfully updated from the network using the NTP protocol. For example: "Fri May 12 12:57:51 EDT 2017"

9. Registry Key Reference

The following are the current assignments as coded into the OS firmware. The Registry may contain other keys as assigned by the user or application developer.

9.1. General Device Configuration

The following keys apply to the JNIOR in general.

Device/Desc

The entry provides a textual description for this JNIOR. This might be displayed by DCP or applications as identification. It may also be included in e-mail notifications to identify the specific source. This description may be optionally added by the user.

9.2. Custom Timezones and Timezone Corrections

Any number of timezones may be defined or overwritten using keys in the `Timezones` Registry node.

Timezones/<name>

The clock subsystem is generally configured using the DATE command. JANOS defines a set of timezones for use in displaying local time. These timezones may or may not utilize Daylight Savings Time (DST).

As there are 24 hours on our clock one might expect that there needs to be only 24 timezones. This however is not the case as some areas offset their clocks by just 30 minutes. In addition, some areas utilize DST while others do not. The following is the default set of timezones. This can be displayed at any time using the DATE -T command.

```
Available timezones:
  UTC  (+0000) Universal_Coordinated      GMT  (+0000) Greenwich_Mean
  CAT  (-0100) Atlantic/Cape_Verde        MAT  (-0200) Atlantic/Mid
  BET  (-0300) Brazil/Brasilia            AGT  (-0300) Argentina/Buenos_Aires
  CNT  (-0330) America/St_Johns(1)        PRT  (-0400) America/Caracas
  EST  (-0500) America/New_York(1)        IET  (-0500) America/Indianapolis
  CST  (-0600) America/Chicago(1)         MST  (-0700) America/Denver(1)
  PST  (-0800) America/Los_Angeles(1)     AKST (-0900) America/Anchorage(1)
  HST  (-1000) Pacific/Honolulu           WST  (-1100) Pacific/Midway
  IDLW (-1200) Date_Line                  NST  (+1200) Pacific/Fiji
  NIT  (+1130) Norfolk Island             SST  (+1100) Russia/East
  LHT  (+1030) Lord Howe Island           AEST (+1000) Australia/Sydney(2)
  ACST (+0930) Australia/Adelaide(2)      JST  (+0900) Japan
  AWST (+0800) Australia/Perth            VST  (+0700) Mongolia/West
  CIT  (+0630) Burma                      ALMT (+0600) Asia/Alma-Ata
  IST  (+0530) India                      PLT  (+0500) Russia/Ural_Mountains
  AFT  (+0430) Afghanistan/Kabul          NET  (+0400) Russia/Moscow
  MET  (+0330) Iran/Tehran                EAT  (+0300) Asia/Riyadh
  EET  (+0200) Europe/Instanbul           ECT  (+0100) Europe/Paris

Note 1:
 DST begins at 02:00 on the Sun on or after Mar 8th.
 DST ends at 02:00 on the Sun on or after Nov 1st.
Note 2:
 DST begins at 02:00 on the Sun on or after Oct 1st.
 DST ends at 03:00 on the Sun on or after Apr 1st.
```

The rules for DST may change from time to time as governments alter their policies. This default list of timezones is likely incorrect for some areas across the globe. INTEG encourages users to let us know when a correction to the timezone tables might be appropriate. JANOS does provide a means by which you may define new timezones with or without DST rules. You may even correct an existing timezone. This is accomplished using one or more `Timezones/<name>` Registry entries.

The following key format is used to create a new timezone or overwrite an existing timezone. Note that timezones are identified by their standard abbreviation (AbbStd). The timezone for Eastern Standard Time is identified as "EST". In fact, since the default definition of this timezone includes a Daylight Savings Time (DST) rule, the DATE command can also select this timezone using the DST abbreviation (AbbDst) which is "EDT". The <name> portion of the key is arbitrary and serves only to differentiate the key from others.

```
reg Timezones/<name> = <offset>, <desc>, <AbbStd> [, <AbbDst>,
       <stMon>, <stDay>, <stDoW>, <stTime>,
       <endMon>, <endDay>, <endDoW>, <endTime>, <dstOfs>]
```

The definition or redefinition of a timezone is straight forward however the specification of the DST rule for the timezone can be a bit more confusing. The following 13 parameters may be required to properly specify a timezone.

<offset> The offset in minutes from UTC specified in military time format. For example -0500 subtracts 5 hours from GMT. The value 0630 adds six and a half hours to GMT.

<desc> Supplies a textual description of the timezone. For instance "Universal Coordinated" for UTC.

<AbbStd> Defines the standard abbreviation for the timezone. This is the identifier that is used with the date and time to specify the current timezone. It is used by the DATE command in setting the current timezone. If this matches an existing timezone the timezone definition will be overwritten. Otherwise a new timezone will be created. This allows you to correct the default timezones should the rules change.

The following 10 parameters are required only when specifying a DST rule.

<AbbDst> Defines an alternate abbreviation for the timezone. This is the identifier that is used with the date and time to specify the current timezone when Daylight Savings Time is in effect. It can be used by the DATE command in setting the current timezone.

<stMon> Specifies the starting month for DST. A 3-character abbrieviation is used: JAN, FEB, MAR, etc. This field is case-insensitive although uppercase is recommended by convention.

<stDay> Specifies the starting day of the month. This is a numeric value where 1 specifies the first day of the month. If it is necessary to specify a certain number of days before the end of the month, a negative value is specified. Since DST usually begins (and ends) on a specific day of the week, this value is used to select the correct part of the month for that day.

<stDoW> Specifies the day of the week on which DST starts. A 3-character abbreviation is used: SUN, MON, TUE, etc. This field is case-insensitive although uppercase is recommended by convention. This is the day of the week on or after the starting day. If it is necessary to specify the day of the week on or before the starting day, a negative sign may be prepended to the string (e.g. "-SUN").

<stTime> Specifies the starting time for DST in military time format. For example 0200 indicates 2 o'clock in the morning. This is point in time when the clocks are to be adjusted.

<endMon> Specifies the ending month for DST. A 3-character abbrieviation is used: JAN, FEB, MAR, etc. This field is case-insensitive although uppercase is recommended by convention.

<endDay> Specifies the ending day of the month. This is a numeric value where 1 specifies the first day of the month. If it is necessary to specify a certain number of days before the end of the month, a negative value is specified.

<endDoW> Specifies the day of the week on which DST ends. A 3-character abbreviation is used: SUN, MON, TUE, etc. This field is case-insensitive although uppercase is recommended by convention. This is the day of the week on or after the ending day. If it is necessary to specify the day of the week on or before the ending day, a negative sign may be prepended to the string (e.g. "-SUN").

<endTime> Specifies the ending time for DST in military time format. For example 0200 indicates 2 o'clock in the morning. This is point in time when the clocks are to be returned to standard time.

<dstOfs> This defines in minutes the adjustment that occurs when daylight savings time is in effect. Typically this value is 60 indicating that the clocks move ahead an hour for DST.

There are two forms to the key. The simple form requires only the first 3 fields. This defines a timezone that does not use DST. The full format requires 13 fields where the additional entries outline the use of DST in that timezone. The DST definition provides an additional abbreviation, specifies start and end timing, and defines the time offset.

For example, the following Registry command makes an entry that redefines the Eastern Timezone in the United States with an ego-centric description for those of us in Pittsburgh Pennsylvania.

```
reg Timezones/YinzerTime = "-0500, America/Pittsburgh, EST"
```

This would not be correct because the EST timezone observes Daylight Savings Time. We need to also include the rule.

```
reg Timezones/YinzerTime =
    "-0500, America/Pittsburgh, EST, EDT, MAR, 8, SUN, 200, NOV, 1, SUN, 200, 60"
```

And perhaps instead of redefining EST we would prefer to create our own timezone, the entry would change as follows. Note that only the AbbStd need be changed but we alter the AbbDst to be consistent.

```
reg Timezones/YinzerTime =
    "-0500, America/Pittsburgh, YST, YDT, MAR, 8, SUN, 200, NOV, 1, SUN, 200, 60"
```

These Registry keys are interpreted, and therefore take effect, on reboot. The new or modified timezones will appear in the table produced by the DATE -T command. The JNIOR may be switched to the new timezone which will remain in existence until the Registry key is removed or altered. Note that when time is reported to external systems, a custom timezone may not be recognized if its abbreviation is not common and known to the rest of the world.

A Timezone key will be ignored if it contains a syntax or value error. These errors will be reported to the system log file `jniorsys.log`.

9.3. Starting Programs on Boot

Programs may be initialized and run at boot. Such programs may be developed to handle custom capabilities that would not normally be part of JANOS. The development of these programs is beyond the scope of this document. Contact INTEG for further information.

Run/<program>

During the boot process JANOS will scan this section of the Registry and attempt to execute programs listed here. The <program> group name is arbitrary and each entry specifies the program file and any arguments that

may be associated with it. For example suppose that the following entry were defined:

```
Run/MyProgram = BootStuff.jar -timer 100
```

JANOS would execute the program `BootStuff.jar` as a background process provided the file can be located in the file system. The parameters shown would be passed to the program for its use. This key follows the command line syntax. The order in the which the keys are processed may vary and cannot be guaranteed.

Note that only the keys defined at this node will be executed and any substructure will be ignored. If it were useful the programmer could design the program to retrieve configuration from this location. For example the additional parameters in the above could be conveyed as follows:

```
Run/MyProgram = BootStuff.jar
Run/MyProgram/timer = 100
```

Again, the program `BootStuff.jar` would have to be designed to either retrieve either the arguments either from the command line or from the Registry (or both). Of course the program configuration could be located elsewhere in the Registry. The point is that the above usage would not interfere with the execution of programs or otherwise confuse the operating system into trying to execute a non-existent program.

With JANOS v1.3 and later any console command line may be executed using a `Run` key.

Env/<environment_parameter>

Each program instance has its own environment. This is very similar to a Registry node but unique to each specific process. The program environment contains parameters such as CMDWORKING defining the current working directory as would be changed using the CD command at the command line. It may contain the ERRORLEVEL value as returned by the previous command/program (0 for success).

Parameters defined in this Registry section are considered to be default environment values and will appear initially in every environment. From the command line the environment may be listed and modified using the SET command and applications have programmatic access to the environment.

9.4. Network Configuration

The following keys work in conjunction with `ipconfig`, `hostname` and `jrconfig` to provide for network configuration.

IpConfig/DHCP

JANOS by default will require that IP addressing be defined by the user through the `ipconfig` command. Initially this requires a serial connection to the JNIOR and access to the command prompt. This registry key may be used to enable DHCP allowing the JNIOR to lease an IP address during boot from the local DHCP server. It is preferred that DHCP be enabled directly rather than manually through the Registry by using the command:

```
ipconfig -d
```

which will immediately enable DHCP not requiring reboot. The associated registry key will be updated automatically by the command.

Since JNIOR is a *server* the IP address or some related name is required to access the unit. When DHCP is used and the DHCP server is configured to cache the JNIOR hostname the hostname can conveniently be used to find the JNIOR.

IpConfig/IPAddress

This defines the network IP Address to be used with this JNIOR. The address may be defined through changes to this Registry key, by using the `ipconfig` command, or through DHCP. Changes take effect on reboot. Use `ipconfig` to make immediate changes. This key is not valid and not used when DHCP is 'enabled'.

IpConfig/SubnetMask

This defines the network Subnet Mask to be used with this JNIOR. The mask may be defined through changes to this Registry key or by using the `ipconfig` command. Changes take effect on reboot. Use `ipconfig` to make immediate changes. This key is not valid and not used when DHCP is 'enabled'.

IpConfig/GatewayIP

This defines the network Gateway IP Address. This address is only required if JNIOR is to communicate outside its home network. This would be the case if JNIOR is to synchronize its clock with an external time server (see IpConfig/NTPServer). Changes take effect on reboot. Use `ipconfig` to make immediate changes. This key is not valid and not used when DHCP is 'enabled'.

IpConfig/PrimaryDNS

This defines the Primary DNS IP Address used for name resolution on the network. This would be required if JNIOR is to synchronize its clock with an external time server as DNS is used to resolve "clock.isc.org" into the appropriate IP Address for communication (see IpConfig/NTPServer). Changes take effect on reboot. Use `ipconfig` to make immediate changes. This key is not valid and not used when DHCP is 'enabled'.

IpConfig/SecondaryDNS

This defines the Secondary DNS IP Address used for name resolution on the network should the Primary DNS not be available. Changes take effect on reboot. Use `ipconfig` to make immediate changes. This key is not valid and not used when DHCP is 'enabled'.

IpConfig/HostName

This defines a machine name for the device. By default "jr<$SerialNumber>" will be used. When email is constructed the default "From" address will be HostName@Domain.

IpConfig/Domain

Defines the Domain Name associated with the local network. This may be used to generate an email address such as jnior@integpg.com where the Domain is "integpg.com". Changes take effect on reboot. Use `ipconfig` to make immediate changes.

IpConfig/MailHost

This specifies the address of the SMTP Mail Server that accepts email from this network. This must be specified if JNIOR is going to send email messages. Changes take effect on reboot. Use `ipconfig` to make immediate changes. Any MailHost defined by a DHCP server will override this key.

IpConfig/Username

Specifies the Username required for SMTP Authentication. This may or may not include the domain as this depends on the requirements of the particular server. SMTP Authentication is used ONLY when a MailHost is defined and when both the Username and Password keys are valid.

IpConfig/Password

Specifies the Password required for SMTP Authentication. The password is encrypted in the Registry and is not displayed by the JNIOR except during initial entry using the Registry Editor. The login credentials must be entered using the `ipconfig` command. Each JNIOR has its own unique encryption key and therefore passwords cannot be copied through INI file transfer. SMTP Authentication is used ONLY when a MailHost is defined and when both the Username and Password keys are valid.

IpConfig/EmailAddress

Specifies the email address used for the sender. This email address should be valid and one associated with the email account associated with the login username. This address appears as the sender in most communications. It is also placed in SSL certificates as the JNIOR's owner.

IpConfig/DNSTimeout

This defines the timeout in milliseconds to be used in waiting for a response from the specified DNS servers. The default is 0 and this allows JNIOR to internally specify an optimized timeout.

IpConfig/NTPServer

JNIOR can synchronize with a network time server supporting Network Time Protocol (NTP). To utilize this capability JNIOR must be connected to a network with access to such a server and a Gateway and DNS Server (unless absolute IP address is given) must be defined. The format for the IpConfig/NTPServer key is as follows:

```
IpConfig/NTPServer = ServerAddress [, ServerPort]
```

By default the ServerAddress is either `time.nist.gov` or `ntp.pool.org`. This varies with different versions of JANOS. The standard NTP port number is 123. You may optionally specify a custom port number following the ServerAddress separated by a comma.

Only one time server can be specified. If that server is not available then the synchronization will be bypassed. Note that JNIOR's clock is maintained by a battery during periods without power. Synchronization is not required but useful periodically as the clock will drift in accuracy over long periods. Typical computer hardware clocks (PCs for instance) typically drift by several seconds per day. NTP synchronization is critical in maintaining accurate time.

Time synchronization occurs during power-on boot. Synchronization is attempted every four hours by default to maintain clock alignment. JNIOR may also be commanded to synchronize using the DATE command in the Command Console.

IpConfig/NTPUpdate

JNIOR attempts to synchronize with a network time server every 4 hours (240 minutes) by default. The update period may be adjusted through this Registry key. This defines the period in minutes and can be set for any amount of time 5 minutes or longer. To disable NTP synchronization you can set this key to 0. This configuration setting takes effect on boot. A reboot is required to enable a change.

IpConfig/MTU

The `IpConfig/MTU` Registry key defines the maximum size of any packet transmitted over the Ethernet port. The Maximum Segment Size (MSS) is defined as MTU - 40 (40 bytes less than the MTU setting) and no packet will be transmitted with a payload exceeding this size. The default MTU setting is 1500. Regardless of the MTU setting JNIOR will properly receive packets of any size up to the standard network MTU of 1500.

Connections that utilize PPPoE (modems, VPN) require additional header bytes and therefore cannot carry as large of a payload. In these cases it may be necessary to reduce the maximum transmitted segment size in order to achieve proper operation. For instance the following Registry key may correct PPPoE problems:

```
Registry IpConfig/MTU = 1480
```

Valid MTU settings are 400 to 1500 inclusive. A change in MTU setting applies to all Ethernet connections and takes effect only upon reboot.

IpConfig/SyslogServer

This specifies the address of the Syslog Server that accepts syslog messages. This may be optionally specified to remotely log system status messages. These are typically the information found in the `jniorsys.log` file. Changes take effect immediately.

IpConfig/Keepalive/Time

This is the timeout in seconds before JANOS will probe a connection. By default it is set to 5 minutes (300 seconds). The connection will be probed only after there has been no packet from the peer in the configured period.

IpConfig/Keepalive/Interval

If there is no response from a probe JANOS will retry after the configured interval. By default this is 30 seconds.

IpConfig/Keepalive/Retry

By default JANOS will retry the probe 8 times before closing the connection.

IpConfig/Socket/ConnectTimeout

By default socket connections initiated by an application will time out after 5 seconds generating an IOException. To alter the default setting, define the new timeout in milliseconds using this key. The default setting is 5000.

IpConfig/CaptureBuffer

The JNIOR by default allocates 512KB of memory for network capture. If network traffic needs to be analyzed the NETSTAT –C command is used to generate a PCAPNG capture file which can be downloaded and opened with Wireshark. This means that there is always recent network history available for capture. The default 512KB can represent minutes or even hours of network operation depending on the amount of network use. Only packets involving the JNIOR are captured. This packet buffer can be increased using this Registry key and can be set for any number KB between 512 and 8192 (8MB Maximum). For instance for the maximum capture buffer:

```
reg IpConfig/CaptureBuffer = 8192
```

Note that the capture buffer is volatile and records network activity from the most recent boot. This Registry key setting takes effect only upon reboot.

IpConfig/Promiscuous

By default the network capture collects packets that specifically reference either the JNIOR's MAC address or IP address either as the source or destination. This then excludes general broadcasts and any other unrelated network traffic that the unit may see.

If you need to see all of the network traffic set this Registry key to `true`. This will enable *Promiscuous Mode* and the capture of all network traffic. Note that changes in this setting do not require a reboot and take effect immediately.

Network *switches* and *routers* generally optimize network traffic and present devices with the subset of communications that are specifically addressed. In Promiscuous Mode you will generally receive additional broadcast packets and packets addressed to other possibly non-existing devices which the switch or router has yet to locate and filter.

The *network hub* has been obsoleted by the *network switch* as traffic and bandwidth optimization is a good thing. However the older technology in the hub may be desirable if you need to analyze communications between two other devices. The hub forwards all network traffic to all interconnected devices. The JNIOR in Promiscuous Mode can then capture packet traffic between the other devices. This may be very useful in debugging larger multi-device systems. If you own a hub you should hang onto it as it can be a useful debugging tool when used as a temporary network switch replacement.

IpConfig/CaptureFilter

The network capture buffer can be filtered. This can extend the period over which network traffic can be collected by limiting the content to only those connections or communications of interest. The syntax used to define a capture filter utilizes logical operations such as NOT, AND, OR and XOR. The filter can include references to MAC addresses, IP addresses (IPv4), and TCP/IP or UDP port numbers. Matters of operation precedence can be handled through the use of parenthesis groups (See Network Filtering). By default the network capture is unfiltered.

In a similar fashion packets can be selected from the network capture buffer in creating the PCAPNG file (`temp/network.pcapng`). The filter syntax is the same. You can therefore use the NETSTAT –C command to prototype and test a packet filter before using it to define an incoming capture filter.

Once you have constructed a valid filter you can install it ahead of the capture buffer by setting the `IpConfig/CaptureFilter` Registry key. The filter setting (assuming there are no syntax errors) takes effect immediately and does not require a reboot.

The NETSTAT –F command can also be used to set the incoming filter. This command first verifies the filter syntax and if no errors are found it then sets the Registry key `IpConfig/CaptureFilter`. This is the preferred method in that it includes the syntax check.

An incoming capture filter is non-volatile and will remain in force through a reboot. To remove the filter you must either remove the Registry key or issue the NETSTAT –F command without further arguments.

9.5. Network Filtering

The content of a network capture or the network clients allowed to interact with the JNIOR can be controlled through dynamic filtering. These filters can be quite simple or, if needed, much more sophisticated.

The `IpConfig/CaptureFilter` Registry key may optionally define a filter which is applied to incoming packet data prior to capture. There is limited storage for captured information and by filtering you can extend the capture period and the amount of pertinent information collected.

A filter may also be used in generating the network.pcap capture file from the capture buffer. Here the filter allows you to extract only pertinent information and otherwise keep file sizes at a manageable level.

The `IpConfig/Allow` Registry key may optionally define a *filter* which is applied to incoming connections. In this case the referenced IP addresses refer to the incoming source IP addresses, those of clients. Referenced port numbers refer only to destination ports, those available on the JNIOR.

IP Address Filters

To filter packets referencing a specific IP address you need only include the IP address in the format "nnn.nnn.nnn.nnn" in the filter string. Any packet that references this IP address either as the source or the destination address will be selected for inclusion. All other packets will be excluded unless covered by some other part of the filter.

To exclude packets referencing a certain IP address you can prefix a '!' exclamation point to the address like this "!nnn.nnn.nnn.nnn". All packets that do reference the IP address as either a source or destination address will NOT be selected for inclusion. This can also be written as "NOT nnn.nnn.nnn.nnn".

Note that an IP address is identified by its format, four decimal values between 0 and 255 separated by the '.' period.

The domain syntax allows a range of IP addresses associated with a netmask to be specified. The format is "nnn.nnn.nnn.nnn/mm" where 'mm' specifies the number of high order bits in the netmask. For example, "10.0.0.0/24" specifies any IP address in the domain that contains IP addresses 10.0.0.1 through 10.0.0.255 and uses a netmask of "255.255.255.0". This would be useful in selecting only local traffic for instance.

MAC Address Filters

Although less often required you can filter on a specific MAC address. The MAC address is included in the filter string in the format "hh:hh:hh:hh:hh:hh". The format is six hexadecimal values (0-9a-f or 0-9A-F) separated by the ':' colon. For instance most INTEG Series 4 JNIORs have MAC address formatted as "9C:8D:1A:hh:hh:hh" where the lower three bytes are assigned in some sequence.

As with IP addressing, packets with MAC addresses may be excluded by writing the filter as "!hh:hh:hh:hh:hh:hh" or "NOT hh:hh:hh:hh:hh:hh". Again a MAC address is identified by its format.

TCP/UDP Port Filters

A port is specified in the filter string as a decimal value between 1 and 65535 inclusive. No punctuation is used. The capture filter does not distinguish between a TCP or UDP port number. A port may be excluded using the negation "!nnn" or "NOT nnn".

There are standard ports assigned for various functions. The capture filter knows some of them by name. Some may be reconfigured through the Registry. As a convenience the port may be specified using its protocol name. The capture will be filtered on the port as configured at the time the filter is compiled (at boot or upon NETSTAT command). JANOS recognizes these port names where the default values are shown in parentheses: SMTP (25), NTP (123), JNIOR (9200), FTP (21), HTTP (80), HTTPS (443), TELNET (23), and BEACON (4444). These ports may be excluded using the same negation syntax as previously shown.

Boolean Constants

The capture filter will also recognize the terms TRUE and FALSE. True indicates that the packet is to be included and False otherwise.

Logical Operations

To filter on a single IP address, MAC address or port (or to exclude a single item) the filter need only specify the address or port in the associated format. The following would select the communications involved in an email transfer. If this is used as an incoming filter, only email transactions would be captured. If this is used with NETSTAT –C in generating the PCAPNG file, the file would only include email communications.

```
NETSTAT –C SMTP
netstat –c 25
```

Note that filters (and also commands) are not case-sensitive. Either form above will create a PCAPNG file with just email communications. This assumes that you have not reconfigured the SMTP port. If you have set `Email/Port` to another port (587 for instance) then the first line will extract your email communications and the second will not. Although the second filter might show an application trying to use the incorrect port.

Filters often need to be slightly more complex in order to include the collection of communications needed. The syntax allows you to specify any number of addresses or ports in any combination using AND, OR and XOR logic. As an alternative you may use the notation '`&&`" and '`||`' for AND or OR respectively.

As an example perhaps you want to filter only email communications with the SERVER.

```
netstat –c 10.0.0.4 && smtp
```

If you want to also include BEACON communications you might write the filter as:

```
netstat –c 10.0.0.4 AND smtp OR beacon
```

But here you might question the order of precedence of the logical operations. The capture filters do not support an order of precedence but perform the operations from left to right. So this would be calculated as follows:

```
netstat –c (10.0.0.4 && SMTP) || BEACON
```

And this would have done what we had said. If there is some question you can use the parentheses in the filter as shown. The following will create the same subset of packets but would not if we were to exclude the parentheses:

```
netstat –c BEACON || (10.0.0.4 && SMTP)
```

A parentheses grouping can be negated as you would expect. The following will create a capture of all activity EXCEPT email communications with the SERVER.

```
netstat –c !(10.0.0.4 && smtp)
```

Finally if we had wanted to mask these email communications from the overall capture buffer we can install this filter using the command:

```
netstat –f !(10.0.0.4 && smtp)
```

This would result in the following Registry setting and would filter out matching communications until such time as the filter is removed.

```
IpConfig/CaptureFilter = !(10.0.0.4 && smtp)
```

9.6. Security

JANOS provides for secure communications over public networks using Transport Layer Security (TLS). This is an implementation of secure communications that replaces previous Secure Sockets Layer (SSL) approaches which have been shown to have vulnerabilities. The overall approach however is still often referred to as "SSL Security".

Various aspects of the secure communications in JANOS may be configured through the Registry. All of these configuration settings are maintained in the 'SSL' branch of the registry.

IpConfig/Allow

The `IpConfig/Allow` Registry key when present defines filtering to be applied to incoming connection requests. This uses the network capture filter syntax (See Network Filtering). This not only provides for the ability to specify IP addresses that are allowed to connect to the JNIOR but gives you the flexibility to block IP addresses. This includes domain ranges (subject to netmask) and destination ports. This filter can be used to not only control *who* can access the unit, it can also be used to define *what* they can access.

SSL/Enabled

Enables or disables the ability to make TLS secured connections. By default this key is TRUE. When set to FALSE this disables the Secure Web Server on Port 443 (HTTPS); Removes the ability to upgrade a JNIOR Protocol, FTP or Telnet connection to the secured state (disables STARTTLS); And, disables the routine Security Update procedure which otherwise is run to update keys.

The 'certmgr' command line function remains fully functional. Security keys and certificates may still be managed while the ability to make secure connections is disabled.

This setting takes effect upon reboot.

SSL/Required

By default this is set to FALSE. When TRUE this forces the use of SSL secured connections. No web services are provided on Port 80 (HTTP). All FTP sessions must be secured through the STARTTLS mechanism. The JNIOR Protocol and Telnet (command line) connections will close should data be received before the connections are secured.

This setting takes effect upon reboot. It is ignored if SSL/Enabled is set to FALSE.

9.6.1. Basic Authentication and Passwords

Access to the JNIOR is password controlled. All protocols provide for a means of login which requires the entry of a username and password. If those connections are not secure (such as standard browser access using HTTP as opposed to HTTPS) then both the username and password are transferred in clear text. These are easily compromised by the simplest of techniques.

NOTE
To insure minimal security, you MUST be certain that ALL protocols are set to require password authentication. Otherwise, even when SSL secure connections are made anyone will be able to alter and/or control your JNIOR.

Not all protocols that are typically used in the industry provide for a standard means of password authentication. MODBUS is an example of this. The JNIOR does extend these protocols providing such a means but this must be specifically enabled through this Registry and may require changes to the connecting client.

Web pages may be secured but may not be by default. These pages must be made private by removing the User read/write/execute privileges on either the path or the individual page files. The following command can, for example, be issued at the JANOS command line to secure the entire JNIOR web site.

```
chmod -rwx flash/www
```

Once these permissions have been changed the browser will request a username and password which will be used for all access to the site while the browser is open. Again, an HTTPS secure connection is required to protect these user credentials which would otherwise cross the network in essentially a plain text format.

9.6.2. Default Guest and Administrator Credentials

Even with care to use both secure connections and password authentication the JNIOR may be easily compromised if the default user accounts are not removed or given unique strong passwords. Surprisingly a large percentage of JNIORs are left with the default user accounts. We have seen some units where care had been taken to change the password on commonly used ‘jnior’ administrator account but failed to change or remove the secondary ‘admin’ administrator default account.

NOTE
To insure security, you MUST be remove any unused user accounts and CHANGE the passwords from their defaults on remaining accounts.

The JNIOR may be supplied with default administrator accounts ‘jnior’ and ‘admin’, a default user account ‘user’ and a default guest account ‘guest’. The default passwords are simply the usernames. JANOS command line functions provide for user management.

Users/IgnoreDefault

The JNIOR comes with two default Administrator accounts enabled. These are the ‘jnior’ and ‘admin’ accounts whose passwords are ‘jnior’ and ‘admin’ respectively. This can represent a significant security risk if either account is left active with the default password. Users often alter the ‘jnior’ account password but neglect to adjust the ‘admin’ account or vice versa. Periodically JANOS will post a warning to the jniorsys.log file if either default account is determined to still be using the default password.

It is highly recommended that the PASSWD command be used to set unique passwords for these two accounts. You may alternatively use the USERMOD command to disable an unused account with the +D option.

If you are comfortable with the risk and would like to continue to use the default accounts and passwords, you can eliminate the warning by setting the `Users/IgnoreDefault` Registry Key to TRUE.

Note that if you do forget your administrator password(s), the SAFE_MODE access procedure may be used to regain control of your JNIOR. You can then assign a new password.

9.6.3. Public/Private Key Pair

Secure communications require RSA keys. 1024-bit or 2048-bit key lengths are typically used today. Longer keys are usually required to protect highly sensitive information and to increase protection as the computer capacity to break (determine the private key associated with a published public key) increases. The JNIOR control is not intended for use in extremely secure environments and its processing capabilities limit it to a maximum 1024-bit key pair.

As shipped the JNIOR is factory configured with a standard 512-bit key. Upon initial power up the 'Security Update' process will run and a unique 1024-bit key will be generated. This may take an hour or so to complete during which time the JNIOR is fully functional and the default key can be used. This can also be interrupted and restarted as you need. The background key update can be ignored. JANOS may update keys on a periodic basis (say monthly) using the Security Update process.

The 'certmgr' command line function may be used to install and externally generated RSA key pair. This is limited to a 1024-bit key length. The Security Update process cannot overwrite an externally loaded key pair.

9.6.4. SSL Certificates

A TLS secured communications channel requires both a RSA key pair and a SSL Certificate. The 'certmgr' command line function may be used to install an externally generated and signed SSL Certificate that must be associated with the separately installed RSA key pair. Typically the internally generated key pair is sufficient.

In the absence of a loaded SSL Certificate, JANOS will generate a self-signed certificate using the current RSA key pair. Registry keys are provided which allow you to customize the information provided in this certificate. Since self-signed certificates are not generally recognized as "trusted" by browsers, users will be confronted by a standard warning. The information in the certificate may be configured so your users may recognize the device and decide on their own to accept the connection. The default values provide for a fully functional connection.

SSL/Cert/C

This text string defines the Country. By default this is "US" as this is where INTEG is located.

SSL/Cert/ST

This text string defines the State. Generally this is not an abbreviation. By default this is "Pennsylvania" as this is where INTEG is located.

SSL/Cert/L

This text string defines the Locality, City or Town. By default this is "Gibsonia" as this is where INTEG is located.

SSL/Cert/O

This text string defines the Organization. By default this is "INTEG Process Group Inc".

SSL/Cert/OU

This text string defines the Organizational unit, Division, Department or other. By default this is “JNIOR I/O Resource”. Here we take the opportunity to identify the device.

SSL/Cert/CN

This text string defines the Common Name or FQDN. For the proper operation of the web site this should reflect the URL used to reach the JNIOR. For example if one uses http://10.0.0.60 to open the JNIOR at IP address 10.0.0.60 then the CN should be “10.0.0.60”. JANOS by default will set this according to the IP configuration of each unit. Consequently the certificate will be regenerated should the unit’s IP address change.

SSL/Cert/SAN

[Requires v1.6.1 JANOS or later.] Certificates are expected to be created for specific domains and should match the URL used to access the unit. The Common Name or FQDN is by default defined to be the IP address of the JNIOR. If you also want to access the unit using a domain name you can add additional DNS names using this Registry key. One or more names may be added using comma (,) separated list. These will appear in the *Subject Alternative Name* extension of the unit’s v3 SSL Certificate. Note that you will need to regenerate your certificate if you make changes to these `SSL/Cert` keys. Use the CERTMGR –C command in the JANOS Console.

SSL/Cert/E

This text string defines the contact email address. By default this will use the email address defined by `/IpConfig/EmailAddress` and in the absence of that an email address back at INTEG.

SSL/Cert/Days

This integer defines the length in days of the period during which the certificate is considered valid. This starts on the date when the certificate is generated or regenerated. By default this is 730 days (2 years).

SSL/Cert/SHA1

The SHA1 cryptographic hash function is no longer considered to be secure. It remains secure for most of the world but those with sufficient resources are assumed now to be capable of breaking it. The JNIOR now uses the SHA256 algorithm (SHA2) as do most of the systems operating around the world. You can disable use of SHA2 if you need to communicate securely with legacy systems or systems that have not yet been upgraded. This is achieved by setting SSL/Cert/SHA1 to true.

As with most of the settings in this category, changes take effect when the certificate is regenerated.

9.7. Event Management

There are various events which occur during the operation of JNIOR. Some are normal occurrences and others not so normal. JANOS can be configured to perform certain actions when these events occur. The following Registry keys define how JANOS is to respond to such events.

Events/Services

JNIOR monitors events and responds to certain situations depending on the configuration established by the Registry. JNIOR also can generate an audit trail of events and otherwise routinely log changes in data. By default these services are “enabled”. This Registry key can be used to completely disable all such services. A setting of “disabled” will stop processing without affecting the configuration. This key takes effect immediately and reboot is not required.

Events/OnBoot

This key can be used to globally enable or disable the activities related to Power On Reset (Boot) defined below. If set to “disabled” this key will override and disable each of the `OnBoot` actions. If set to “enabled” this will enable all of the `OnBoot` activities not specifically set within the section.

Events/OnBoot/Email

When “enabled” this instructs JNIOR to send an Email on Boot. The format is as follows.

```
reg Events/OnBoot/Email = enabled
```

For example, the following will send an (admittedly basic) email to an INTEG employee when JNIOR boots.

```
reg Events/OnBoot/Email = enabled
reg Email/OnBoot/ToAddress = jdoe@integpg.com
```

See Email Specification below for more information.

Events/OnBoot/EmailBlock

This key specifies an optional named `Email` section that defines the content and addressing of the particular message to be used. The format is as follows.

```
reg Events/OnBoot/EmailBlock = "EmailBlock"
```

If this key is undefined the email is sent using the "default" Email message definition. A custom or "named" Email message might optionally be referenced by defining it using the named block. For example:

```
reg Events/OnBoot/Email = enabled
reg Events/OnBoot/EmailBlock = AdminNotify
reg Email/AdminNotify/ToAddress = jdoe@integpg.com
reg Email/AdminNotify/Attachment = jniorboot.log
```

The key in this case defines the subsection/node of the `Email` section or that part following "`Email/`" and this can be any descriptive name that may be helpful. It must match that used in the EmailBlock specification or the email will not be generated. See Email Specification below for more information.

Events/OnAlarm

This key may be optionally defined and set to "disabled" to override and disable the individual Digital Input state alarm services defined by the individual `IO/inputs/din#/Alarm/` Registry sections. If any state alarm services are desired then this key may be left undefined as it is "enabled" by default.

Events/OnAlarm1

This key may be optionally defined and set to "disabled" to override and disable the individual Digital Input Counter alarm Type 1 services defined by the individual `IO/inputs/din#/Alarm1/` Registry sections. If any counter alarm Type 1 services are desired then this key may be left undefined as it is "enabled" by default. Counter Type 1 and Type 2 alarm services differ only in the set points used.

Events/OnAlarm2

This key may be optionally defined and set to "disabled" to override and disable the individual Digital Input Counter alarm Type 2 services defined by the individual `IO/inputs/din#/Alarm2/` Registry sections. If any counter alarm Type 2 services are desired then this key may be left undefined as it is "enabled" by default. Counter Type 1 and Type 2 alarm services differ only in the set points used.

Events/OnAlarm/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
Events/OnAlarm/Email = enabled [, EmailBlock]
```

This key may be optionally set to "disabled" to override and disable the transmission of Email Notifications in response to individual Digital Input state Alarms as defined by the individual `IO/inputs/din#/Alarm/Email` Registry keys. If state alarms Email Notifications are desired then this key may be left undefined as it is "enabled" by default. Counter Type 1 and Type 2 alarm services differ only in the set points used.

An optional EmailBlock may be defined specifying the alternate default Email format to be used for Digital Input state alarms. If the individual `IO/inputs/din#/Alarm/Email` keys omit an Email Block specification

the one defined here would be used. If no block is given in either location then the default Email format defined in the `Email/` section would apply. See the Email Specification below.

Events/OnAlarm1/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
Events/OnAlarm1/Email = enabled [, EmailBlock]
```

This key may be optionally set to "disabled" to override and disable the transmission of Email Notifications in response to individual Digital Input counter Type 1 Alarms as defined by the individual `IO/inputs/din#/Alarm1/Email` Registry keys. If counter Type 1 alarms Email Notifications are desired then this key may be left undefined as it is "enabled" by default. Counter Type 1 and Type 2 alarm services differ only in the set points used.

An optional EmailBlock may be defined specifying the alternate default Email format to be used for Digital Input counter Type 1 alarms. If the individual `IO/inputs/din#/Alarm1/Email` keys omit an Email Block specification the one defined here would be used. If no block is given in either location then the default Email format defined in the `Email/` section would apply. See the Email Specification below.

Events/OnAlarm2/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
Events/OnAlarm2/Email = enabled [, EmailBlock]
```

This key may be optionally set to "disabled" to override and disable the transmission of Email Notifications in response to individual Digital Input counter Type 2 Alarms as defined by the individual `IO/inputs/din#/Alarm2/Email` Registry keys. If counter Type 2 alarms Email Notifications are desired then this key may be left undefined as it is "enabled" by default. Counter Type 1 and Type 2 alarm services differ only in the set points used.

An optional EmailBlock may be defined specifying the alternate default Email format to be used for Digital Input counter Type 2 alarms. If the individual `IO/inputs/din#/Alarm1/Email` keys omit an Email Block specification the one defined here would be used. If no block is given in either location then the default Email format defined in the `Email/` section would apply. See the Email Specification below.

Events/OnUsage

This key may be optionally defined and set to "disabled" to override and disable the individual Usage Alarm services defined by the individual `IO` Registry sections. If any usage alarm services are desired then this key may be left undefined as it is "enabled" by default.

Events/OnUsage/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
Events/OnUsage/Email = enabled [, EmailBlock]
```

This key may be optionally set to "disabled" to override and disable the transmission of Email Notifications in response to individual Usage Alarms as defined by the individual `IO` Registry keys. If Usage Alarm Email Notifications are desired then this key may be left undefined as it is "enabled" by default.

An optional EmailBlock may be defined specifying the alternate default Email format to be used for Usage Alarms. If the individual `IO` keys omit an Email Block specification the one defined here would be used. If no block is given in either location then the default Email format defined in the `Email/` section would apply. See the Email Specification below.

Events/OnConfig

Certain services may be performed when JNIOR updates the jnior.ini file in response to changes in the Registry. Set this key to “disabled” to override and disable all services related to a detected configuration change. By default this key is “enabled”.

Events/OnConfig/Email

When configuration changes are made and settings altered in the Registry the jnior.ini file will be updated to reflect those changes. JNIOR may be configured to send a Configuration Change Notification when this occurs. This key contains two optional entries. The format is as follows.

```
reg Events/OnConfig/Email = enabled
reg Events/OnConfig/Email = “enabled, EmailBlock”
```

If a Configuration Change Notification email is desired this key must be set to “enabled” and the `Events/OnConfig` key must not be set to “disabled”. Optionally an EmailBlock can be specified to select the email format to use (*see Email Specification*).

9.8. Configuring Email Notifications

JANOS can send email messages in response to certain events. Any number of unique Email messages can be defined for use as the situation requires. A generic (not situation specific) Email is defined by the following keys. Unique Email construct can be defined and assigned to unique Registry sections. These may be separately referenced and used as the situation may require. To create a situation specific E-mail message replace <block> with a unique message identifier in those keys where it appears.

Email/<block>/ToAddress

This defines one or more destination email addresses of the form user@domain.com. Multiple addresses are separated by commas. For example:

```
Email/AdminNotify/ToAddress = jdoe@integpg.com, admin@integpg.com
```

Note that Registry key values are limited to 255 characters in length and this would place a limit on the number of email address that may be defined. It is the user’s responsibility to ensure that these addresses are valid and are updated as needed.

Email/<block>/CcAddress

This defines one or more destination email addresses of the form user@domain.com. These addresses will also receive the defined message but as a CC: recipient.

Email/<block>/BccAddress

This defines one or more destination email addresses of the form user@domain.com. These addresses will also receive the defined message but as a BCC: recipient.

Email/<block>/Subject

This defines the Subject line to be used with the message. JNIOR requires that a Subject be defined for all messages although this is not strictly a requirement for email itself. If the Subject key is not given, JNIOR will

utilize a default Subject as appropriate to the purpose of the email.

Email/<block>/Message

This defines text Message content to be sent in the email. JNIOR does not require that this message content be supplied. This may be used in conjunction with a Message File and the text defined here will appear as a prefix to the content of the file. This is not as applicable in combination with HTML formatted file content as in that case this text would have to comply with the HTML structure as well.

Email/<block>/MessageFile

This defines the file containing the textual Message content to be included in the email. If separate Message text is supplied the content of this file will be appended to that text in the message.

Email/<block>/HTMLMessageFile

This defines the file containing the Message content which is assumed to be defined using valid HTML coding. If the file exists this supersedes the MessageFile specification. If separate message text is supplied the HTML content will be appended to that text and therefore the separate text must comply with the HTML structure.

Email/<block>/Attachments

This lists one or more attachments (file names) to be sent with the email message. Each file specification is to be separated by a ';' semicolon.

Email/FromAddress
Email/ToAddress
Email/CcAddress
Email/BccAddress
Email/Subject
Email/Message
Email/MessageFile
Email/HTMLMessageFile
Email/Attachments

A Default Email Format may be specified by omitting the <block> section. The above keys define the format for an email that would be used when a uniquely named Email Block is not specified.

Email/Signature

By default the Series 4 JNIOR includes a signature line in emails indicating the model and serial number of the sending unit. The version of JANOS is also included. You may provide a custom signature line overriding the default using this registry entry.

Email/RetryCount

There may be difficulties in delivering an e-mail to the destination server. By default after an attempt has failed JNIOR will schedule the delivery of the message and will do so up to 12 times. Failures on an initial attempt are typical these days as servers are implementing greylisting techniques to reduce the amount of unsolicited spam e-mail. In general the Internet is a lossy network and retries are not unusual. Set RetryCount to 0 or 1 to disable retries.

Email/RetryDelay

After a failed e-mail delivery attempt JNIOR will reschedule another delivery at a later time. By default this is configured for 10 minutes. The RetryDelay key defines this time in minutes. E-mail servers implementing greylisting my routinely reject initial e-mail deliveries for typically 30 minutes. These techniques are designed to cause spammers some difficulties and help to cut down the amount of unsolicited e-mail. The JNIOR should be set to attempt repeated deliveries for at least a couple of hours to increase chances of success.

Email/Port

The default Simple Mail Transfer Protocol SMTP port is 25. This key may be used to specify an alternate port as may be required by your MailHost. Note that the MailHost and any associated SMTP Authentication settings (Username and Password) are set by keys in the IpConfig section.

9.9. Web (HTTP) Server

The following configuration settings control the WWW interface to JNIOR.

WebServer/Server

This entry can be used to disable the HTTP Server. This may be desirable if communications with JNIOR will be through some other means and connections to the JNIOR HTTP Port are to be ignored. The default setting is "enabled". Changes take effect on reboot.

WebServer/Port

This specifies the TCP/IP port to use for HTTP services. The default is standard port 80.

WebServer/SSLPort

This specifies the TCP/IP port to use for HTTPS secure connections using TLS/SSL. The default is standard port 443.

WebServer/Root

This specifies the folder within the JNIOR file system that represents the root of the website. This is the folder that would contain the unit's home page and related pages are located in this folder or in subfolders. The default is "/flash/www/". This folder must be specified from the root of the file system (starting with a '/'), The trailing '/' is optional.

WebServer/Index

This specifies the name of the page to be served as the unit's home page. This is the HTML document that would appear if only the JNIOR's IP address were referenced from the browser for instance. The defaults are "index.php" followed by "index.html". Names may be separated by a semicolon ';'.

WebServer/Path

This may be used to specify alternate search paths for web content. The JNIOR first searches the Root of the WebServer for the requested content. If the page is not located then each path defined in this key is searched in sequence. Paths must be specified from the root of the file system (starting with a '/') and the trailing '/' is optional. Multiple paths are separated by a semicolon ';'.

WebServer/Login

By default the Web Server requires a successful login. It is highly recommended. If the JNIOR is connected to a private secure network this login requirement can be removed by setting this Registry key to 'disabled' and defining a user account for anonymous login through the `WebServer/Anonymous` key. These changes take effect immediately. You will not be logged out of your current session(s). Note that login may still be required if folder or file permissions are restricted (See CHMOD console command). By default initially all users have access to all folders and files.

WebServer/Anonymous

If the JNIOR is running on a private secure network a login may not be a necessity. The login requirement is removed using the `WebServer/Login` key. When the login is bypassed a user account must be defined for the JNIOR to use. The `WebServer/Anonymous` key must be set to a valid active user account with the entry of either a UserID or Username. With the default set of user accounts if you want these anonymous sessions to have Administrator privileges then you would set this key to 'jnior' as an example. Note this if the anonymous account is invalid or disabled the JNIOR will continue to request login credentials.

9.10. Websocket Interface

The WebServer provides the ability to upgrade any connection to support the Websockets Protocol. JANOS supplies a built-in Websocket interface that uses JSON creating a message and response structure. This can replace all of the functionality of the JNIOR Protocol and provide much more. In addition, application programs can be created as custom Websocket servers.

Websocket/Login

The Websocket interface fully supports administrative management and data monitoring functions. It requires a successful login. When this service is accessed through a local website served by the JNIOR's WebServer the login credentials used to access the web pages are applied automatically to the Websocket interface. This Registry key then should only need to be altered if you need access from an external website and cannot accommodate the login. This is not recommended. If necessary, set this key to 'disabled' and also define a user account using the `Websocket/Anonymous` key to remove the login requirement.

Websocket/Anonymous

Defines the user ID (integer) applied to anonymous logins. When a Websocket connection requires a login (default) the login must reference a defined username using the correct password for that account. In order to accommodate a scheme whereby data monitoring would not require login but control or configuration would, JANOS allows for *anonymous* login. When the Websocket/Anonymous key is defined (exists) and the `Websocket/Login` key is set to 'disabled' anonymous login is allowed. The key must contain a valid User ID for a user account with the permissions appropriate for anonymous use. To prevent anonymous login this key should be removed from the Registry.

Websocket/Files

The built-in Websocket interface supports file management. Files may listed, read, written, renamed and deleted. Similarly folders can be created, renamed and removed. For additional security this feature can be disabled by setting this key to `disabled`. This removed the file management function from the interface (after reboot) . The key also signals the DCP to remove the File Folders tab.

Websocket/Console

Each connection to the built-in Websocket interface can support a single command line (console) session. For additional security this feature can be disabled by setting this key to `disabled`. This removes the console functionality for the Websocket interface (after reboot). The key also signals the DCP to remove the Console tab.

Websocket/Registry

The built-in Websocket interface supports functions for Registry key management. This is a basic requirement for JNIOR configuration. This cannot be disabled. This key, however when set to `disabled` signals the DCP to remove the Registry Editor tab.

9.11. Jnior Protocol Server

The following are configuration settings for the JNIOR protocol server.

JniorServer/Server

This entry can be used to disable the JNIOR Server. This may be desirable if communications with JNIOR will be through some other means and connections to the JNIOR I/O Port are to be ignored. The default setting is "enabled". Changes take effect on reboot.

JniorServer/Port

This defines the IP port on which JNIOR will listen for connections. The default port is 9200. Changes take effect on reboot.

JniorServer/Login

By default the JNIOR protocol requires a successful login. This is achieved through a function as part of the protocol. It is highly recommended. If the JNIOR is connected to a private secure network this login requirement can be removed. Set this Registry key to 'disabled'. The change takes effect immediately. Note that this requires that `JniorServer/Anonymous` be set.

JniorServer/Anonymous

Defines the user ID (integer) applied to anonymous logins. When the JNIOR protocol requires a login (default) the login must reference a defined username using the correct password for that account. In order to accommodate a scheme whereby data monitoring would not require login but control or configuration would, the JNIOR OS allows for *anonymous* login. When the JniorServer/Anonymous key is defined (exists) anonymous login is allowed. The key must contain a valid User ID for a user account with the permissions appropriate for anonymous use. To prevent anonymous login this key should be removed from the Registry.

For a Series 4 JNIOR the User ID for any user account can be found using the `users` command at the command line prompt.

An anonymous login is one where both the username and password are omitted (null) in the associated login request.

JniorServer/RemoteIP

The JNIOR protocol server can be configured to make a connection to one remote host. This is configured through the `JniorServer/RemoteIP` and `JniorServer/RemotePort` registry keys. The `JniorServer/RemoteIP` key defines the IP Address of the Remote Host that the JNIOR will connect to.

JniorServer/RemotePort

The JNIOR protocol server can be configured to make a connection to one remote host. This is configured through the `JniorServer/RemoteIP` and `JniorServer/RemotePort` registry keys. The `JniorServer/RemotePort` key defines the Port of the Remote Host that the JNIOR will connect to.

9.12. FTP Server

JANOS supports a fully functional FTP server. This is typically used to transfer files to and from the JNIOR for configuration.

FTP/Server

This entry can be used to disable the FTP Server. The default setting is “enabled”. Changes take effect on reboot.

FTP/Port

This defines the IP port on which JNIOR will listen for FTP command connections. The default port is 21.

FTP/Anonymous

Normally the FTP server requires a valid user login. If security is not of concern the server may allow anonymous logins. To enable anonymous login set this registry key to true.

FTP/UnixStyle

The specification for the File Transfer Protocol does not specify the format for directory listings. Originally the detail was only for display and could be in the system’s native format. There are two generally used layouts. Systems based on the Windows™ operating system provide an MS-DOS™ style listing while most others provide a Unix format. JANOS provides the MS-DOS style by default.

FTP clients typically now need to interpret the listing for graphical display and tracking of directory/folder content. Most client programs will detect the formatting and process the content as needed. Other clients might expect one style or the other.

If an FTP client has difficulty retrieving the directory listing from the JANOS FTP Server you may set this Registry Key to TRUE. The FTP Server will then supply the Unix formatted directory listing when requested.

9.13. Telnet Server

JANOS provides a Telnet server providing access to the Command Console. This is typically used for configuration and diagnostics.

Telnet/Server

This entry can be used to disable the Telnet Server. The default setting is “enabled”. Changes take effect on reboot. The Command Console is also available through Web Server websockets using the configuration pages.

Telnet/Port

This defines the Telnet port . The default port is 23.

9.14. BEACON Service Protocol

The BEACON protocol is used by the Support Tool to identify JNIORs on the network, configure initial IP addressing, and provide some management functions. This protocol was developed by INTEG for this purpose. The BEACON protocol has been documented separately. Changes to settings in this section require a reboot in order to take effect.

Beacon/Enabled

For added security the ability of this protocol to respond to remote commands may be disabled by setting this Registry key to `false`. It is enabled by default.

Beacon/Announce

The BEACON protocol by default announces the presence of the JNIOR on the network using a broadcast message. The Support Tool uses this to list local JNIORs on the Beacon tab. This announcement is not disabled with the setting of the `Beacon/Enabled` key. This feature may be disabled by setting this key to `false`. Setting both keys to `false` insures that all BEACON operation is disabled.

Beacon/AutoAnnounce

The BEACON protocol announces the JNIOR on boot by default and whenever key configuration settings change. There may be extended periods without an announcement. This protocol may be used to monitor the health of the JNIOR. A periodic announcement every 30 seconds can be enabled by setting this key to `true`. By default it is not enabled.

9.15. I/O Logging

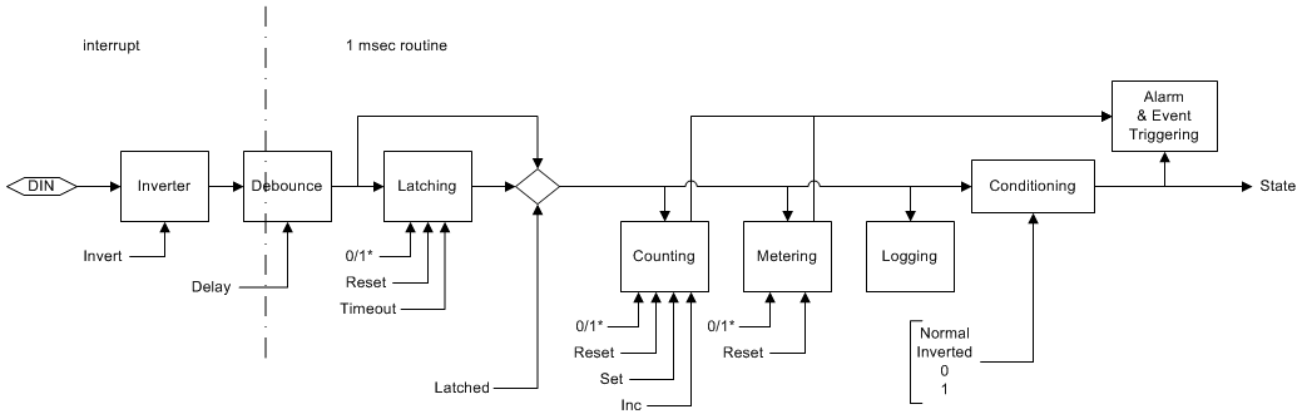
The following keys apply specifically to the JNIOR Digital Inputs and Relay Outputs in general.

IO/Log

The JNIOR can provide a record of changes to its inputs and outputs. When changes are detected an entry is made in the `/jniorio.log` file. By default this logging feature is disabled. To enable logging set this key to "enabled". Logging can be enabled or disabled on an input or output group basis or for each individual input or output. This is controlled by separate key entries in the associated sections as described below.

9.16. Configuring Digital Inputs

Each digital input may be configured in a number of ways to achieve the desired function. The following diagram shows how inputs are handled by JNIOR.



The following keys are associated with the Digital Inputs. Note: In each of the following Keys replace the '#' with the appropriate channel number 1-4 (Model 412), 1-8 (Model 410) or 1-12 (Model 414).

IO/Inputs/Log

If the IO/Log key is set to “enabled” a record of input changes is generated. This key can be optionally used to specifically disable logging in general for the Digital Inputs as a group. By default it is enabled.

IO/Inputs/din#/Desc

This defines the textual description for the associated Digital Input (String). The defaults should be “Digital In #” where ‘#’ is replaced by ‘1’ through ‘12’ as appropriate.

IO/Inputs/din#/OnDesc

This defines the textual description used when the associated input is in the “On” state. “On” should be the default.

IO/Inputs/din#/OffDesc

This defines the textual description used when the associated input is in the “Off” state. “Off” should be the default.

IO/Inputs/din#/Inversion

Set to “enabled” to invert the sense of the Digital Input. When enabled the input will be considered “On” when no voltage is applied to the external circuit (LED off). The input will be considered “Off” when voltage is applied (LED on). The default is “disabled”. This inversion is applied immediately to the input and affects all other functions (Alarming, Counting, etc.).

IO/Inputs/din#/Conditioning

Sets the final input signal conditioning. By default the input state is as reported by the latching or debounce stages. You may configure inversion here or force the input to be read as 0 (OFF) or 1 (ON). Note that counting and usage metering remain operational even when inputs are forced to a fixed state. The following settings are valid:

- 0 – Normal (default)
- 1 – Inverted
- 2 – Forced to 0 (OFF) state
- 3 – Forced to 1 (ON) state

IO/Inputs/din#/Debounce

Relays and switches have mechanical contacts which physically make or break a circuit. Rarely will the contacts come together solidly or separate decisively without bouncing (briefly making and breaking the circuit). This can raise havoc with digital latching and counting circuits that might be monitoring through the relay/switch contact. It can result in latching at the wrong time (when the relay opens for instance) or in extra counts. Both are undesirable.

By default the JNIOR digital inputs are *debounced*. The default debouncing delay is 200 milliseconds. An input must remain quiet (not change) for 200ms before any transition on that input will be processed (latched, counted or logged). This is sufficient to eliminate almost all of the issues arising from contact bounce.

This Registry Key optionally allows for the adjustment of the debounce filter. A setting of 0 (zero milliseconds) disables the software debounce. In this case the JNIOR is capable of counting transitions occurring at rates up to 2,000/sec. If the Registry Key is absent the JNIOR will use whatever setting had been previously given. It does

not fall back to factory default if you remove the key. The filter setting is non-volatile. Valid settings are 0 thru 255 (milliseconds).

IO/Inputs/din#/Latching

Set to “enabled” when the associated input is to be latched. When enabled, the input will be considered to remain in the state defined by the LatchState setting after the voltage applied to the external input is changed. The input is considered to be latched. If the LatchTime is set to 0 seconds the User must manually reset the input. This can be done through the Monitoring website or other external application. The default is “disabled”. This does not affect Counting.

IO/Inputs/din#/LatchTime

This defines the time in seconds (0.1 equals 100 milliseconds) that an input remains latched before being automatically reset. A value of 0.0 will require the User to separately reset the latched input. The default is 0.0 seconds.

IO/Inputs/din#/LatchState

When Latching is enabled this specifies whether the input is latched in the “On” state (1) or in the “Off”(0) state. The key is set to ‘1’ or ‘0’ respectively.

IO/Inputs/din#/Log

If the IO/Log key is set to “enabled” and IO/Inputs/Log key has not been set to “disabled” a record of input condition changes will be written to the /jniorio.log file. This key can be optionally used to disable logging on an input by input basis. If an input is going to be rapidly changing the time spent in the logging process can degrade system performance. In such circumstances it is recommended that logging be disabled for the input.

IO/Inputs/din#/$HourMeter

This dynamically reports the total number of hours that the individual digital input has physically been in the state specified by UsageState. This value is non-volatile maintaining content through power loss and until it is specifically reset. It is reported here in hours to the one-hundredth. The Hour Meter is accurate to the millisecond and this high resolution value may be read through the JNIOR Protocol or through the JRMON command.

IO/Inputs/din#/Alarming

Set to “enabled” when the associated input is to generate a state alarm.

IO/Inputs/din#/Alarm/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
IO/Inputs/din#/Alarm/Email = enabled [, EmailBlock]
```

This key may be set to “enabled” to enable the transmission of Email Notifications in response to a Digital Input state change Alarm. An optional EmailBlock may be defined specifying the Email format to be used.

IO/Inputs/din#/Alarm/Inversion

Set to “enabled” when the associated input is to alarm upon entering the “Off” state. The default is “disabled” and normally when alarming is enabled the alarm is generated when the input turns “On”.

IO/Inputs/din#/Alarm/OnAlarm

When set to “enabled” JNIOR will provide services related to the occurrence of Input Alarms generated by a state change on this Digital Input. Counter Alarms are handled separately. If the key is set to “disabled” no Digital Input state alarm services will be provide. By default services are “enabled”.

IO/Inputs/din#/Alarm/HoldOff

This defines the amount of time in milliseconds that the Digital Input state alarm must remain clear before any subsequent state alarm on this input will be acted upon. The default is 300000 or 5 minutes. When an alarm occurs the services associated with that event are performed. The alarm must reset and remain so for this amount of time before those actions would be performed again.

IO/Inputs/din#/CountState

This specifies whether an input transition from “Off” to “On” is counted (1) or if the transition from “On” to “Off” is counted. The key is set to ‘1’ or ‘0’ respectively.

IO/Inputs/din#/Count/Units

This defines the “units” text to be displayed with the associated input counter. The default should be “counts”.

IO/Inputs/din#/Count/Multiplier

When the Count Multiplier is set to 0.0 (default) the absolute counter value should be displayed. When a non-zero Multiplier is specified then the value is used to scale the counter value for display. The scaled counter value is used for the count alarm trigger points.

IO/Inputs/din#/Count/SampleTime

Counts accumulate until reset separately by the User using the DCP or other application. This is the case when SampleTime is 0.0 (default). When a non-zero SampleTime is used the counter displays the total count accumulated during that period (in seconds). For instance, with the appropriate combination of Multiplier and SampleTime the counter can display RPM for a strobe input. The alarm trigger points may be set to monitor either high or low count ranges.

IO/Inputs/din#/Count/Alarm1

Set to enable an alarm when the absolute count exceeds the Limit1 count value.

IO/Inputs/din#/Count/Limit1

This defines the trigger point for the count alarm. An alarm (type 1) can be generated with the scaled counter exceeds this value.

IO/Inputs/din#/Count/Alarm2

Set to enable an alarm when the absolute count exceeds the Limit2 count value.

IO/Inputs/din#/Count/Limit2

This defines the trigger point for the count alarm. An alarm (type 2) can be generated with the scaled counter exceeds this value.

IO/Inputs/din#/Alarm1/OnAlarm

This key may be optionally defined and set to “disabled” to disable services related to the occurrence of Digital Input Type 1 Counter Alarms on this input. Counter Type 1 and Type 2 Alarms differ only in the set points used.

IO/Inputs/din#/Alarm1/HoldOff

This defines the amount of time in milliseconds that the Digital Input counter type 1 alarm must remain clear before any subsequent type 1 alarm on this input will be acted upon. The default is 300000 or 5 minutes. When an alarm occurs the services associated with that event are performed. The alarm must reset and remain so for this amount of time before those actions would be performed again.

IO/Inputs/din#/Alarm1/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
IO/Inputs/din#/Alarm1/Email = enabled [, EmailBlock]
```

This key may be set to “enabled” to enable the transmission of Email Notifications in response to a Digital Input counter Type 1 Alarm. If a counter Type 1 alarm Email Notification is desired then this key or the `Events/OnAlarm1/Email` must be defined and “enabled”. Counter Type 1 and Type 2 alarm services differ only in the set points used. An optional EmailBlock may be defined specifying the Email format to be used.

IO/Inputs/din#/Alarm2/OnAlarm

This key may be optionally defined and set to “disabled” to disable services related to the occurrence of Digital Input Type 2 Counter Alarms on this input. Counter Type 1 and Type 2 Alarms differ only in the set points used.

IO/Inputs/din#/Alarm2/HoldOff

This defines the amount of time in milliseconds that the Digital Input counter type 2 alarm must remain clear before any subsequent type 2 alarm on this input will be acted upon. The default is 30000 or 5 minutes. When an alarm occurs the services associated with that event are performed. The alarm must reset and remain so for this amount of time before those actions would be performed again.

IO/Inputs/din#/Alarm2/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
IO/Inputs/din#/Alarm2/Email = enabled [, EmailBlock]
```

This key may be set to “enabled” to enable the transmission of Email Notifications in response to a Digital Input counter Type 2 Alarm. If a counter Type 2 alarm Email Notification is desired then this key or the `Events/OnAlarm2/Email` must be defined and “enabled”. Counter Type 1 and Type 2 alarm services differ only in the set points used. An optional EmailBlock may be defined specifying the Email format to be used.

IO/Inputs/din#/UsageState

This specifies whether usage time is accumulated with the input in the “On” state (1) or in the “Off” state (0). The key is set to ‘1’ or ‘0’ respectively.

IO/Inputs/din#/Usage/Alarm

Set to enable an alarm when the associated $HourMeter usage meter reaches a specified number of hours.

IO/Inputs/din#/Usage/Limit

Defines the alarm setpoint in hours and may include a decimal portion. The associated input goes into alarm when the $HourMeter usage meter reaches or exceeds this setpoint.

IO/Inputs/din#/Usage/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
IO/Inputs/din#/Usage/Email = enabled [, EmailBlock]
```

This key may be set to “enabled” to enable the transmission of Email Notifications in response to a Digital Input Usage Alarm. If a Usage Alarm Email Notification is desired then this key or the `Events/OnUsage/Email` key must be defined and “enabled”. An optional EmailBlock may be defined specifying the Email format to be used.

IO/Inputs/din#/Usage/HoldOff

This defines the amount of time in milliseconds that the Digital Input usage alarm must remain clear before any subsequent usage alarm on this input will be acted upon. The default is 30000 or 5 minutes. When an alarm occurs the services associated with that event are performed. The alarm must reset and remain so for this amount of time before those actions would be performed again.

IO/Inputs/din#/Usage/OnAlarm

This key may be optionally defined and set to “disabled” to disable services related to the occurrence of Digital Input Usage Alarms on this input.

9.17. Configuring Relay Outputs

The following keys are associated with the Relay Outputs. Note: In each of the following Keys replace the ‘#’ with the appropriate channel number 1-8.

IO/Outputs/Log

If the IO/Log key is set to “enabled” a record of output changes is generated. This key can be optionally used to specifically disable logging in general for the Relay Outputs as a group. By default it is enabled.

IO/Outputs/Rout_9-12

The JNIOR model 310 has 8 internal outputs. You can expand the number of outputs by purchasing the 4 Relay Output Expansion Module. You can connect up to 2 of them per JNIOR to obtain output channels 9 – 16. This registry key defines the module address corresponding to channels 9 – 12. This key is automatically assigned when you connect 1 or modules and reboot. If more than one module is present during the reboot then one module is arbitrarily assigned.

IO/Outputs/Rout_13-16

The JNIOR model 310 has 8 internal outputs. You can expand the number of outputs by purchasing the 4 Relay Output Expansion Module. You can connect up to 2 of them per JNIOR to obtain output channels 9 – 16. This registry key defines the module address corresponding to channels 13 – 16. This key is automatically assigned when you connect 2 or modules and reboot. If more than two modules are present during the reboot then one module is arbitrarily assigned.

IO/Outputs/rout#/Desc

This defines the textual description for the associated Relay Output (String). The defaults should be "Relay Out #" where '#' is replaced by '1' through '8' as appropriate.

IO/Outputs/rout#/ClosedDesc

This defines the textual description used when the associated relay has been activated and is in the "Closed" state. "Closed" should be the default.

IO/Outputs/rout#/OpenDesc

This defines the textual description used when the associated relay is in the "Open" state. "Open" should be the default.

IO/Outputs/rout#/Log

If the IO/Log key is set to "enabled" and IO/Outputs/Log key has not been set to "disabled" a record of output condition changes will be written to the /jniorio.log file. This key can be optionally used to disable logging on an output by output basis.

IO/Outputs/rout#/InitialState

This key is used to define the initial behavior of relay outputs on boot up. The value defines a pulse duration in milliseconds where a value of 0 indicates infinity. Setting the key to a value of 0 would effectively close the output. Setting the key to a positive integer would cause the output to pulse for the duration defined by the value An undefined or negative value results in no action.

IO/Outputs/rout#/$HourMeter

This dynamically reports the total number of hours that the individual relay output has physically been in the "Closed" state. This value is non-volatile maintaining content through power loss and until it is specifically reset. It is reported here in hours to the one-hundredth. The Hour Meter is accurate to the millisecond and this high resolution value may be read through the JNIOR Protocol or through the JRMON command.

IO/Outputs/rout#/UsageState

This specifies whether usage time is accumulated when the relay is in the "Closed" state (1) or in the "Open" state (0). The key is set to '1' or '0' respectively.

IO/Outputs/rout#/Usage/Alarm

Set to enable an alarm when the associated $HourMeter usage meter reaches a specified number of hours.

IO/Outputs/rout#/Usage/Limit

Defines the alarm setpoint in hours and may include a decimal portion. The associated relay output goes into alarm when the $HourMeter usage meter reaches or exceeds this setpoint.

IO/Outputs/rout#/Usage/Email

This key may be optionally defined and has two separate values. It is formatted as follows.

```
IO/Outputs/rout#/Usage/Email = enabled [, EmailBlock]
```

This key may be set to "enabled" to enable the transmission of Email Notifications in response to a Relay Output Usage Alarm. If a Usage Alarm Email Notification is desired then this key or the `Events/OnUsage/Email` key must be defined and "enabled". An optional EmailBlock may be defined specifying the Email format to be used.

IO/Outputs/rout#/Usage/HoldOff

This defines the amount of time in milliseconds that the Relay Output usage alarm must remain clear before any subsequent usage alarm on this input will be acted upon. The default is 30000 or 5 minutes. When an alarm occurs the services associated with that event are performed. The alarm must reset and remain so for this amount of time before those actions would be performed again.

IO/Outputs/rout#/Usage/OnAlarm

This key may be optionally defined and set to "disabled" to disable services related to the occurrence of Relay Output Usage Alarms on this output.

IO/Outputs/rout#/Slave

Each **JNIOR** Relay Output can track another Digital Input or Relay Output located on a remote **JNIOR**. In this case the relay output that is tracking is said to be *slaved* to the other I/O point. With proper configuration two **JNIOR** devices can be set up as an I/O relay through the network. In this case perhaps an input on one controls the relay output on the other and vice versa. This function is suitable for very low frequency signals (switches) as there can be considerable delay between the change of an input state and the reaction of the slaved relay. Within a small LAN this might be roughly ½ second but through Internet over some distance there may be a few seconds before the slave response occurs.

The I/O Slave Registry key is used to establish the link with the remote I/O point. This is done as part of the **JNIOR** boot. The connection is maintained and reestablished as would be necessary to continually achieve the tracking. Changes in Slave keys do not take effect until the **JNIOR** is rebooted. Use of the `reboot` command is recommended as this unloads the Registry and insures that all changes are saved prior to restarting the **JNIOR**.

The `IO/Outputs/rout#/Slave` key has the following multi-part format:

```
IO/Outputs/rout#/Slave = IPAddress, nPort, UserName, Password[, IOPoint]
```

Where:

`IPAddress`	Defines the hostname or IP address of the remote JNIOR. This will be processed through DNS if necessary. It is a required entry.
`nPort`	The TCP/IP Port number on the remote JNIOR servicing the JNIOR Protocol. If left blank, 9200 will be assumed.

`UserName` The username of a valid user on the remote JNIOR. This is a required entry.

`Password` The valid password associated with the above username. JNIOR must be able to successfully log into the remote JNIOR in order to track its I/O. This is a required entry.

`IOPoint` Defines the remote I/O point to track on the above JNIOR. This has to be precisely one of the following:

din1	rout1
din2	rout2
din3	rout3
din4	rout4
din5	rout5
din6	rout6
din7	rout7
din8	rout8

If omitted the Digital Input corresponding to the associated `rout#` will be assumed.

For instance the following entries cause Relay Output 2 and Relay Output 3 to track the state of Digital Input 1 on the **JNIOR** at 10.0.0.223:

```
IO/Outputs/rout2/Slave = 10.0.0.223, 9200, jnior, jnior, din1
IO/Outputs/rout3/Slave = 10.0.0.223, 9200, jnior, jnior, din1
```

Note that multiple relays can track the same I/O point and relays can track I/O points on multiple **JNIOR** units. The requirement is that all of the **JNIOR**s are running a version 2.01.323 or later Operating System and have the JNIOR Protocol enabled.

Should communications be lost and the **JNIOR** is unable to reestablish the connection with several seconds the Relay Outputs affected will go into Safe Mode and turn off (open). These outputs will again reflect the status of the remote I/O Points once the connection can be reestablished. Refer to the `jniorsys.log` file for a record of events associated with connection problems.

IO/Outputs/rout#/$TriggerCount

NOTE: This key was IO/Outputs/rout#/TriggerCount prior to release 3.3.x.y

Each **JNIOR** Relay Output can be configured to close upon reaching a defined number of counts on its associated Digital Input (din1 for rout1, din2 for rout2, etc.). If defined this Registry Key enables this high speed control feature and specifies the desired trigger count. For instance:

```
IO/Outputs/rout8/$TriggerCount = 1500
```

Assuming that the input counter for Digital Input 8 has been cleared, Relay Output 8 will close immediately upon seeing the 1500th low-to-high transition on Digital Input 8. The relay closes in well under 1 millisecond of receipt of the triggering transition. **JNIOR** can process input pulses at a rate as high as 2,000 per second. Input counters are 4-byte integer values which can tally as many as 4.3 billion transitions.

Once the relay closes it will remain closed until commanded by the user's application to open (or power is lost). If the user's application does not increase the TriggerCount value or reset the Digital Input Counter, the relay will again close on the next low-to-high transition of the associated input. The relay is commanded to close on each and every low-to-high transition resulting in an input counter value equal to or greater than the defined trigger count.

Setting TriggerCount to 0 or removing the key entirely for any relay disables this feature for that relay. By default the feature is disabled on all relays. Relays will remain closed until commanded separately by the user's application or manually through the DCP or other means. The trigger count is configured by writing the Registry Key. This occurs at the moment the key is written. This is true now for all I/O configurations whereas previously (prior to v2.13.83) it may have taken up to a minute after adjusting Registry settings for the I/O configuration to update. This is true no matter how the Registry Key is written either through a protocol, the DCP, Registry Editor, or by command at the command prompt.

IO/Outputs/rout#/ForcePulseDuration

Each **JNIOR** Relay Output can be configured to pulse for a configured number of milliseconds instead of performing the requested close, open, or toggle commands. This will act like a failsafe and was implemented to provide for a pulse when a pulse command cannot be sent.

This key, if defined, causes the JNIOR to pulse the output for the specified number of milliseconds instead of doing the requested open or close. If an open was requested then a pulse low occurs. Alternatively if a close was requested a pulse high will occur.

9.18. Sensor Port

One or more One-Wire devices may be connected through the external one-wire port or *Sensor Port*. JNIOR detects these devices during boot and updates the Registry to identify new devices and information regarding those currently available. The following Registry keys apply. *Note: <address> represents the 16-character device address where appropriate below.*

OneWire/Devices

This entry contains the list of currently active (detected as of the most recent boot) One-Wire devices. This is a list of <address> separated by commas. If there are no devices available then the key will not be present.

OneWire/<address>/Desc

This provides a text description for the device whose address is <address>. The entry is created automatically when a new device is detected during boot. It can be edited as appropriate. The entry will remain in the Registry even after this device is removed. It can be manually deleted.

OneWire/<address>/Name

This provides a list of device names (DS1920 for a temperature sensor for instance) that are appropriate for the device type. The entry is created automatically when a new device is detected during boot. The entry will remain in the Registry even after this device is removed. It can be manually deleted.

OneWire/<address>/Type

This indicates the *type* of One-Wire device. For instance `Type=10` for a class of Temperature sensors.

OneWire/<address>/Mount

This defines the Flash Memory mount name for external devices containing Flash Memory support. For example a setting of '`panel`' for a User Panel with the Flash Memory option will locate the files contained therein under the folder `panel/`. Note that if left undefined the external flash memory devices will be mounted in folders named: `extrn/` `extrn1/`, `extrn2/`, etc. External flash devices are automatically mounted during boot when they are discovered. See the `jrflash` command for information on maintaining flash memory devices and mounting devices at other times.

OneWire/<address>/$Value

For certain devices (notably the Type 10 Temperature Sensors) this key will indicate the current value. In the case of a temperature sensor it will contain both the current Fahrenheit and Celsius readings. Note that '$' keys are not posted to the jnior.ini file. These are dynamic values.

OneWire/<address & 0x28>/TempResolution

This key is used to set the resolution for the DS18B20 one wire device, type 0x28. The valid values are 9, 10, 11, and 12. They represent the number of bits used to calculate the temperature. A smaller value will cause the conversion time to be faster but less precise. The following table shows the effect of changing the resolution.

Resolution	9 bit	10 bit	11 bit	12 bit
Conversion Time (ms)	93.75	187.5	375	750
LSB (°C)	0.5	0.25	0.125	0.0625

9.19. Default Key Settings

Registry Keys may be defined or undefined. An *undefined* key is one that has not been specifically assigned a value in the current Registry. In many cases the absence of a value for a key implies a *default*. In other situations the absence of the value defers the function to another key. Certain key settings may *override* others.