NAME

ConfigTool

SYNOPSIS

```
Configuration tools for MMC and SIAM
```

DESCRIPTION

Configuring MMC hardware and the SIAM application can be complicated, since the parameters are represented in a large number of files, which include Linux system files as well as application specific configuration files. Also, the logic for setting up a SIAM node is fairly complicated. There are also special configuration requirements, including the ability to configure a node via a serial console (i.e., in the absence of an ethernet network), and that the configuration tool be cross-platform.

ConfigTool is an application that addresses these requirements by presenting configuration parameters in logical groups, masking both the distributed nature of the configuration and the logic required to set up a SIAM node. The **ConfigTool** consists of a Jetty web server and Java servlet container and the ConfigTool Java servlet which implements the configuration logic and generates the user interface. The UI may be operated using most common web browsers, including the **links** text-based browser which may be used over a serial console.

Currently, the ConfigTool reads and writes configuration files, some of which are Linux system files. Changing these files while the application is running is not advised; **Do not run the ConfigTool server while the SIAM application is running**.

Operation Overview

To use ConfigTool, the server must be started from the MMC command line. Once the server is running a browser is pointed to the server's URL to access the user interface. If the server is being run from a serial console or in a single ssh session, it may be run in the background to allow a browser to run on the same console. In general, since the ConfigTool server requires a JVM, in order to conserve memory it is not allowed to run when not in use .

Basic Operations

Starting ConfigTool Server	Exit the SIAM application before starting the ConfigTool server.
	The ConfigTool server must be run as root.
	Usage: **`configTool -[wml] [-p <configtool properties>]`**
	w: x86-Win32
	m: MMC
	l: x86-Linux System
	p: properties file [default is $SIAM_HOME/properties/configtool.properties]
	Starting the configTool server When running in the background, redirect output to a file to allow a text browser to run in the same session, e.g.:
	`configTool -w -p myconfigtool.properties >& server.log &`
	`configTool -m >& /dev/null &`

Starting the User Interface	The user interface is accessed via web browser, by pointing to the configTool servlet URL:
	Start links browser on localhost (in a different console session from the server or with server running in the background):
	`links http://localhost:8181/config`
	Configure host node137:
	`http://node137:8181/config`
	Configure host by IP address:
	`http://137.89.4.137:8181/config`

Stopping the User Interface	Stopping the user interface is done simply by exiting the web browser or directing the browser to another page. To exit the links browser, use **`ALT-F x`** or **`ESC f x`** (ESC invokes the links menu bar, f selects the file menu, x exits links)
Stopping the ConfigTool Server	To stop the ConfigTool server, use the "Exit ConfigTool" button on the user interface main page. It is also possible to use **`CTRL-C`** or **`kill <JVM PID>`** to stop a server running in the foreground or background, respectively.

Navigation Overview

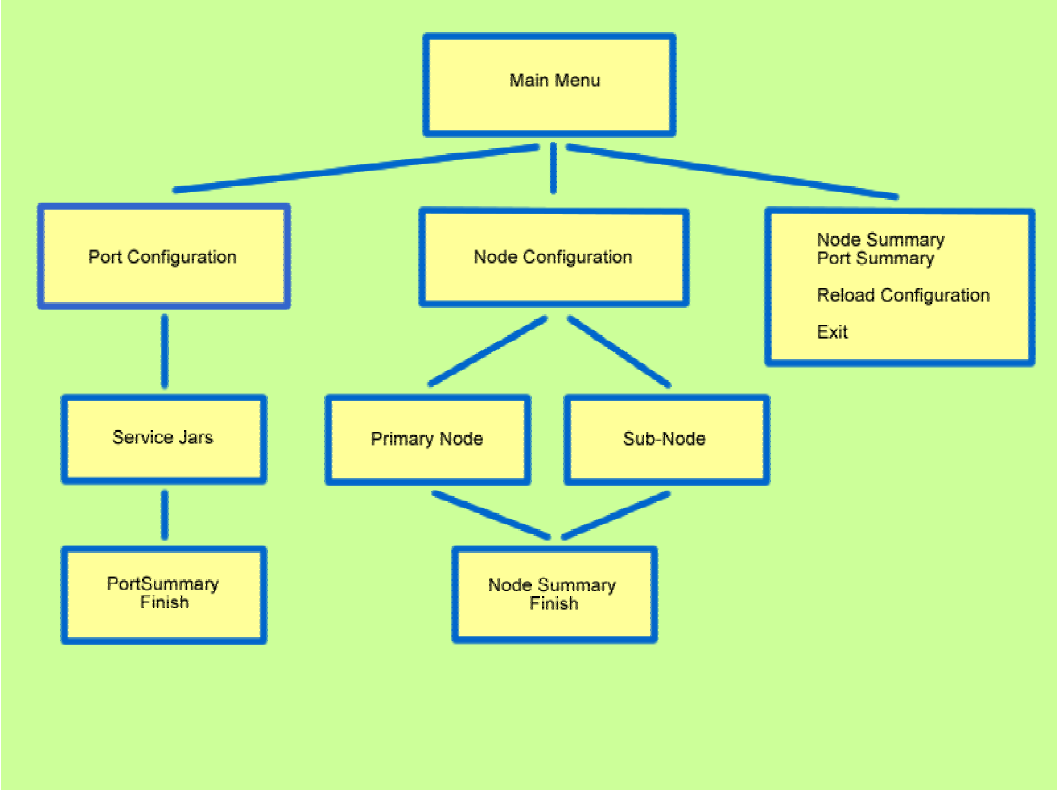
The ConfigTool is intended to make the configuration process as easy as possible by organizing configuration processes into a set of simple "wizards". Each wizard

guides the user through a series of related configuration options.

The first time the server is contacted after it is started, it makes a copy of the current state represented by the relevant configuration files. Once the current configuration is loaded, changes to the server's copy of the settings persist as long as it is running, even across multiple (web browser) editing sessions. Changes are not written back to the configuration files until the user selects the "finish" option from one of the wizard menus, thus the settings represented by the ConfigTool wizard may not represent the actual state of the system. The current state of the configuration settings can be reloaded at any time using the "Reload Configuration" option from the main menu.

There are currently two wizards: one for node hardware and software configuration options, and another for configuring the MMC serial ports. In addition, it is possible to review a summary of the current configuration settings. The wizards are context sensitive and dynamically change the options presented based on current choices. Users manipulate HTML form controls to change configuration settings.

Once a wizard's options have all been set, it presents of a summary of the options configured and allows the user write the changes back to the configuration files, cancel or go back to make further changes. Once the user chooses to write the configuration changes using any of the wizards, all files are updated (all wizards use the same process, writing all of the files). Each time it writes the configuration files, ConfigTool makes a copy of each configuration file named *<configFileName>.mmddyy.bak*, saving the last version written on a given day.



Main Menu Options	• Configure Node • Configure Ports • Reload Configuration • Current Node Settings • Current Port Settings • Exit ConfigTool

Using links Browser	• Up/Down arrows: previous/next link on page
	• Left/Right Arrows: previous/next page
	• ESC: menu
	• CTRL-R: reload current page
	• Controls
	  • Buttons
	  • Dropdown Box
	  • Checkbox
	  • Text Box
	• ESC-F-x: exit links
	• Links menu options File, View, Link, Downloads, Setup, Help

Configuration Parameters

The table below describes the configuration parameters exposed by the ConfigTool user interface. Some of these parameters map directly into configuration files; a number of other system configuration parameters are generated internally by ConfigTool, based on selected settings.For a more

complete mapping of system configuration parameters, see diagram in ConfigToolParamMap.pdf

Configuration Parameter	Description	Values
nodeType	Nodes may be either primary nodes or subnodes. On a MOOS mooring network, there is one primary node, which acts as a gateway connecting the mooring network to the shore network. The other nodes are subnodes, which differ in their physical connection to the primary node or other subnodes; ethernet (fiber optic) and RS-422 (copper) connections are supported.	primary_node ETH_subnode RS422_subnode
nodeID	The node ID is a positive (long) integer which comprises a unique identifier within the SSDS data system. This number may be obtained from the SSDS ID database (http://ssdspub.mbari.org:8080/access/index.jsp).	long integer > 0
ethernetEnabled	If this parameter is set to TRUE, the ethernet networking will be brought up when the node is booted.	TRUE FALSE
autostart	If this parameter is set to true, the SIAM node application will be started automatically when the node is booted.	TRUE FALSE
debugLevel	This setting determines the amount of debug information provided and logged by the SIAM application when it is running. In order of increasing information, the supported levels are ERROR, WARN, INFO and DEBUG. This parameter affects the amount of storage space required for the node logs; DEBUG should not be used for deployment.	ERROR WARN INFO DEBUG
ethernetAddressBase	This parameter comprises the first two octets of the mooring network addresses used for the mooring network (e.g., 134.89. for a mooring network connected to the MBARI network). A complete mooring network IP addresses consists of *<ethernetAddressBase>.<networkID=networkNumberBase+NetworkNumber>.<nodeOUI>* Where node OUI is the 6th octet of the MMC ethernet controller MAC address. The networkID is used to create a range of network addresses used by different MOOS mooring networks.	N.N. 1<N<255
sleepManagerEnabled	When sleepManagerEnabled is TRUE, the SIAM application manages power by suspending processor operation whenever there is no activity scheduled for some minimum period of time.Otherwise, the processor runs continuously. Note that when the processor is sleeping, the console does not respond to input. Instrument serial ports may be configured to wake up the processor upon receiving characters (this is done by setting the appropriate bits of the arguments to the setPWER command in suspend.sh, but is not currently brought out through the ConfigTool interface),	TRUE FALSE
useDHCP	When useDHCP is TRUE, code is included in rc.local which switches the ethernet interface (eth0) to DHCP, and subsequently modifies the hosts file to correctly map the hostname to the newly obtained IP	TRUE FALSE

	address. This is used for connecting a mooring network to the building network and is not currently supported for deployed systems.	
wirelessNetwork	Primary nodes may specify a wireless telemetry link to shore.Currently globalstar satellite modem, freewave 900 MHz radio modem or null modem RS-232C are supported. The comms manager (telemetry subsystem) settings are automatically configured appropriately for the selected wireless link. This setting applies only to primary nodes.	none globalstar freewave null modem
networkNumberBase	The third octet of a node's IP address is obtained by summing the networkNumberBase and the networkNumber (below), resulting in a networkID. The networkID is used to create a range of network addresses used by different MOOS mooring networks. The networkNumber base may depend on, for example, the building network that the mooring needs to connect to. For MBARI, the network number base 35; network addresses A complete mooring network IP addresses consists of *<ethernetAddressBase>.<networkID=networkNumberBase+NetworkNumber>.<node OUI>* `Where node OUI is the 6th octet of the MMC ethernet controller MAC address.` This setting applies only to primary nodes.	0 < integer < 255
networkNumber	Network number is combined with the networkNumberBase to create the networkID for the mooring network. The networkID is used to create a range of network addresses used by different MOOS mooring networks. A complete mooring network IP addresses consists of *<ethernetAddressBase>.<networkID=networkNumberBase+NetworkNumber>.<node OUI>* `Where node OUI is the 6th octet of the MMC ethernet controller MAC address.` This setting applies only to primary nodes.	integer > 0
Service Type	The Service Type parameter indicates the source of the instrument service code for a particular instrument serial port. When set to PUCK, the SIAM application will query the associated serial port using the PUCK protocol to obtain instrument service code. When set to JAR, the Service JAR parameter specifies the name of a Java Archive (JAR) file containing insturment service code. If set to DOWNLINK, the associated serial port will be used to connect to a subnode. If set to NULL, the port is unused.	PUCK JAR DOWNLINK NULL
Service Jar	The name of a Java Archive (JAR) file containing SIAM instrument service code for a specified instrument serial port.	JAR file name

Configuration File Reference

Related Configuration File(s)	Description	References	Modifies
siamPort.cfg	Sets SIAM application configuration parameters	-	serialPortN powerPortN

			serviceJarN nodeID nodeType networkNumberBase networkNumber ethernetAddressBase. advertiseService wirelessLink ethernetEnabled useDHCP jarLocation logLocation platformSerialPorts codeBaseLocation SleepManager.enabled SleepManager.sleepString #SleepManager.pollSeconds SleepManager.sleepDelay #SleepManager.wakeupSeconds #SleepManager.minSleepSeconds NodeService.leaseInterval CommsManager.enabled CommsManager.onString CommsManager.offString CommsManager.protocolWaitTime CommsManager.processWaitTime CommsManager.leaseRenewalInterval
siamEnv	Sets environment SIAM application and ops user	/j9/j9env	. /j9/j9env JAVA JAVA_OPTIONS SIAM_BIN SIAM_CODEBASE PARENT_HOST SIAM_CLASSPATH SIAM_AUTOSTART LOG4J_THRESHOLD=DEBUG PATH SIAM_PPP_SCRIPT ETHERNET_ADDRESS_PREFIX
rc.local	Performs user defined system configuration operations at the end of the boot process	mount /etc/siam/manageLog /var/log/syslog /root/ricohRTC /proc/sys/net/ipv4/ip_forward source /root/.bashrc source /etc/siam/siamEnv $SIAM_HOME/properties/hosts.template /etc/hosts /etc/hostname /usr/bin/ifswitch-to-dhcp $SIAM_HOME/bin/arm-linux/ipaddr ifdown ifup PRIMARY_NODE_NAME PRIMARY_NODE_ADDRESS ETHERNET_ADDRESS_PREFIX	/var/log/syslog /var/log/messages /var/lock /var/lock /root/ricohRTC -s /proc/sys/net/ipv4/ip_forward /usr/bin/ifswitch-to-dhcp /etc/hostname /etc/hosts eth0

hosts	Maps known host names to network IP addresses	PRIMARY_NODE_NAME PRIMARY_NODE_ADDRESS ETHERNET_ADDRESS_PREFIX	127.0.0.1 localhost loc local familiar 10.1.1.1 shore # portal node's ppp interface address 10.1.1.2 surface-rf mooring # surface node's ppp interface address 134.89.37.200 surface # surface node's ethernet address (static) # Subnode names have sidearm OUI appended 134.89.37.1 node401 node1 134.89.37.2 node402 node2 134.89.37.3 node403 node3
hostname	Defines the name of the local ethernet interface	-	hostname ("surface")
binRouting.sh	Defines network routing for surface-shore ppp network	PRIMARY_NODE_NAME	route add -net 10.1.1.0 netmask 255.255.255.0 gw surface
interfaces	Defines ethernet interfaces	PRIMARY_NODE_ADDRESS	auto lo iface lo inet loopback auto eth0 iface eth0 inet static address 134.89.37.200 netmask 255.255.255.00 gateway 134.89.37.200
longhaul	Defines primary wireless link ppp parameters	/etc/siam/rfpower.ops	-
shorthaul	Defines secondary wireless link ppp parameters	-	-
auxTelem.sh	Periodically enables secondary wireless link	/etc/siam/siamEnv /etc/siam/cpuAwake /etc/siam/rfpower	
j9env	Defines environment for j9 JVM	PATH JAVA_HOME LD_LIBRARY_PATH	JAVA_HOME PATH LD_LIBRARY_PATH
.bashrc	Defines environment for ops user	/etc/bashrc SIAM_HOME	PS1='\h:\w\$ '
suspend.sh	Suspends processor operation as part of power managment	/bin/sync /root/setPWER /root/suspend /root/ricohRTC # ifdown eth0 # ifup eth0	# Set PWER # The current bits set are: # 0x80000000 = SA1100 RTC (/root/suspend needs this) # 0x8 and 0x4 = Exar UARTs (/dev/ttySX{0-15}) # 0x1 = MSP430 # See the Sidearm4 documentation on the the assignment # of GPIO bits. Set additional bits as necessary. /root/setPWER 80000001 /root/suspend -v $sleepTime /root/ricohRTC -s

ConfigTool Servlet Configuration

The stable release of the ConfigTool servlet does not require configuration; by default it is configured to run on an MMC node. When bench testing on other platforms (x86-win32/cygwin or x86-linux), the servlet needs to be configured appropriately. The ConfigTool servlet is configured via a set of Java properties files (specifying paths to the required files and executables) and configuration file templates (defining configuration-file-specific parameters and defining the configuration items in each configuration file and the user interface controls used to represent them).

configtool.properties		
Property	**Description**	**Comments**
jarPath	Specifies directory where instrument service JAR files are located	relative to siam_home example: jarPath=ports/
serialPorts	Number of serial ports to configure.	The ConfigItems for each serial port must be defined in <configFile>.properties, and there must be already be entries in siamPort.cfg for all serial ports and their power ports.
<configFile>.type=<TYPE>	Declares a configuration file named <configFile> of type <TYPE> to the configuration servlet. The <TYPE> string identifies which config file should be represented in the user interface. the <configFile> name is used to specify the properties file (see properties.<configFile>) and the location of the configuration file (see config.<configFile>)	Currently supported types are: SIAMPORT SIAMENV RCLOCAL HOSTS HOSTNAME INTERFACES BINROUTINGSH example: siamEnv.type=SIAMENV
properties.<configFile>	Specifies the (configuration file specific) properties used by the ConfigFile object and defines the ConfigItems represented by the ConfigFile, including the user interface controls used to represent them in the user interface. This allows some aspects of the user interface to be modified without recompiling any code.	relative to siam_home example: properties.rcLocal=properties/rclocal.properties
siam_home	Specifies the path to the top level SIAM directory.	siam_home is an absolute path and should match the $SIAM_HOME environment variable. example: siam_home=/mnt/hda/siam/
ipAddressCommand	Specifies path to a command that will return the system's IP address. This is, in general, a native binary, and resides in different directories within the SIAM directory structure, depending on the platform.	relative to siam_home example: ipAddressCommand=bin/arm-linux/ipaddr
ethernetAddressCommand	Specifies path to a command that will return the system's MAC address. This is, in general, a native binary, and resides in different directories within the SIAM directory structure, depending on the platform.	relative to siam_home example: ethernetAddressCommand=bin/arm-linux/ethaddr
scriptRunner	Specifies path to a command that will run scripts on the platform. This is needed to allow scripts to be executed from within Java, as Runtime.exec() only works correctly on native executables. This is, in general, a native binary, and resides in different directories within the SIAM directory structure, depending on the platform.	relative to siam_home example: scriptRunner=bin/arm-linux/commandRunner
actionScript	Specifies the path to the post-configuration script that should be run after modifying the configuration files. Note: this script is run as root. Damage to system could occur if the wrong script is specified, or if the script is modified.	relative to siam_home example: actionScript=utils/configAction
actionPath	Specifies the PATH environment variable for the action script.	absolute path, does not resolve $PATH environment variable example: actionPath=.:/bin:/sbin:/usr/bin:/usr/sbin:...
config.<configFile>	Specifies the path to the actual configuration file to be modified or replaced.	config paths are absolute example: config.siamPort=/mnt/hda/siam/properties/siamPort.cfg

Configuration File Specific Properties

In general, each configuration file has a set of parameters that are represented (brought out) to the ConfigTool user interface. Internally, these parameters are represented as ConfigItems, which manage the value of their parameters and maintains an HTML control in the user interface to that is used to set their value. The ConfigItem must be configured to read (import) and write (export) its parameter's value from/to the configuration file. The HTML control it displays to set the parameter value must also be defined and configured. Each configuration file uses a Java properties file to define and configure its ConfigItems and HTML controls.

Property entries in the configuration file properties files have the general form:

```
<ConfigFile>.<configItem>.pageID=<htmlPageName>
```

Additionally, configuration file properties files may define other parameters and properties used internally.

<configFile>.properties		
Property	**Description**	**Comments**
<ConfigFile>.<configItem>.pageID=<htmlPageName>		example: `SiamPortCfg.nodeType.pageID=NodeMain`
<ConfigFile>.<configItem>.controlType=<CONTROL_TYPE>		example: `SiamPortCfg.nodeType.controlType=SELECT`
<ConfigFile>.<configItem>.validateRegexp= <regexp>		example: `SiamPortCfg.nodeType.validateRegexp=`
<ConfigFile>.<configItem>.values=<value>, <value>,...		example: `SiamPortCfg.nodeType.values=primary_node,ETH_subnode,RS422_subnode`
<ConfigFile>.<configItem>.importRegexp=<regexp>		example: `SiamPortCfg.nodeType.importRegexp=^[ \\t]*nodeType[ \\t]*=[ \\t]*`
<ConfigFile>.<configItem>.exportRegexp=<regexp>		example: `SiamPortCfg.nodeType.exportRegexp=nodeType = <VALUE>`
<ConfigFile>.<configItem>.attribute.ROW=<row>		example: `SiamPortCfg.nodeType.attribute.ROW=1`
<ConfigFile>.<configItem>.attribute.COL=<col>		example: `SiamPortCfg.nodeType.attribute.COL=0`
<ConfigFile>.<configItem>.attribute.<attributeName>=<value>		
<ConfigFile>.<configItem>.itemRef=<configItemName>		example: `Tags.tagHostname.itemRef=hostname`
<ConfigFile>.<configItem>.valueRef=<configItemName>		example: `SiamPortCfg.serviceJar0.valueRef=jarList`

APPENDICES

MMC Serial Ports

Serial Port Function	Device Name
Console port	/dev/ttySA0
Synchronous serial port	/dev/ttySA1
Debug port	/dev/ttySA2
Instrument backplane serial ports	/dev/ttySX0 through /dev/ttySX11
RF modem/transceiver port 0	/dev/ttySX13 (RFIO connector closest to edge of sidearm)
RF modem/transceiver port 1	/dev/ttySX12
Uplink RS-485 port	/dev/ttySX14
MSP-430	/dev/ttySX15

Configuring Power Management

Set PWER
Set PWER
The current bits set are:
0x80000000 = SA1100 RTC (/root/suspend needs this)
0x8 and 0x4 = Exar UARTs (/dev/ttySX{0-15})
0x1 = MSP430
See the Sidearm4 documentation on the the assignment
of GPIO bits. Set additional bits as necessary.
/root/setPWER 80000001

Configuring PPP Networking

ConfigTool Internals Reference

BUGS

TO DO

- Make the ConfigTool server embeddable and run it in a separate thread from the SIAM Application.
- Create an error handler that redirects exceptions to the browser

ENVIRONMENT

The following environment variables must be set:

- JAVA - name of the JVM. E.g. on a Sidearm, JAVA should be set to *j9*. On a workstation with Sun's Java installed, JAVA could be set to *java*.
- SIAM_HOME - top of the SIAM directory tree
- SIAM_CLASSPATH - on a Sidearm, SIAM_CLASSPATH should be set to *$JAVA_HOME/lib/jclMax/classes.zip:$JAVA_HOME/lib/prsnlmot.jar:$JAVA_HOME/lib/RXTXcomm.jar:$SIAM_HOME/classes*. On X86 system, SIAM_CLASSPATH should be set to *$SIAM_HOME/classes*
- LD_LIBRARY_PATH - path to j9executables; typically /j9/bin

SEE ALSO

SIAM installation