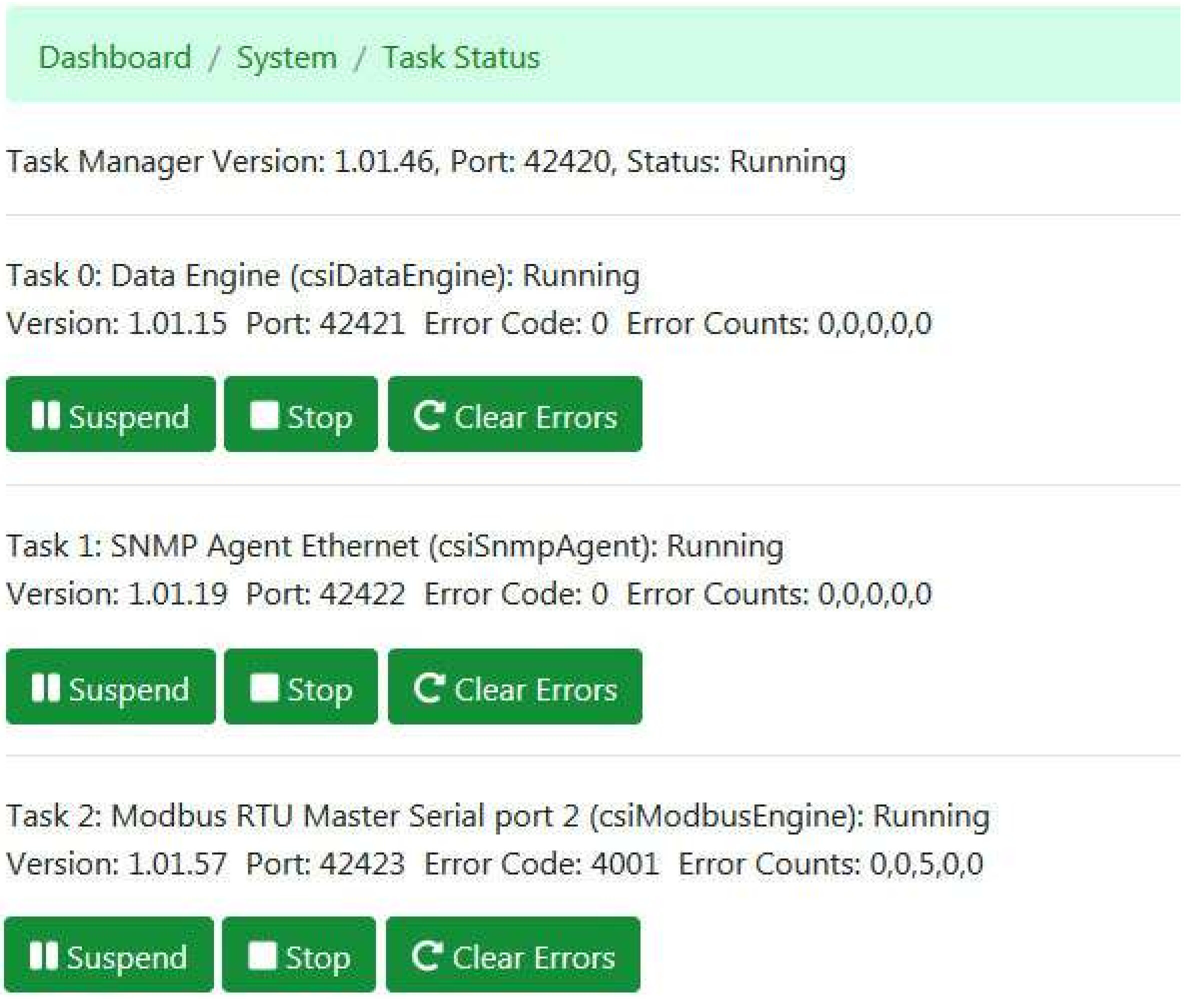
BB4-8422 System - Task Manager

Task Status



Task Manager Version: 1.01.46, Port: 42420, Status: Running

Task 0: Data Engine (csiDataEngine): Running
Version: 1.01.15 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Task 1: SNMP Agent Ethernet (csiSnmpAgent): Running
Version: 1.01.19 Port: 42422 Error Code: 0 Error Counts: 0,0,0,0,0

Task 2: Modbus RTU Master Serial port 2 (csiModbusEngine): Running
Version: 1.01.57 Port: 42423 Error Code: 4001 Error Counts: 0,0,5,0,0

This page shows each task currently configured, and whether running, suspended, or stopped. If running, you may click the Suspend or Stop buttons to halt that task. The buttons will then change to Resume or Start to resume or restart the task.

Click the Clear Errors buttons to clear/reset the error codes or counts if any are indicated. This is optional, and useful to see if any new errors occur later.

The software version of each task is displayed here for reference. The port number used by the internal API is listed for diagnostic reference, and is not normally used directly by a user.

The error code is zero if no error, or any of the following:

Error codes 1001-1999 are reserved for database related errors.

- 1001 = internal error - sqlite_prepare failed, malloc failed, etc
- 1002 = failed to open DB
- 1003 = error while attempting to create table
- 1004 = error reading
- 1005 = error writing (insert/update, delete)
- 1006 = SQL malloc failed
- 1007 = failed to lock database

Error codes 2001-2999 are reserved for task management related errors.

- 2001 = failed to initialize
- 2002 = task initialization malloc failed
- 2003 = the I'm alive timeout expired
- 2004 = problem with global config file
- 2005 = attempt to open error log file failed
- 2006 = error in global configuration
- 2007 = internal error - shm size, pid problem, invalid task state, etc
- 2008 = invalid license file, task suspended

Error codes 3001-3999 are reserved for universal application errors.

- 3001 = empty configuration
- 3002 = invalid configuration
- 3003 = validation failed (invalid parameters, out of range, etc)
- 3004 = null pointer error

Error codes 4001+ are open for application specific use.

Task manager error codes specific to SNMP:

- 4001 = communication error has been reported

Task manager error codes specific to Modbus:

- 4001 = communication error has been reported
- 4002 = communication with coprocessor has faulted
- 4003 = coprocessor communication configuration fault
- 4004 = unable to open port to coprocessor
- 4005 = coprocessor firmware version wrong, can't run with this
- 4006 = failed to hard reset coprocessor

NOTE: Error 4001 is most often associated with a client having difficulty polling a server in a read map or write map. Additional error information will be available on the client's status page. If error 4001 appears for a Modbus TCP server task, it will most often mean more than one TCP server has been configured and they are trying to open the same network port number. Only one server can be listening on any given port.

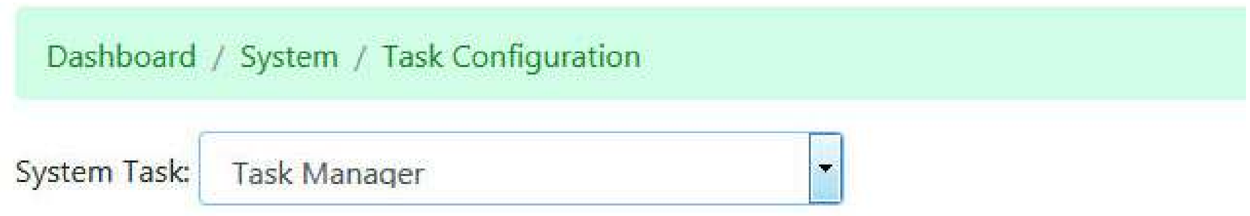
The error count information will be a sequence of five numbers a,b,c,d,e where:

- a = tally of task lock timeouts
- b = tally of object lock timeouts
- c = tally of communication errors currently existing
- d = indication of task initialization error
- e = reserved for future use

You prefer to see nothing but 0,0,0,0,0 here. If non-zero values are indicated, the STA LED will be flashing yellow or STB will be flashing red. If the Clear Errors button is clicked, the error indications will be reset to zero and the STA and STB LEDs should resume their green heartbeat blink.

Task Configuration

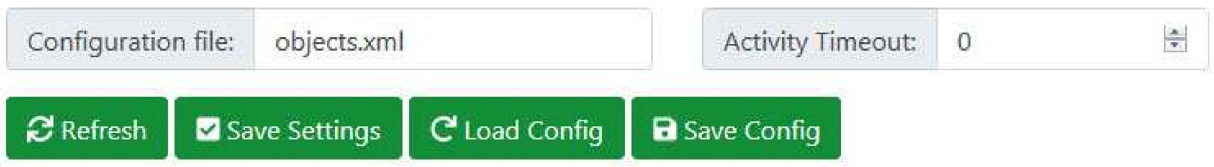
Task Manager Interface



The task manager is your first stop for telling the IoTServer what you want it to do. There are 32 slots that may be occupied by some function, typically a protocol engine or other application. This means you could hypothetically have up to 30 different protocols all talking to each other via the IoTServer, if we actually had that many protocols to pick from.

Select the task you want to configure from the drop-down list, or select an empty slot to assign to a new function. An empty slot is simply denoted "Data Channel N". When assigned to a function, that slot will take on the name of the function.

Different protocols, applications, or task types will have different requirements for configuring the task. The specific configuration parameters for each available task function are given at the links below. Regardless of task type, each task will have certain attributes in common as described here.



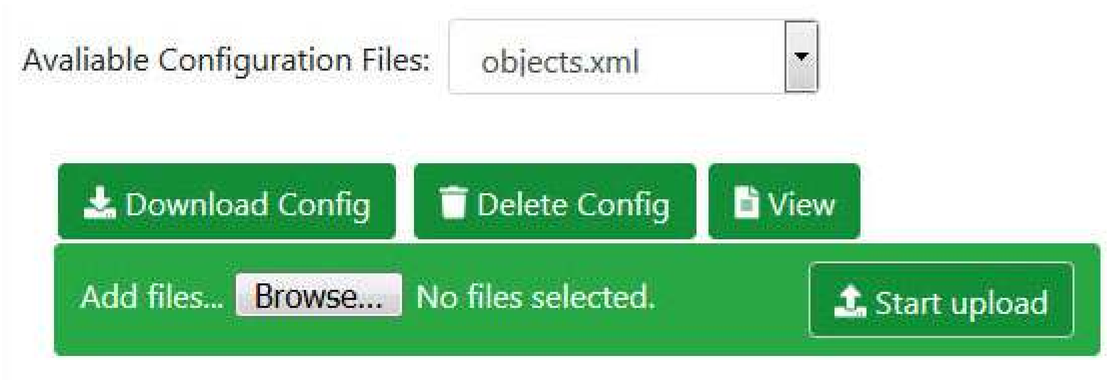
Click the Refresh button to have the web UI query the actual task and obtain its present configuration parameters.

If you change any of the settings, click Save Settings to retain and register them. In some cases it may be necessary to stop the task first. In almost all cases, it is necessary to restart the task to cause the new settings to take effect.

The file selected from the drop-down list below will appear in the Configuration File box. If you want to configure the task from an XML file, select that file, and click the Load button. In the case of protocol engines, this will most often include loading all of its read/write maps, rules, etc. If the task was suspended, clicking Load will also automatically resume the task after loading the XML file as its new configuration.

To save configuration for the given task, select or enter a file name, and click the Save Config button. In the case of a protocol engine, this means saving its read/write maps, rules, etc.

IMPORTANT: The task manager configuration, i.e. settings registered by clicking Save Settings, are retained in the XML file for the task manager itself. To save these settings in an XML file, select Task Manager from the list at the top, and click Save Config there.

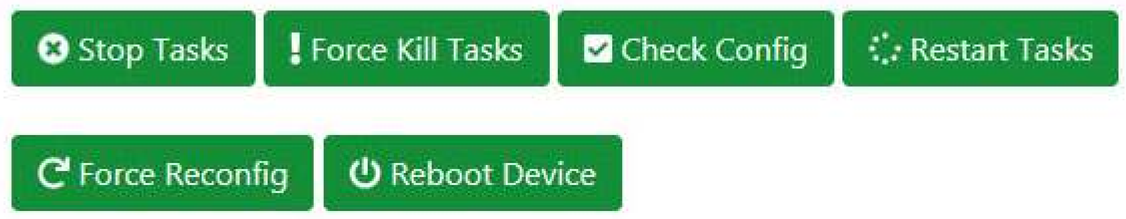


The file management provided here is most often mirrored on the Config File page of the given task. The drop-down list will show a list of all configuration files currently found in the device's configuration folder. When you select an XML file from this list, the name will be copied to the Configuration File window for pending load or save.

You may view the selected file by simply clicking View. You can delete the file by clicking Delete.

You may upload files to the IoTServer from your PC. Start by clicking Browse, and then use the browser's file dialog to locate the file on your PC. Once a file is selected on your PC, click the "Start upload" button to initiate the transfer.

You may also download files from the IoTServer to your PC. Click the Download button to transfer the selected file to your PC.



Stop Tasks - Stops all tasks except the task manager itself. If you want to selectively stop only one task, you can do that from the Task Status page.

Force Kill Tasks - The "stop task" is a request submitted to the task asking it to stop itself. However, if a task is hung in a manner that prevents it from responding to that task, this button will forcefully kill the tasks that remain in a state other than stopped after attempting to stop them.

Check Config - There are certain configuration restrictions such as you cannot have more than one SNMP Trap Receiver configured in the system, cannot have more than one Primary Modbus Server, etc. Clicking Check Config will check to see if you have configured anything that creates a conflict, and provide a message to that effect if so.

Restart Tasks - Restarts all configured tasks that have been stopped.

Force Reconfig - Will delete all internal configuration databases and force reconfiguration from XML configuration files. Tasks should first be stopped - this action will not be permitted if tasks are still running. USE WITH CAUTION. If you have made configuration changes that were saved to the database but not explicitly saved to an XML file, those changes will be lost with Force Reconfig.

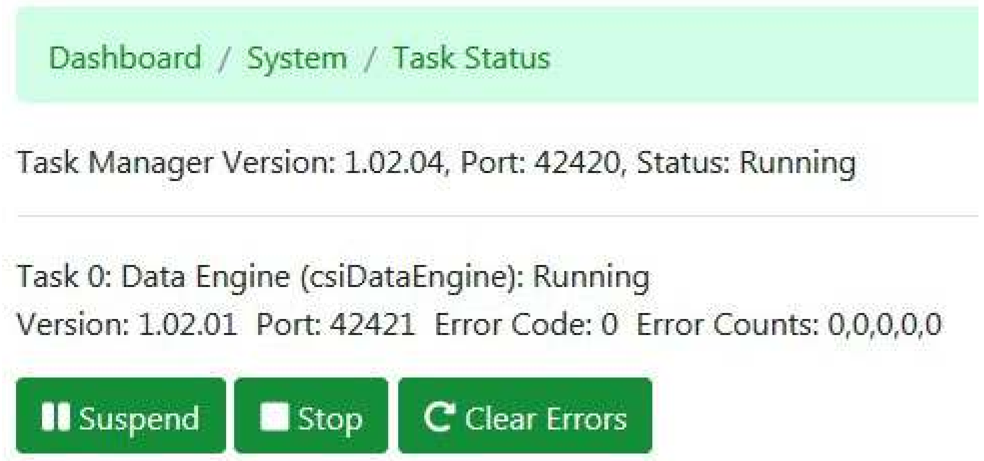
Reboot Device - This will gracefully shut down the system and reboot. It will accomplish the same thing as a power cycle, but do so more gracefully.

Initial Configuration

The IoTServer (Babel Buster 4) will arrive with only the Data Engine configured to start on power-up. The configuration library provided will be a minimal configuration for each possible protocol. These serve as a starting point to bring up that protocol. Once running, you can proceed to edit or upload additional more extensive configurations.

Initial Task Startup

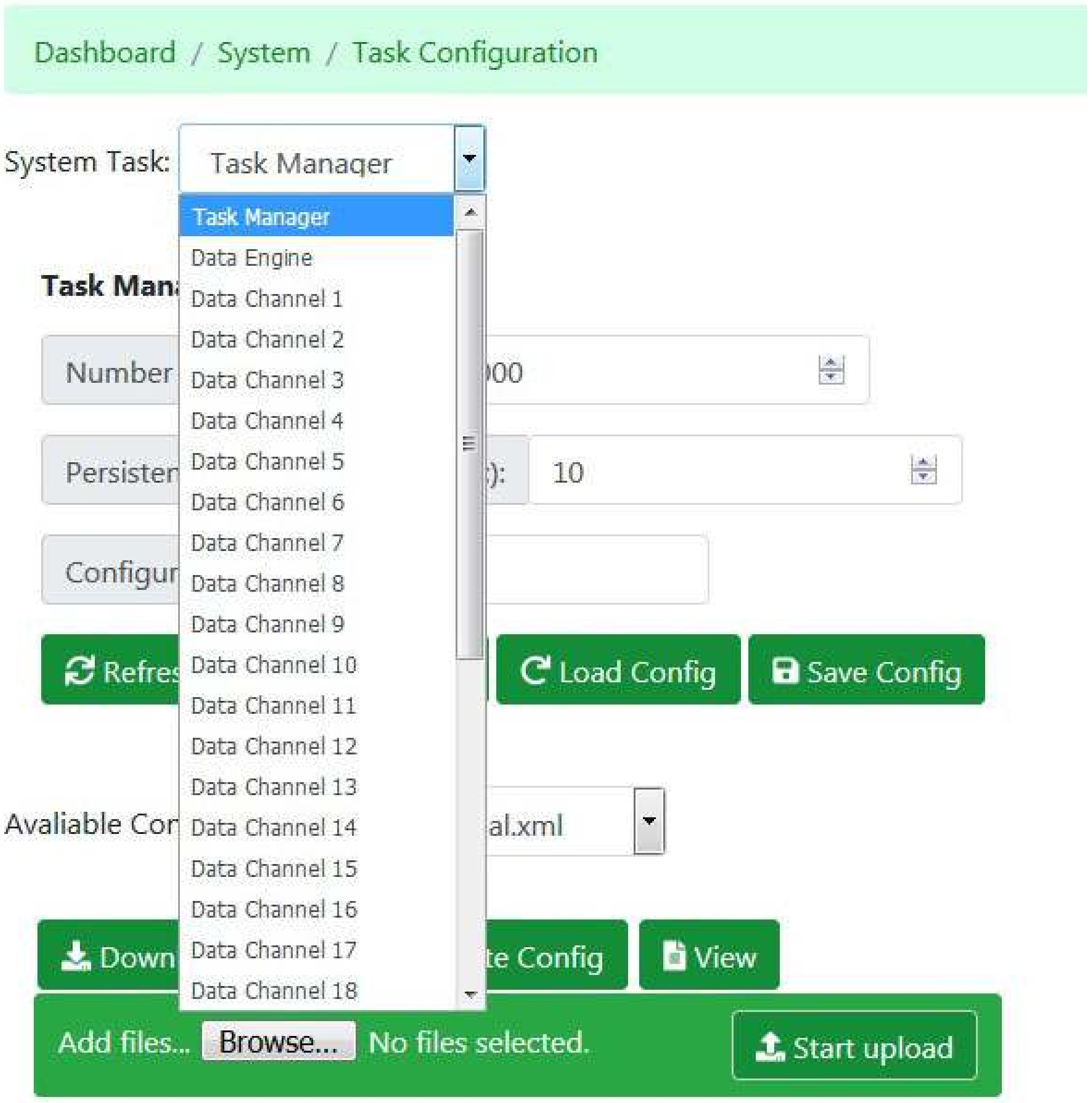
The Task Status page for an IoTServer with no protocols configured yet will have only the Data Engine displayed.


Dashboard / System / Task Status

Task Manager Version: 1.02.04, Port: 42420, Status: Running

Task 0: Data Engine (csiDataEngine): Running
Version: 1.02.01 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Go to the Task Configuration page. The task list will appear as illustrated below. Select the task you wish to configure. Keep in mind channel 0 is always the Data Engine, channel 1 will always be the SNMP Agent if you wish to include one, and channels 2 and 3 are always reserved for the serial ports 2 and 3.


Dashboard / System / Task Configuration

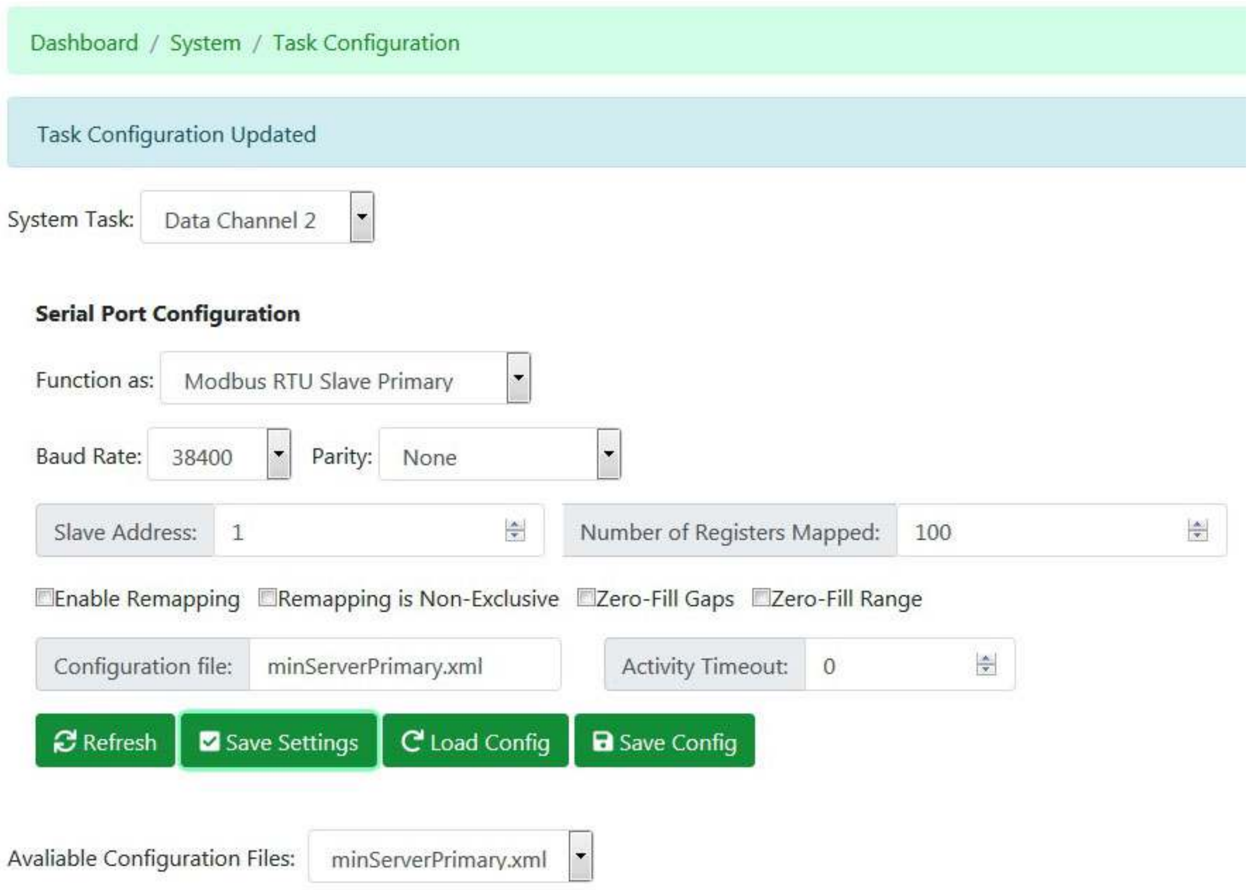
System Task: Task Manager

Task Manager
Data Engine
Data Channel 1
Data Channel 2
Data Channel 3
Data Channel 4
Data Channel 5
Data Channel 6
Data Channel 7
Data Channel 8
Data Channel 9
Data Channel 10
Data Channel 11
Data Channel 12
Data Channel 13
Data Channel 14
Data Channel 15
Data Channel 16
Data Channel 17
Data Channel 18

Let's configure data channel 2, or serial port 2, as an example. Select that channel from the system task list. Next, select its function from the function list. Then enter the applicable parameters like baud rate, etc. The nature of additional parameters will vary according to function selected for the channel.

All tasks must be initially started up with at least a "minimal" XML file. You cannot start a task with absolutely no starting point. If you have no previously saved XML that you wish to load, you can use the minimal file already provided in the device, and simply edit the minimal configuration to make changes as desired at a later time.

After you have entered all of the necessary settings, and selected a configuration file, click Save Settings.



Dashboard / System / Task Configuration

Task Configuration Updated

System Task: Data Channel 2

Serial Port Configuration

Function as: Modbus RTU Slave Primary

Baud Rate: 38400 Parity: None

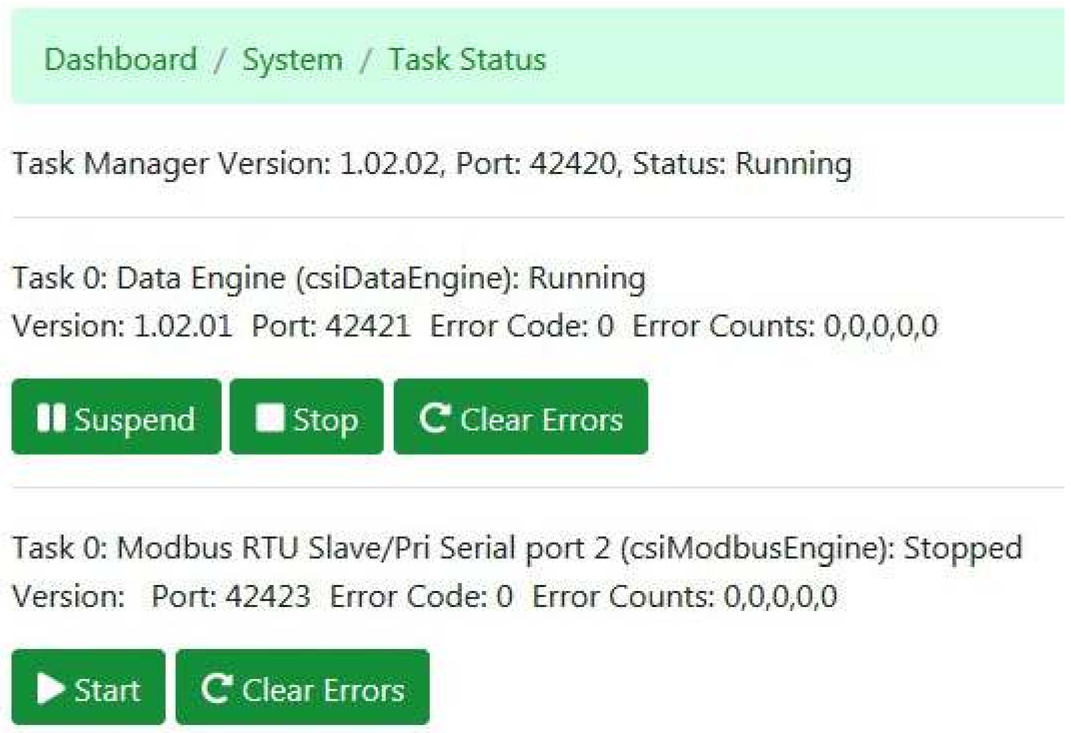
Slave Address: 1 Number of Registers Mapped: 100

☐Enable Remapping ☐Remapping is Non-Exclusive ☐Zero-Fill Gaps ☐Zero-Fill Range

Configuration file: minServerPrimary.xml Activity Timeout: 0

Avaliable Configuration Files: minServerPrimary.xml

Your task is now ready to run. Go to the Task Status page. You will find the task listed with status "Stopped". Click the start button. Upon startup, new databases for this task are created, the databases are loaded for the first time using the content of the XML configuration file you provided, and the task will proceed on to normal operation.



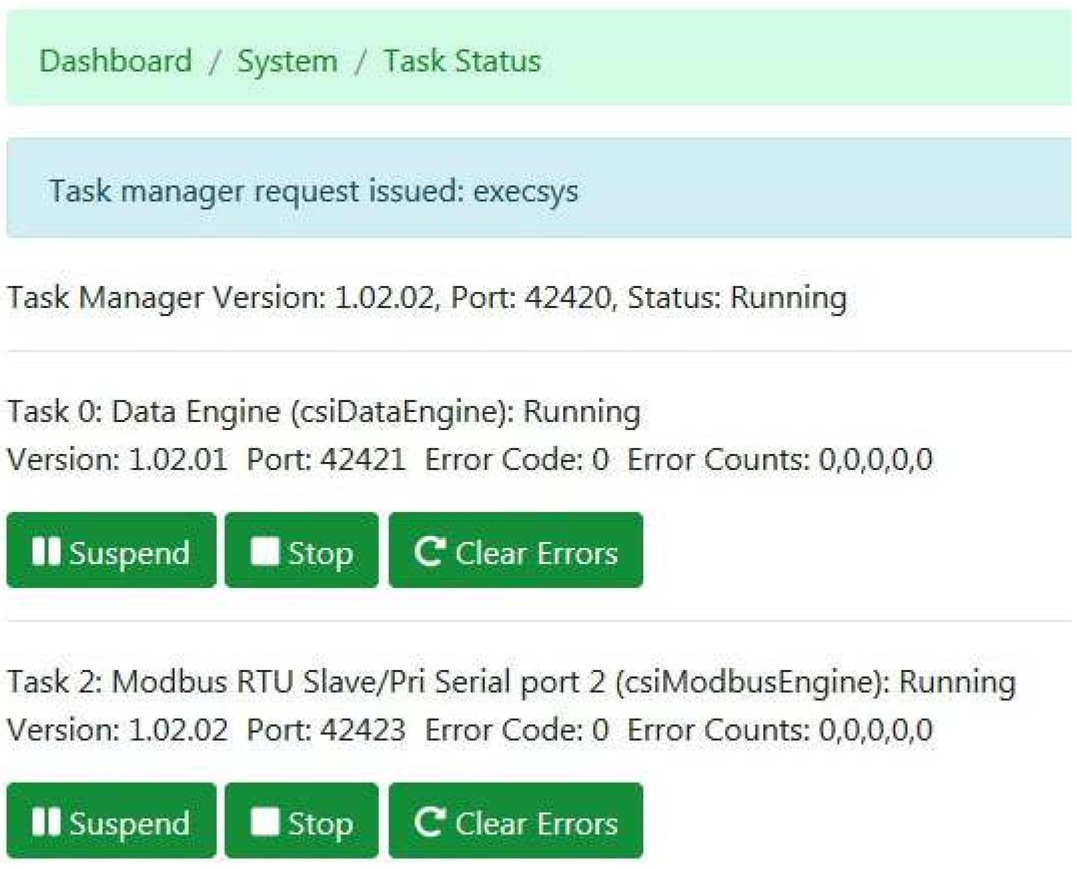
Task Manager Version: 1.02.02, Port: 42420, Status: Running

Task 0: Data Engine (csiDataEngine): Running
Version: 1.02.01 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Task 0: Modbus RTU Slave/Pri Serial port 2 (csiModbusEngine): Stopped
Version: Port: 42423 Error Code: 0 Error Counts: 0,0,0,0,0

After a pause while the task starts, the Task Status page will show "Running" and the buttons will change accordingly.

You will need to refresh your page to get an updated list, but once a new protocol is added to the configuration, it will automatically appear in the Protocol Status and Protocol Configuration menus on the left. To further configure the task you just created, go to its configuration menu under Protocol Configuration.


Dashboard / System / Task Status

Task manager request issued: execsys

Task Manager Version: 1.02.02, Port: 42420, Status: Running

Task 0: Data Engine (csiDataEngine): Running
Version: 1.02.01 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Task 2: Modbus RTU Slave/Pri Serial port 2 (csiModbusEngine): Running
Version: 1.02.02 Port: 42423 Error Code: 0 Error Counts: 0,0,0,0,0

Minimum configuration files that are provided already preloaded are as follows:

File Name	Function or purpose
global.xml	Minimal task manager configuration with only a Data Engine enabled.
minObjects.xml	Minimal set of 3 data objects.
minServerPrimary.xml	Minimal configuration for primary Modbus RTU slave or Modbus TCP server. Should be used only once.
minServerSecondary.xml	Minimal configuration for secondary Modbus RTU slave or Modbus TCP server. Use this for additional instances of slave/server after primary has been defined once.
minRtuMaster.xml	Minimal configuration for Modbus RTU Master.
minTcpClient.xml	Minimal configuration for Modbus TCP Client (master).
minSnmpAgent.xml	Minimal configuration for an SNMP Agent based on the minimal set of data objects in minObjects.xml.
minSnmpClient.xml	Minimal configuration for an SNMP Client.
minSnmpTrapRecv.xml	Minimal configuration for an SNMP Trap Receiver.

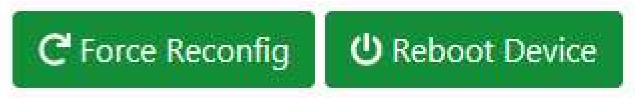
Reconfiguring the System

Forced Reconfigure - Hidden Reset Button

You have the option of unconfiguring the IoTServer using the brute force method available with the reset button. This is talked about in the Reset Button Handling section of Hardware Details.

If you use the reset button to force a reconfigure, the global.xml file will be overwritten with one that only provides for a Data Engine. The configuration file created for the data engine will be defaultobjects.xml and it will contain only one data object. This will be a very minimal configuration, even more minimal than the "minimal" configuration originally shipped.

Forced Reconfigure - "Force Reconfig" Button on Web Page

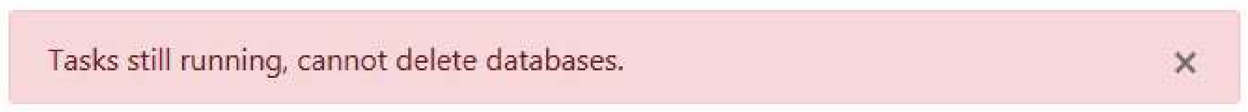


You will find these two buttons at the bottom of the Task Configuration page. Force Reconfig means wipe out the databases, and reload everything from the XML files that are displayed for the various tasks within the task configuration page and task selections. After clearing out all of the existing databases, then click Reboot Device to force a reboot, at which time it will reload the databases from the XML files provided.

IMPORTANT: The Force Reconfig button clears the configuration database. But upon reboot, the task list will be rebuilt from global.xml, and all tasks defined in global.xml will be reconfigured from their respectively assigned XML files. If you want to truly clear out all configuration, follow the Managed Deconfigure process below.

IMPORTANT: Any time Force Reconfig is used, you MUST immediately reboot the device. Clicking Save Settings has only temporary results as no task manager database is recreated without a reboot.

NOTE: The Force Reconfig button cannot be used until all tasks are stopped. You can use the Stop Tasks button to do that. If you try to use the Force Reconfig button with tasks running, you will get the following error message:



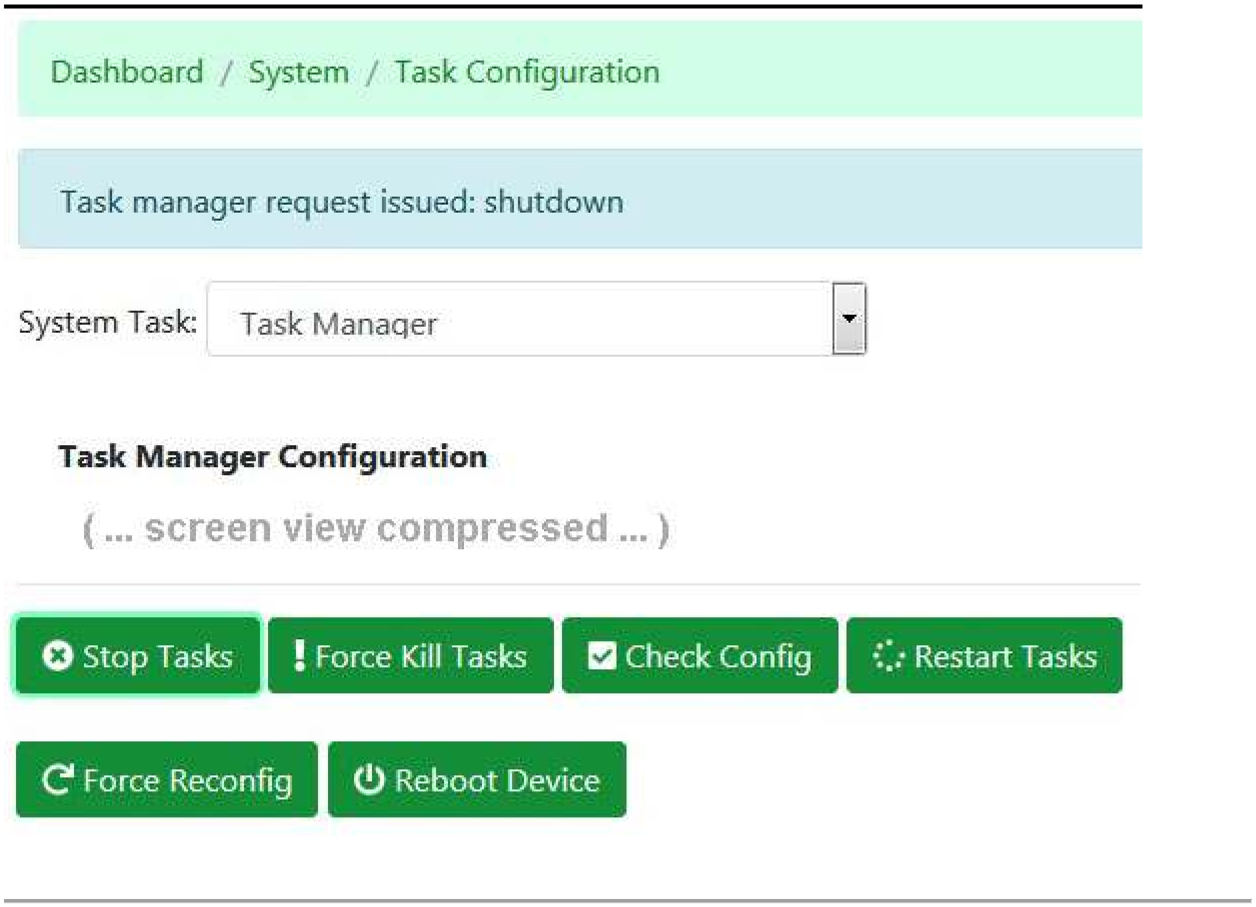
Tasks still running, cannot delete databases.

Master Configuration File Load

You can reconfigure the system by uploading a master file using the Master Load button found on the Task Configuration page, and then rebooting the system. This is only effective if you have previously saved a working configuration (as a master file) that you want to restore. There is more about the Master Save and Master Load buttons on the Task Configuration page.

Managed Deconfigure

The more graceful approach to completely unconfiguring the IoTServer so you can start over is outlined here. Start by clicking Stop Tasks to initiate a shutdown.


Dashboard / System / Task Configuration

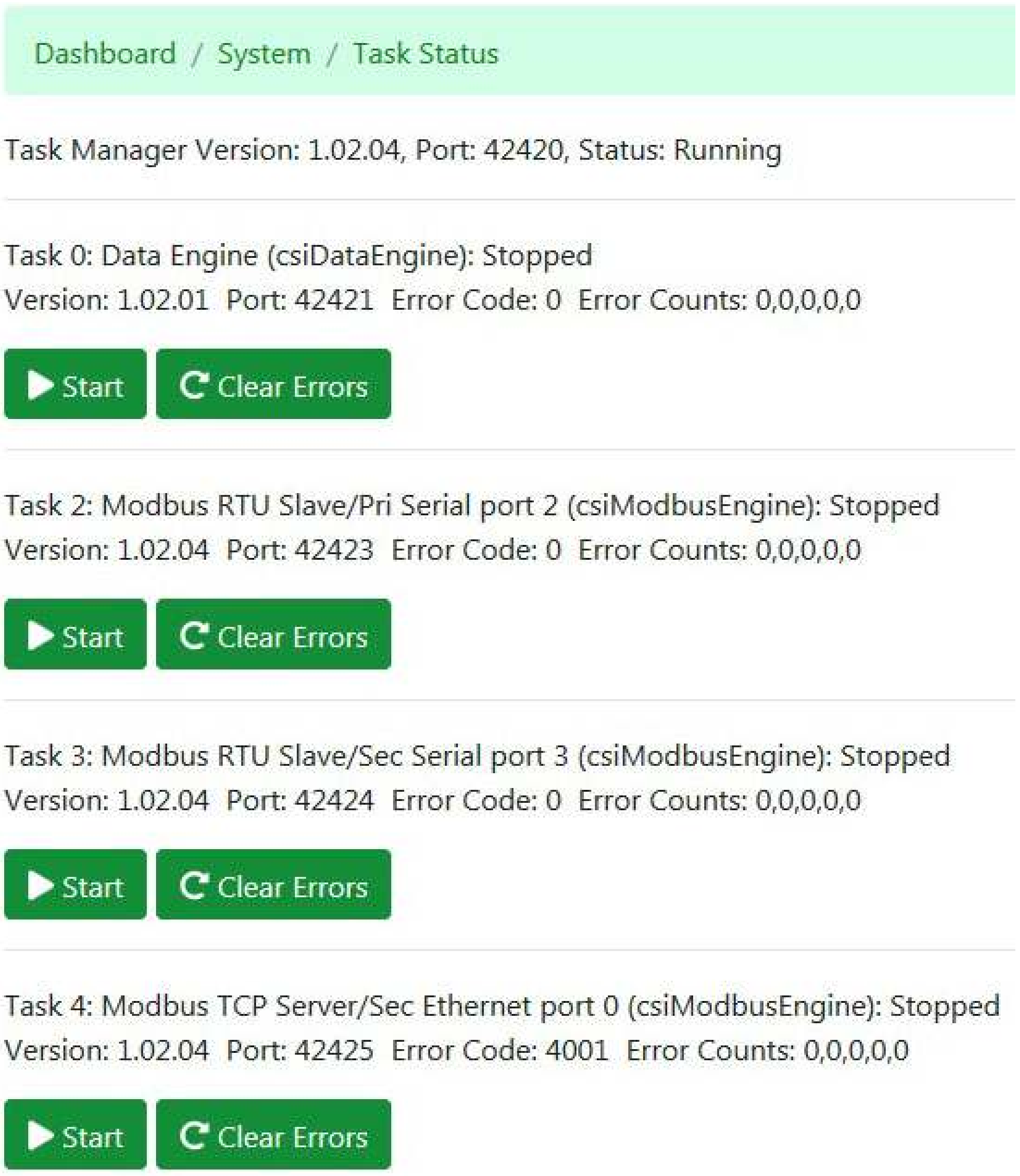
Task manager request issued: shutdown

System Task: Task Manager

Task Manager Configuration

(... screen view compressed ...)

Once all tasks are shut down, you will be able to confirm this by seeing "Stopped" listed as status for all tasks known to the task manager.


Dashboard / System / Task Status

Task Manager Version: 1.02.04, Port: 42420, Status: Running

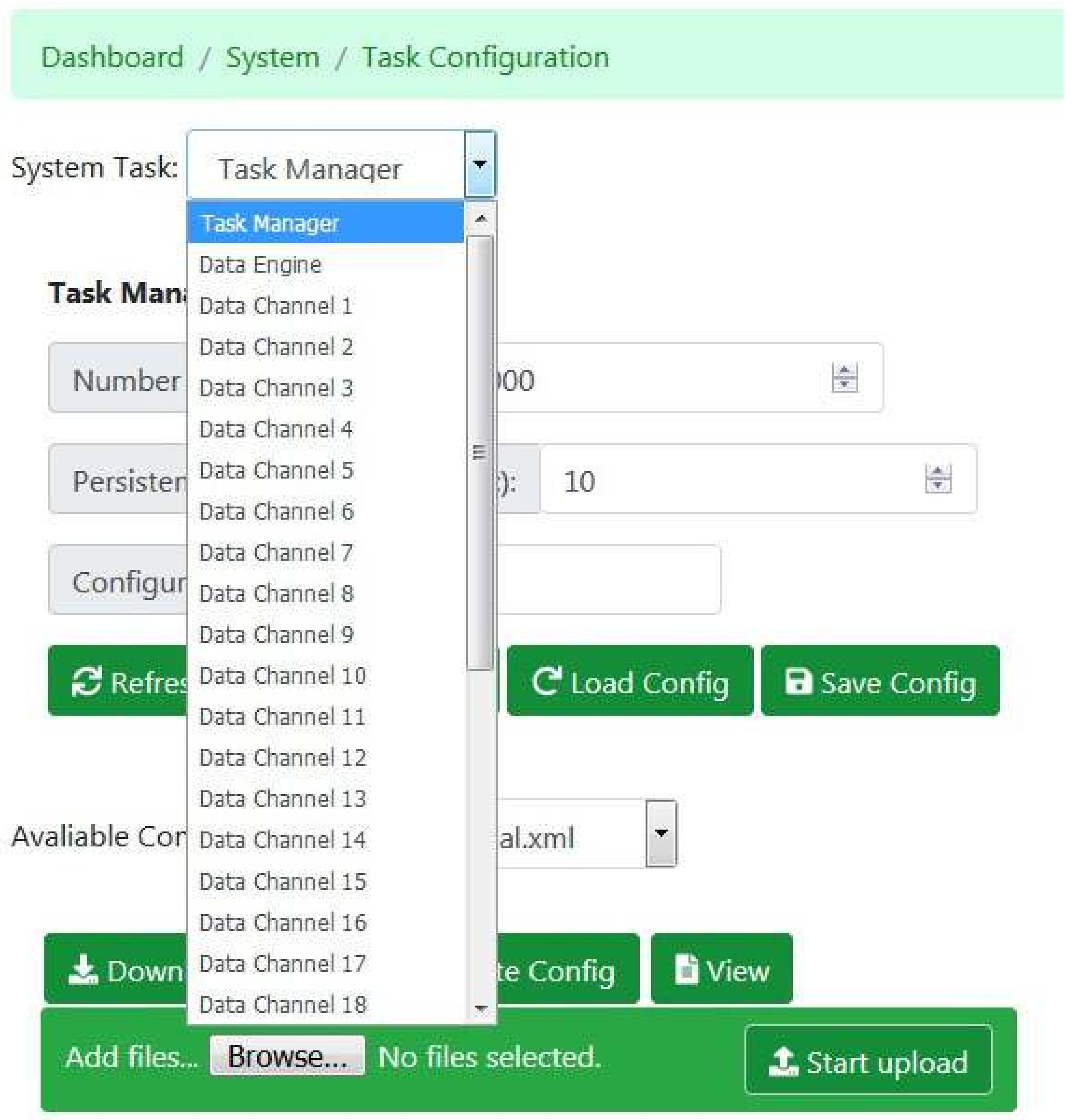
Task 0: Data Engine (csiDataEngine): Stopped
Version: 1.02.01 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Task 2: Modbus RTU Slave/Pri Serial port 2 (csiModbusEngine): Stopped
Version: 1.02.04 Port: 42423 Error Code: 0 Error Counts: 0,0,0,0,0

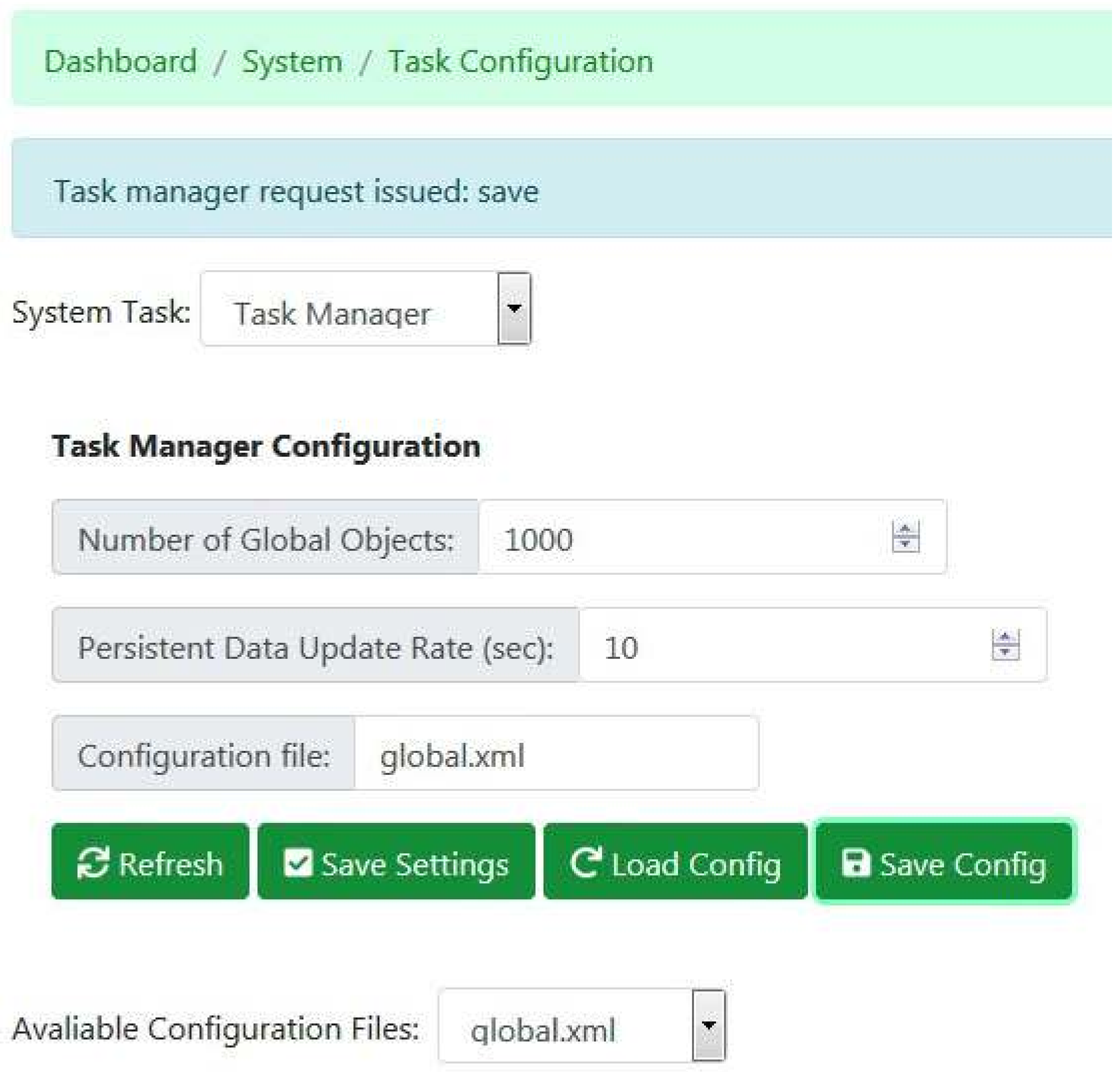
Task 3: Modbus RTU Slave/Sec Serial port 3 (csiModbusEngine): Stopped
Version: 1.02.04 Port: 42424 Error Code: 0 Error Counts: 0,0,0,0,0

Task 4: Modbus TCP Server/Sec Ethernet port 0 (csiModbusEngine): Stopped
Version: 1.02.04 Port: 42425 Error Code: 4001 Error Counts: 0,0,0,0,0

Go through all tasks that are still defined, other than the task manager itself and the data engine, and set their function to "inactive". Refresh the page to update the task list. When all tasks have been unconfigured, the task list will look like this:



Now, with Task Manager selected, and global.xml shown as its configuration file name, click Save Config. Wait for the confirmation that settings have been saved.


Dashboard / System / Task Configuration

Task manager request issued: save

System Task: Task Manager

Task Manager Configuration

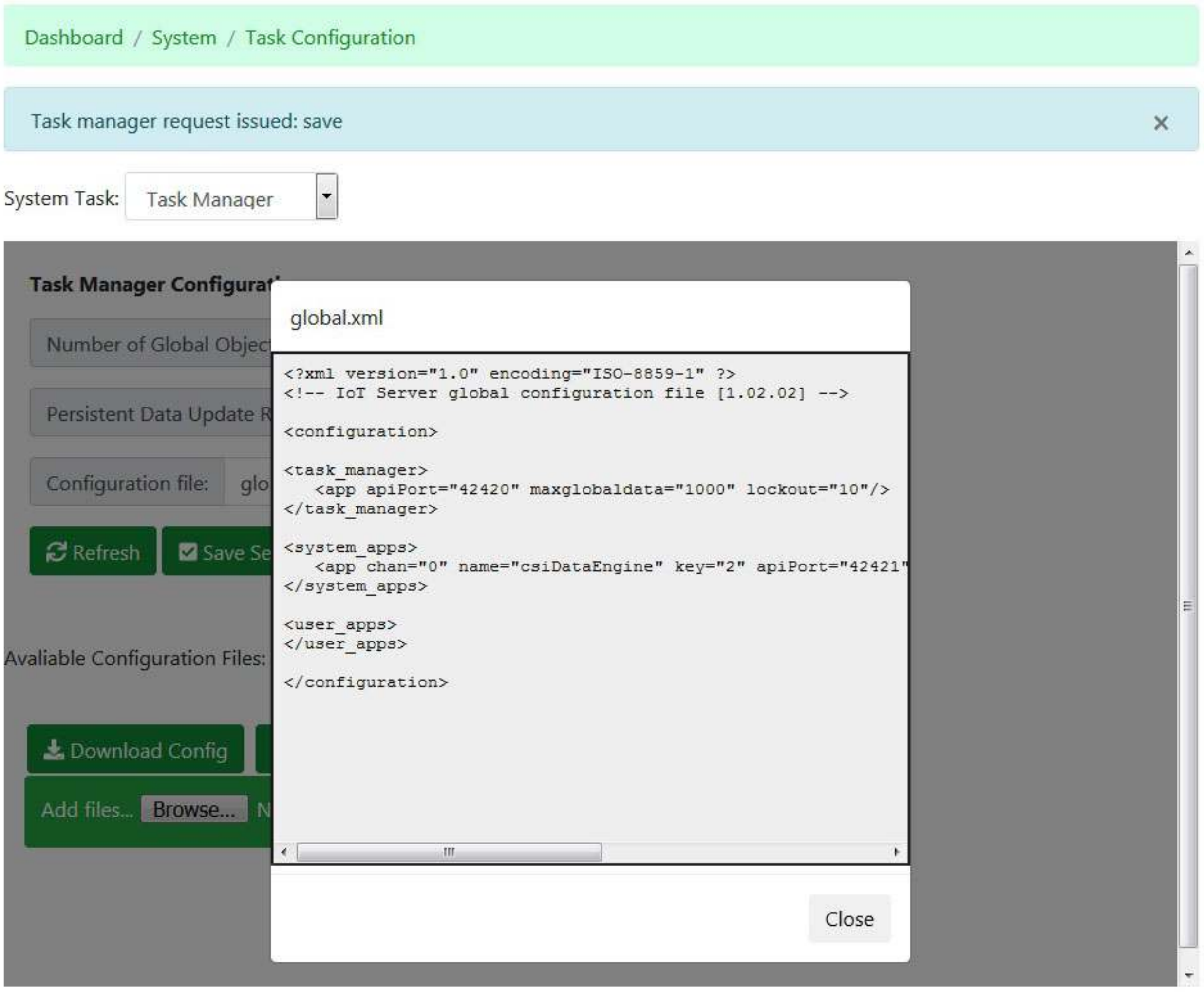
Number of Global Objects: 1000

Persistent Data Update Rate (sec): 10

Configuration file: global.xml

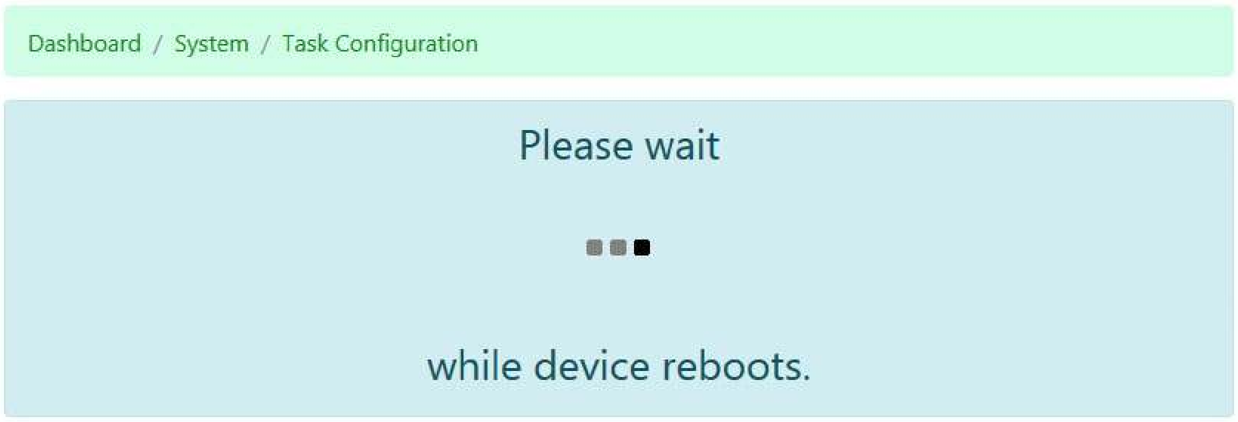
Avaliable Configuration Files: global.xml

Click the View button to view global.xml. Confirm that the only system app listed is the data engine "csiDataEngine".

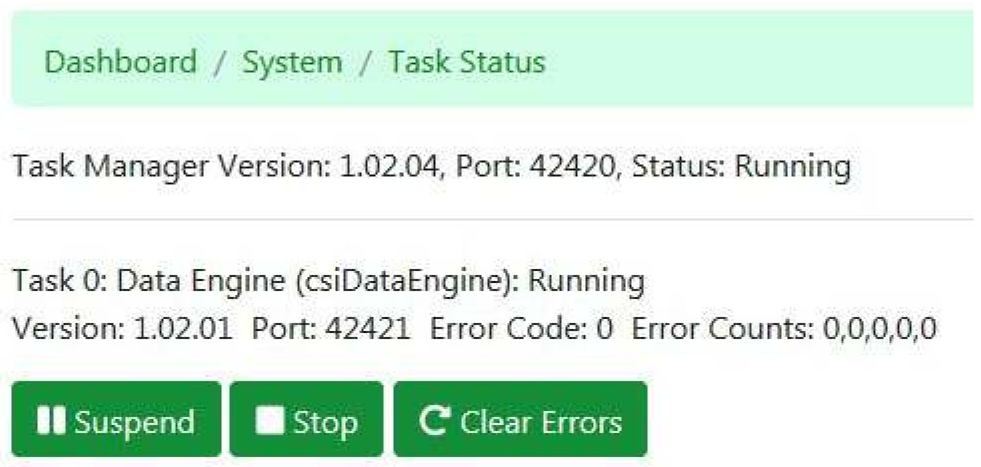


```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- IoT Server global configuration file [1.02.02] -->

<configuration>

<task_manager>
   <app apiPort="42420" maxglobaldata="1000" lockout="10"/>
</task_manager>

<system_apps>
   <app chan="0" name="csiDataEngine" key="2" apiPort="42421"
</system_apps>

<user_apps>
</user_apps>

</configuration>
```

Click the Reboot Device button. You will see this "please wait" screen for a half a minute or so. When reboot is complete, you will be returned to the login page where you must now log in again.



The Task Status page should now show only the data engine.



Dashboard / System / Task Status

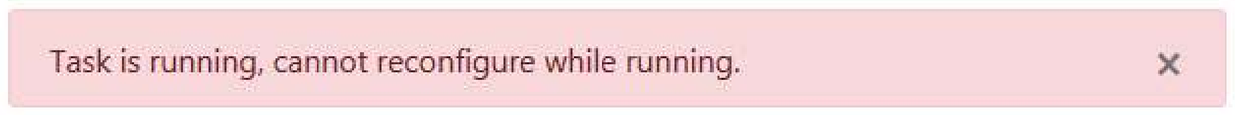
Task Manager Version: 1.02.04, Port: 42420, Status: Running

Task 0: Data Engine (csiDataEngine): Running
Version: 1.02.01 Port: 42421 Error Code: 0 Error Counts: 0,0,0,0,0

Configuring the Tasks

IMPORTANT: If you are going to reconfigure a task that was previously running, STOP the task first, then restart it after reconfiguring. Do not suspend and resume. If just suspended, the resume will continue attempting to run whatever that task's old configuration was.

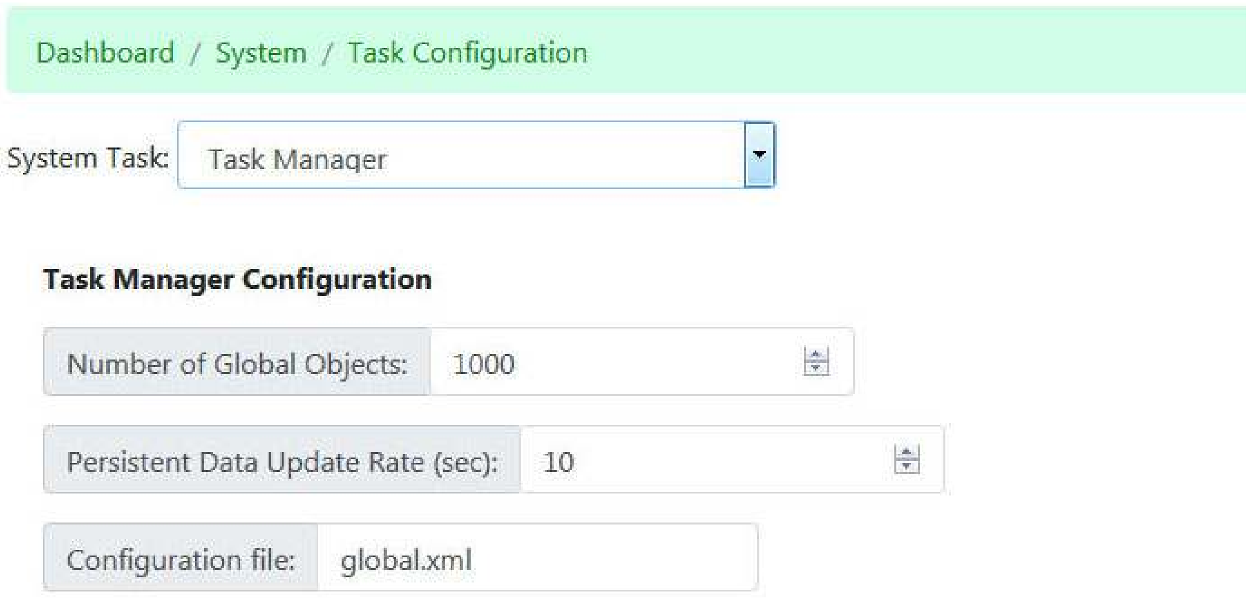
If you attempt to reconfigure a task while it is running, you will see this warning:



Task is running, cannot reconfigure while running.

Each different task has a different set of configuration requirements at the task level. These define things like maximum number of objects that the data engine will serve, maximum number of read and write maps for a protocol client, etc. Configuring the tasks here will also determine which functions show up in the menu on the left the next time you refresh the page. Once a protocol is "turned on" by configuring its task here, that protocol will start to show up under Protocol Status and Protocol Configuration on the left. Click the links below to review configuration for the task types of interest.

Task Manager Task



System Task: Task Manager

Task Manager Configuration

Number of Global Objects: 1000

Persistent Data Update Rate (sec): 10

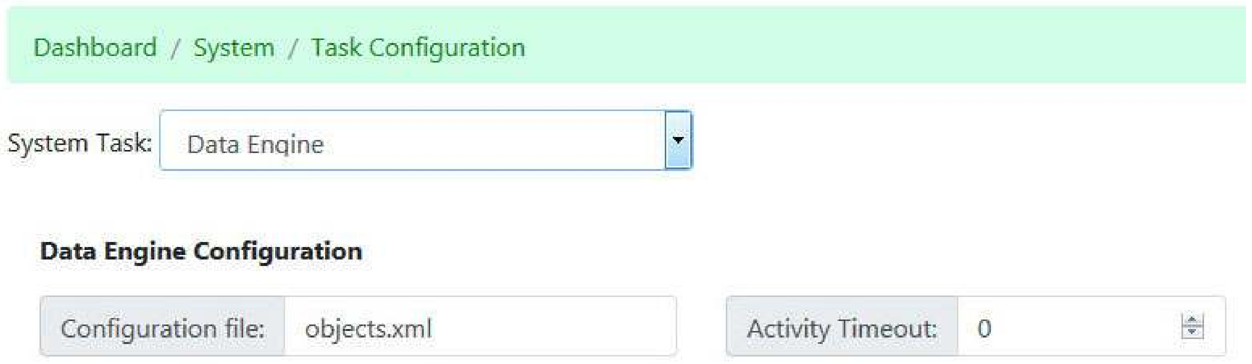
Configuration file: global.xml

The task manager is where you specify the maximum number of data objects that your application will require. You may have up to 10,000 objects. Reducing the number to something closer to what you will actually use will improve efficiency of the system.

The persistent data update rate specifies how frequently persistent data objects are stored to Flash memory. The reason one might want to set this to as long a period as you can tolerate is to extend the life of the Flash memory, which has a finite number of write cycles in its lifetime. The flash technology does include automatic wear leveling, but it is still good practice to not rewrite flash any more often than necessary.

The XML file for the task manager is where all of the task configuration settings are retained when you click the Save XML button. These settings are always retained in the internal database, but for purposes of creating a backup of your overall task configuration, save the XML file and export to your PC.

Data Engine Task


Dashboard / System / Task Configuration
System Task:
Data Engine Configuration

System Task: Data Engine

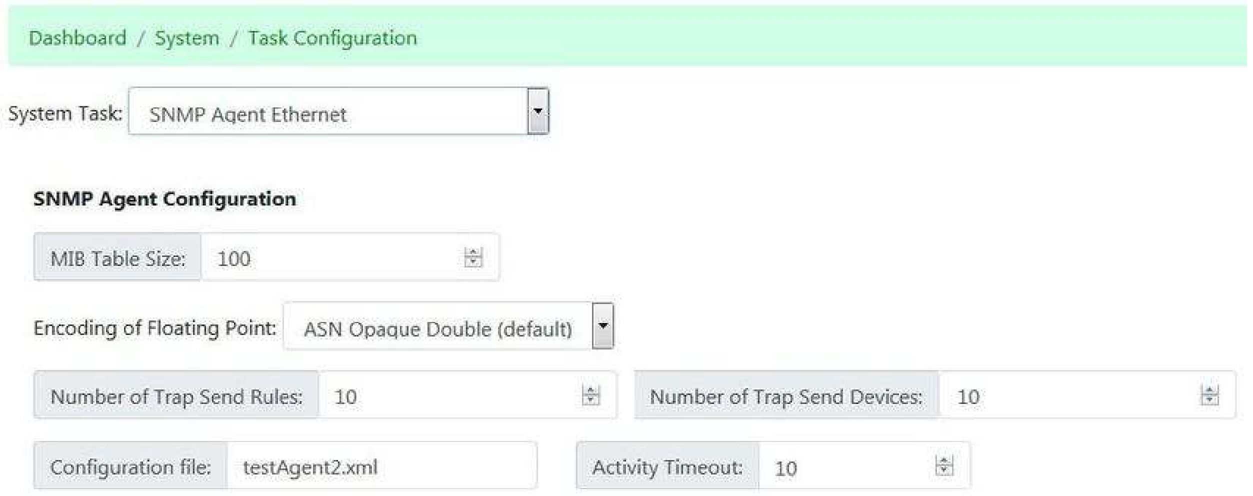
Configuration file: objects.xml

Activity Timeout: 0

There is really not much to configure for the Data Engine, but if you have exported a collection of data objects from another IoTServer and wish to replicate that collection, here is where you can import a file containing the configuration of those objects.

Activity timeout is a form of watchdog. If the task manager does not see the Data Engine update its activity indicator within this amount of time, the task manager will restart the Data Engine. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

SNMP Agent Task



System Task: SNMP Agent Ethernet

MIB Table Size: 100

Encoding of Floating Point: ASN Opaque Double (default)

Number of Trap Send Rules: 10

Number of Trap Send Devices: 10

Configuration file: testAgent2.xml

Activity Timeout: 10

There are four branches in the IoTServer's MIB where you can map data objects. All four tables will be set to the same length that you specify as MIB Table Size here.

If you will be mapping floating point data in your MIB, you must select how you would like it to be encoded. There is no universally recognized representation of floating point in SNMP, but there are de facto standards to choose from. The floating point encodings available are included in the drop-down list. ASN Opaque refers to the Net-SNMP definition for floating point.

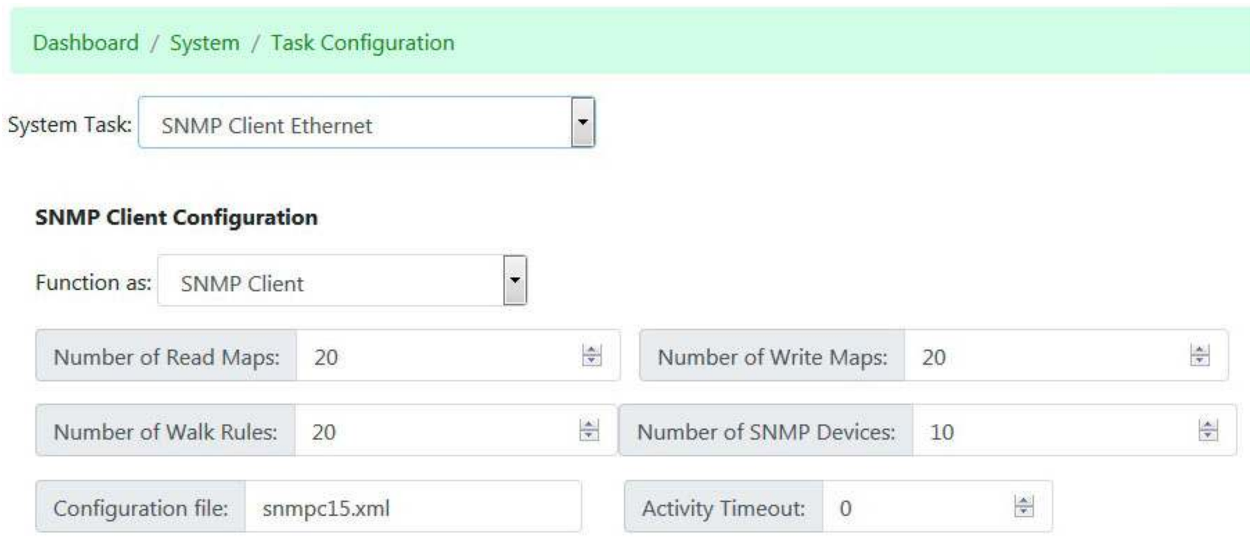
The maximum table sizes for the trap sender are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

The default XML configuration file is entered here. This is the file that the SNMP Agent will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the SNMP Agent.

Activity timeout is a form of watchdog. If the task manager does not see the SNMP Agent task update its activity indicator within this amount of time, the task manager will restart the SNMP Agent. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

IMPORTANT: Do not set the activity timeout for SNMP to less than 120 seconds. While the SNMP functions will update their timers much faster during normal operation, they will time out during startup while waiting for all of the background SNMP tasks and daemons to start. Setting the activity timeout too short for SNMP tasks will result in a system that can never fully start up.

SNMP Client Task



System Task: SNMP Client Ethernet

SNMP Client Configuration

Function as: SNMP Client

Number of Read Maps:	20	Number of Write Maps:	20
Number of Walk Rules:	20	Number of SNMP Devices:	10
Configuration file:	snmpc15.xml	Activity Timeout:	0

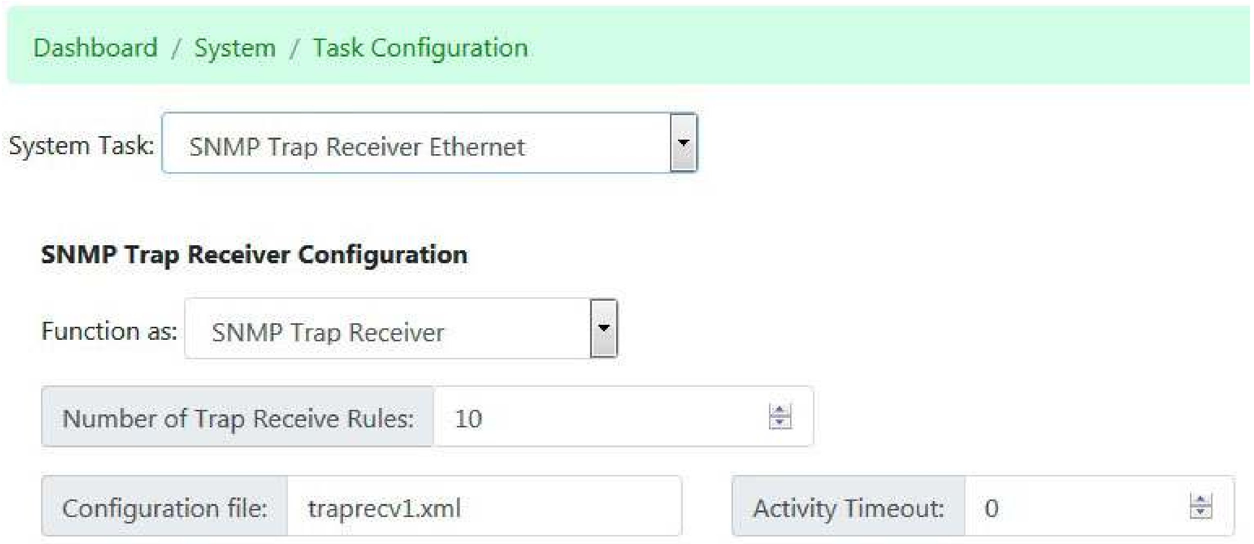
The maximum table sizes for the SNMP Client are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

The default XML configuration file is entered here. This is the file that the SNMP Client will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the SNMP Client.

Activity timeout is a form of watchdog. If the task manager does not see the SNMP Client task update its activity indicator within this amount of time, the task manager will restart the SNMP Client. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

IMPORTANT: Do not set the activity timeout for SNMP to less than 120 seconds. While the SNMP functions will update their timers much faster during normal operation, they will time out during startup while waiting for all of the background SNMP tasks and daemons to start. Setting the activity timeout too short for SNMP tasks will result in a system that can never fully start up.

SNMP Trap Receiver Task


Dashboard / System / Task Configuration

System Task: SNMP Trap Receiver Ethernet

SNMP Trap Receiver Configuration

Function as: SNMP Trap Receiver

Number of Trap Receive Rules: 10

Configuration file: traprecv1.xml

Activity Timeout: 0

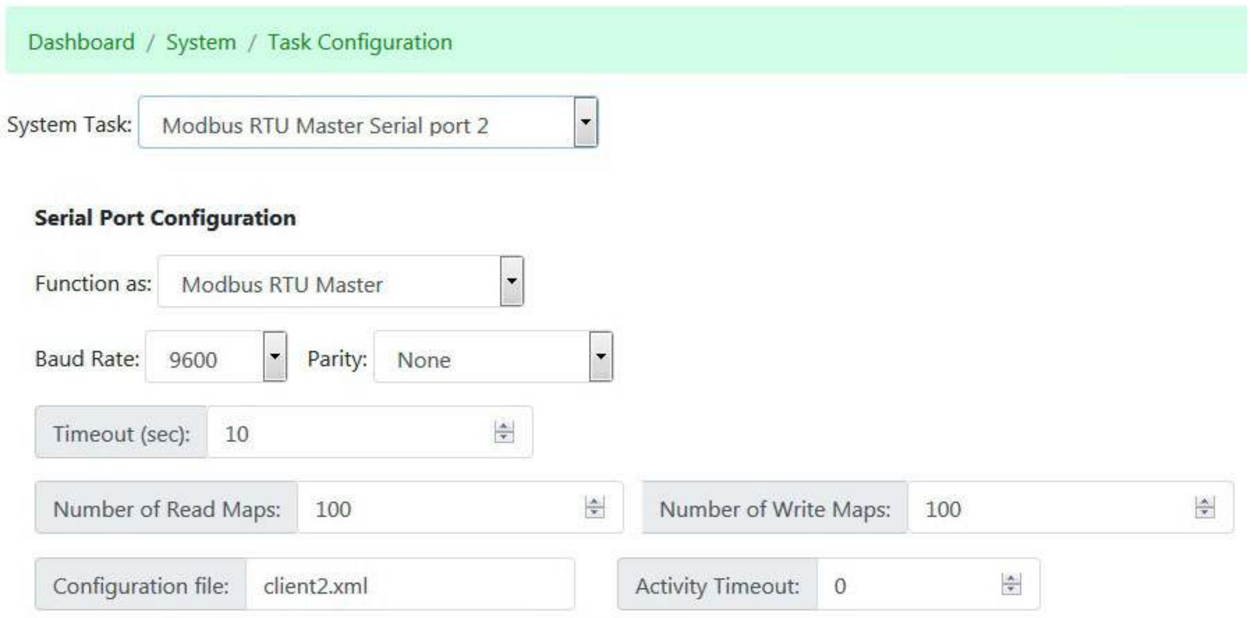
The maximum table sizes for the SNMP Trap Receiver are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

The default XML configuration file is entered here. This is the file that the SNMP Trap Receiver will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the SNMP Trap Receiver.

Activity timeout is a form of watchdog. If the task manager does not see the SNMP Trap Receiver task update its activity indicator within this amount of time, the task manager will restart the SNMP Trap Receiver. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

IMPORTANT: Do not set the activity timeout for SNMP to less than 120 seconds. While the SNMP functions will update their timers much faster during normal operation, they will time out during startup while waiting for all of the background SNMP tasks and daemons to start. Setting the activity timeout too short for SNMP tasks will result in a system that can never fully start up.

Modbus RTU Master Task



System Task: Modbus RTU Master Serial port 2

Serial Port Configuration

Function as: Modbus RTU Master

Baud Rate: 9600 Parity: None

Timeout (sec): 10

Number of Read Maps: 100 Number of Write Maps: 100

Configuration file: client2.xml Activity Timeout: 0

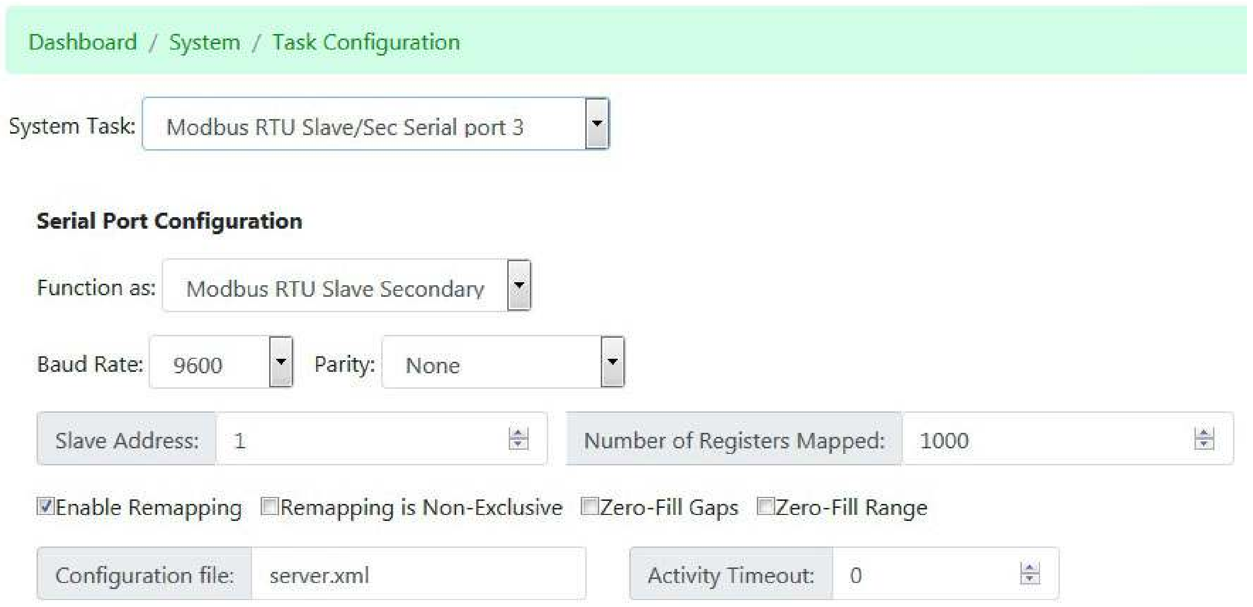
The port settings here are the same settings also accessible on the Port Settings page for Modbus. You can set them from either page.

The maximum table sizes for the Modbus RTU Master are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

The default XML configuration file is entered here. This is the file that the Modbus RTU Master will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the Modbus RTU Master.

Activity timeout is a form of watchdog. If the task manager does not see the Modbus RTU Master task update its activity indicator within this amount of time, the task manager will restart the Modbus RTU Master. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

Modbus RTU Slave Task


Dashboard / System / Task Configuration

System Task: Modbus RTU Slave/Sec Serial port 3

Serial Port Configuration

Function as: Modbus RTU Slave Secondary

Baud Rate: 9600 Parity: None

Slave Address: 1 Number of Registers Mapped: 1000

☑Enable Remapping ☐Remapping is Non-Exclusive ☐Zero-Fill Gaps ☐Zero-Fill Range

Configuration file: server.xml Activity Timeout: 0

Ports or channels defined as a Modbus server (or slave) are either primary or secondary. There can be only one primary, and the primary defines the register mapping that other Modbus clients/masters will see when polling the IoTServer (Babel Buster 4). Secondary servers will see the same set of registers, but have the option of remapping them to different register numbers.

The port settings here are the same settings also accessible on the Port Settings page for Modbus. You can set them from either page.

The maximum table sizes for the Modbus RTU Slave are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

Remapping options are set here for the Modbus Slave/Server. Check the box to enable the desired options.

Remapping: If one device wants to see a data value at address 0 while another device on another port wants to see the same data at address 100, that is what remapping will do.

Remapping, non-exclusive: The default is exclusive, meaning if the requested register address is not found in the remapping table, an “illegal data address” exception is returned. If the server should look first at remapped addresses, and then second at unmapped addresses, and return either-or, then the non-exclusive option should be enabled. The non-exclusive option is a modifier of the remapping option. Setting only the non-exclusive bit will have no effect.

Zero fill: If the register set has gaps in the address range, attempts to access register addresses in the gap would normally return an “illegal data address” exception. If the server should simply zero-fill the non-existent registers in the range requested, then this option should be enabled.

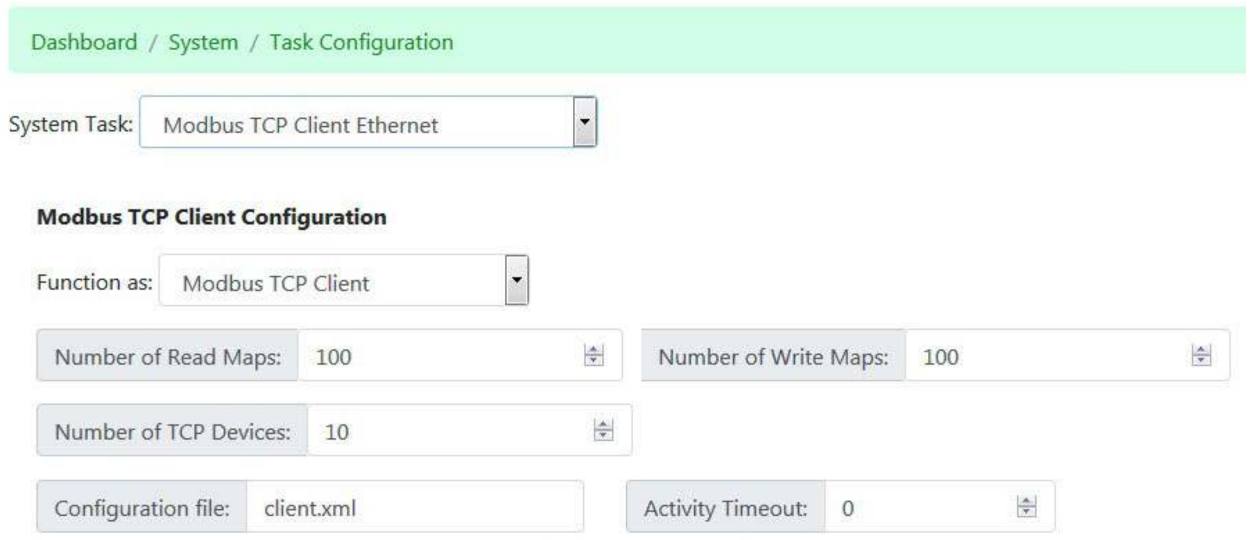
This zero-fill option only applies to gaps within the defined register range. Example: If registers 1 and 3 exist, reading register 2 would return zero while reading register 4 would result in an exception with remapping enabled. Without remapping enabled, zero fill becomes the equivalent of zero fill all.

Zero fill all: This option works the same as zero fill, except the entire address range is zero filled. In the above example, where reading register 4 resulted in an exception, register 4 (or higher) would return zero with the "all" option. The zero fill all bit is a modifier of the zero fill option if remapping is enabled and has no effect if set by itself. Without remapping enabled, the zero fill option becomes the equivalent of zero fill all.

The default XML configuration file is entered here. This is the file that the Modbus RTU Slave will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the Modbus RTU Slave.

Activity timeout is a form of watchdog. If the task manager does not see the Modbus RTU Slave task update its activity indicator within this amount of time, the task manager will restart the Modbus RTU Slave. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

Modbus TCP Client Task



System Task: Modbus TCP Client Ethernet

Modbus TCP Client Configuration

Function as: Modbus TCP Client

Number of Read Maps: 100

Number of Write Maps: 100

Number of TCP Devices: 10

Configuration file: client.xml

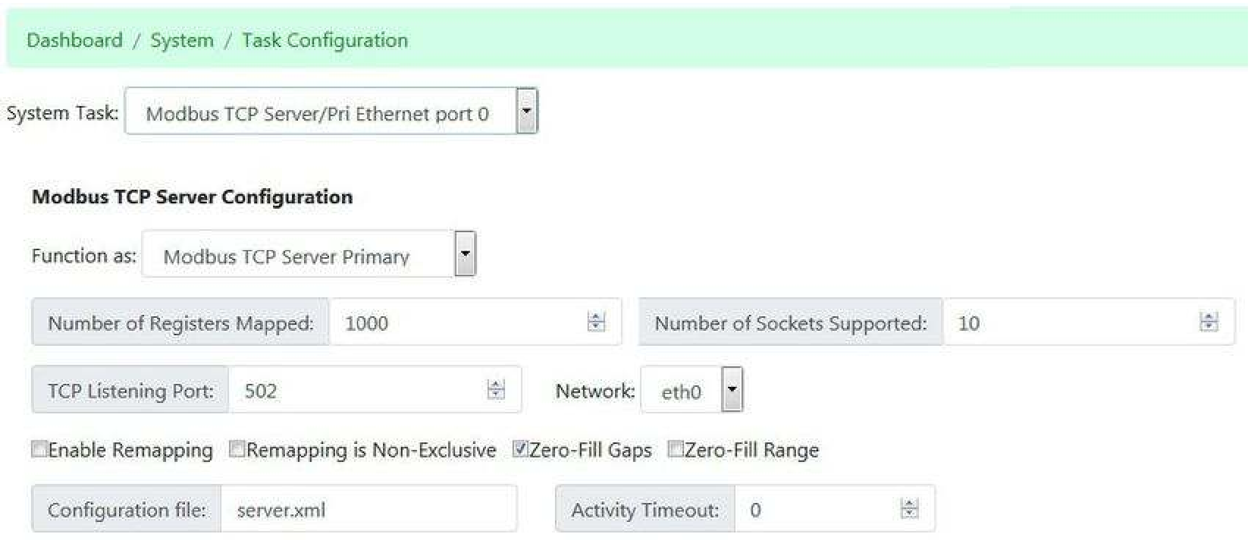
Activity Timeout: 0

The maximum table sizes for the Modbus TCP Client are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

The default XML configuration file is entered here. This is the file that the Modbus TCP Client will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the Modbus TCP Client.

Activity timeout is a form of watchdog. If the task manager does not see the Modbus TCP Client task update its activity indicator within this amount of time, the task manager will restart the Modbus TCP Client. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

Modbus TCP Server Primary Task



System Task: Modbus TCP Server/Pri Ethernet port 0

Modbus TCP Server Configuration

Function as: Modbus TCP Server Primary

Number of Registers Mapped: 1000

Number of Sockets Supported: 10

TCP Listening Port: 502

Network: eth0

☐ Enable Remapping ☐ Remapping is Non-Exclusive ☑ Zero-Fill Gaps ☐ Zero-Fill Range

Configuration file: server.xml

Activity Timeout: 0

Ports or channels defined as a Modbus server are either primary or secondary. There can be only one primary, and the primary defines the register mapping that other Modbus clients/masters will see when polling the IoTServer (Babel Buster 4). Secondary servers will see the same set of registers, but have the option of remapping them to different register numbers.

The maximum table sizes for the Modbus TCP Server are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

Select the port that this Modbus TCP Server should listen on. The default port for Modbus TCP is 502. Select either eth0 or eth1 for physical Ethernet port that this server should listen on.

Remapping options are set here for the Modbus TCP Server. Check the box to enable the desired options.

Remapping: If one device wants to see a data value at address 0 while another device on another port wants to see the same data at address 100, that is what remapping will do.

Remapping, non-exclusive: The default is exclusive, meaning if the requested register address is not found in the remapping table, an "illegal data address" exception is returned. If the server should look first at remapped addresses, and then second at unmapped addresses, and return either-or, then the non-exclusive option should be enabled. The non-exclusive option is a modifier of the remapping option. Setting only the non-exclusive bit will have no effect.

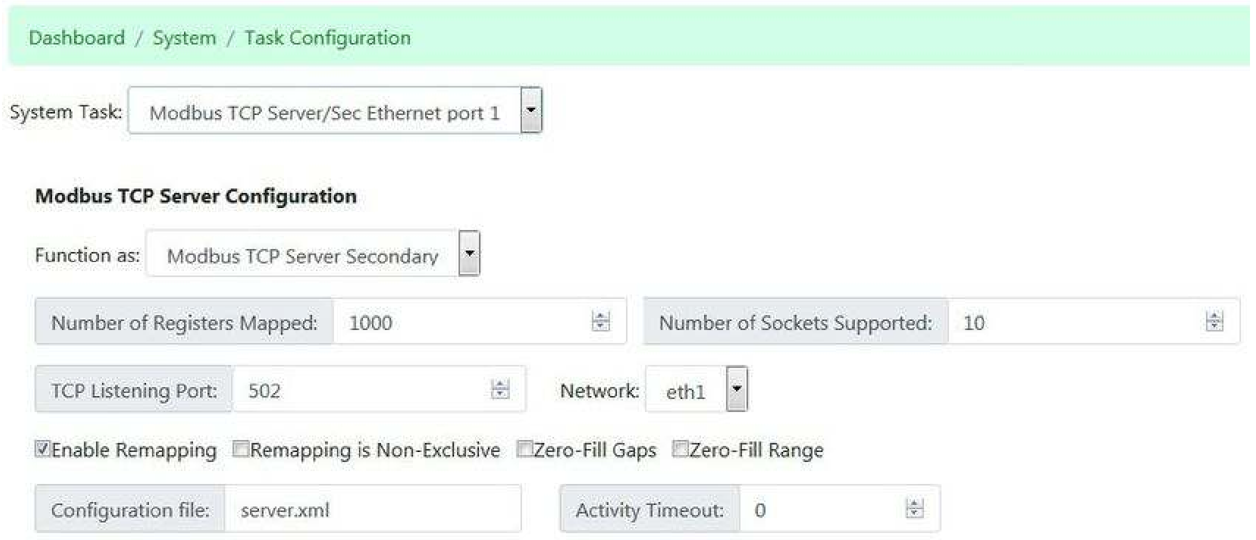
Zero fill: If the register set has gaps in the address range, attempts to access register addresses in the gap would normally return an “illegal data address” exception. If the server should simply zero-fill the non-existent registers in the range requested, then this option should be enabled. This zero-fill option only applies to gaps within the defined register range. Example: If registers 1 and 3 exist, reading register 2 would return zero while reading register 4 would result in an exception with remapping enabled. Without remapping enabled, zero fill becomes the equivalent of zero fill all.

Zero fill all: This option works the same as zero fill, except the entire address range is zero filled. In the above example, where reading register 4 resulted in an exception, register 4 (or higher) would return zero with the “all” option. The zero fill all bit is a modifier of the zero fill option if remapping is enabled and has no effect if set by itself. Without remapping enabled, the zero fill option becomes the equivalent of zero fill all.

The default XML configuration file is entered here. This is the file that the Modbus TCP Server will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the Modbus TCP Server.

Activity timeout is a form of watchdog. If the task manager does not see the Modbus TCP Server task update its activity indicator within this amount of time, the task manager will restart the Modbus TCP Server. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

Modbus TCP Server Secondary Task


Dashboard / System / Task Configuration

System Task: Modbus TCP Server/Sec Ethernet port 1

Modbus TCP Server Configuration

Function as: Modbus TCP Server Secondary

Number of Registers Mapped: 1000
Number of Sockets Supported: 10

TCP Listening Port: 502
Network: eth1

☑ Enable Remapping ☐ Remapping is Non-Exclusive ☐ Zero-Fill Gaps ☐ Zero-Fill Range

Configuration file: server.xml
Activity Timeout: 0

Ports or channels defined as a Modbus server are either primary or secondary. There can be only one primary, and the primary defines the register mapping that other Modbus clients/masters will see when polling the IoTServer (Babel Buster 4). Secondary servers will see the same set of registers, but have the option of remapping them to different register numbers.

The maximum table sizes for the Modbus TCP Server are defined here. Keeping the tables set to a range that you will actually use will improve efficiency.

Select the port that this Modbus TCP Server should listen on. The default port for Modbus TCP is 502. Select either eth0 or eth1 for physical Ethernet port that this server should listen on.

Remapping options are set here for the Modbus TCP Server. Check the box to enable the desired options.

Remapping: If one device wants to see a data value at address 0 while another device on another port wants to see the same data at address 100, that is what remapping will do.

Remapping, non-exclusive: The default is exclusive, meaning if the requested register address is not found in the remapping table, an “illegal data address” exception is returned. If the server should look first at remapped addresses, and then second at unmapped addresses, and return either-or, then the non-exclusive option should be enabled. The non-exclusive option is a modifier of the remapping option. Setting only the non-exclusive bit will have no effect.

Zero fill: If the register set has gaps in the address range, attempts to access register addresses in the gap would normally return an “illegal data address” exception. If the server should simply zero-fill the non-existent registers in the range requested, then this option should be enabled. This zero-fill option only applies to gaps within the defined register range. Example: If registers 1 and 3 exist, reading register 2 would return zero while reading register 4 would result in an exception with remapping enabled. Without remapping enabled, zero fill becomes the equivalent of zero fill all.

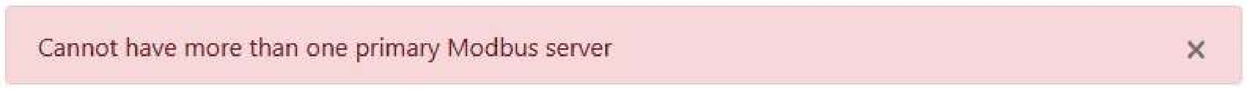
Zero fill all: This option works the same as zero fill, except the entire address range is zero filled. In the above example, where reading register 4 resulted in an exception, register 4 (or higher) would return zero with the “all” option. The zero fill all bit is a modifier of the zero fill option if remapping is enabled and has no effect if set by itself. Without remapping enabled, the zero fill option becomes the equivalent of zero fill all.

The default XML configuration file is entered here. This is the file that the Modbus TCP Server will attempt to automatically load upon first time startup. After that, settings are retained in the database, and further file activity can be managed on the Config File page for the Modbus TCP Server.

Activity timeout is a form of watchdog. If the task manager does not see the Modbus TCP Server task update its activity indicator within this amount of time, the task manager will restart the Modbus TCP Server. If set to zero, the watchdog is disabled. The task manager itself is restarted by the Linux cron if it stops responding.

Check Configuration

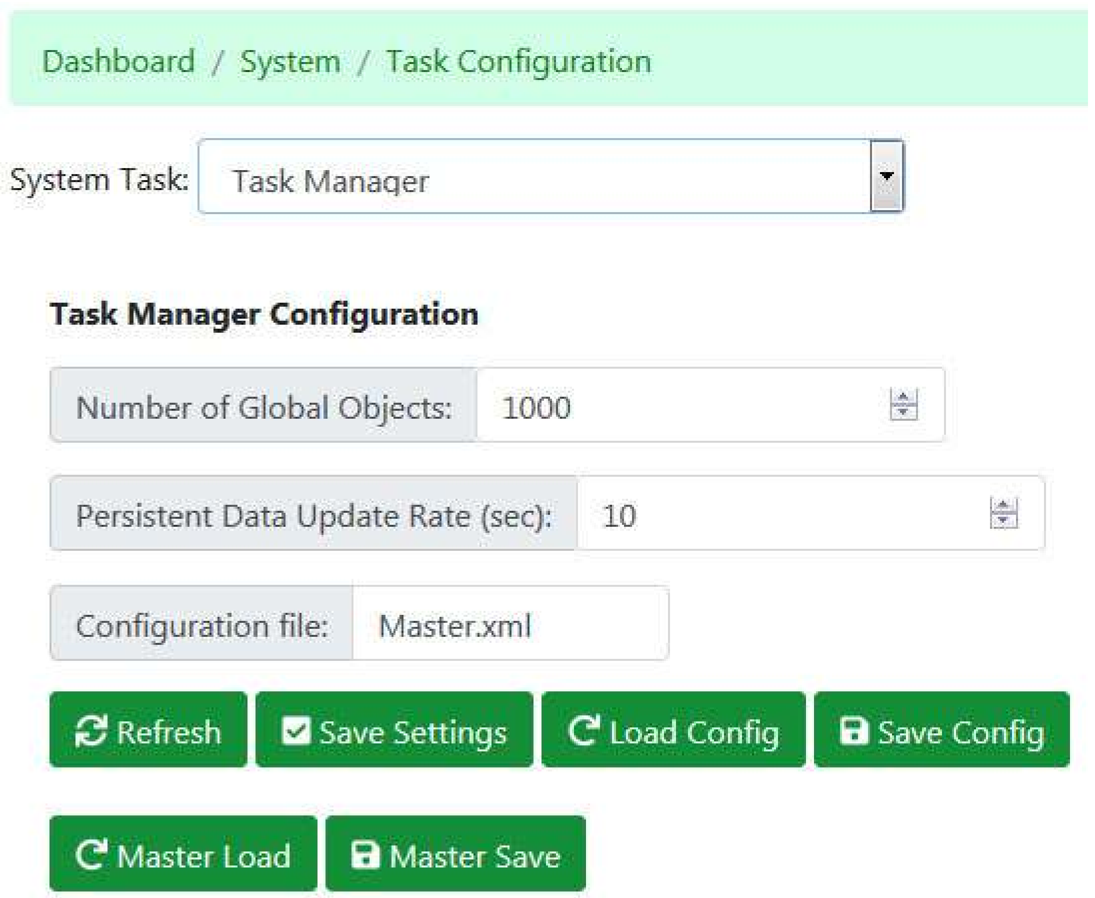
There is a Check Config button toward the bottom of the Task Configuration page. After you have configured your various tasks, click this button to check for conflicts. If there are conflicts in the configuration, you will see an error message displayed at the top of the Task Configuration page. An example looks like this:


Cannot have more than one primary Modbus server ×

There can be only one primary Modbus server (slave). All additional servers or slaves must be defined as secondary. If you try to configure multiple primary servers, this is the message you will get when you click Check Config.

Master Configuration Files

There are two special load/save buttons that only show up when the Task Manager is selected as System Task. The task manager has a “master save” and “master load” feature.



System Task: Task Manager

Task Manager Configuration

Number of Global Objects: 1000

Persistent Data Update Rate (sec): 10

Configuration file: Master.xml

The master save will first go down the list of all tasks currently configured to run, make sure they are running, then issue the "save config" request to each task and wait for its completion. This is the equivalent of the system automatically going through all of the Config File pages and clicking Save for your. Then it creates a master file by concatenating all of the individual files with a file tag to separate them. It does not concatenate all of the files in the config folder, just those files freshly saved representing configuration currently in use, one per task. It will also include snmpdGenerated.conf and snmptrapdGenerated.conf if they exist.

You must enter a name for the master file in the Configuration File window, and it must be a new name (file must not currently exist on the system). If you wish to replace an existing file, delete the old file first, then reuse its name as a new filename.

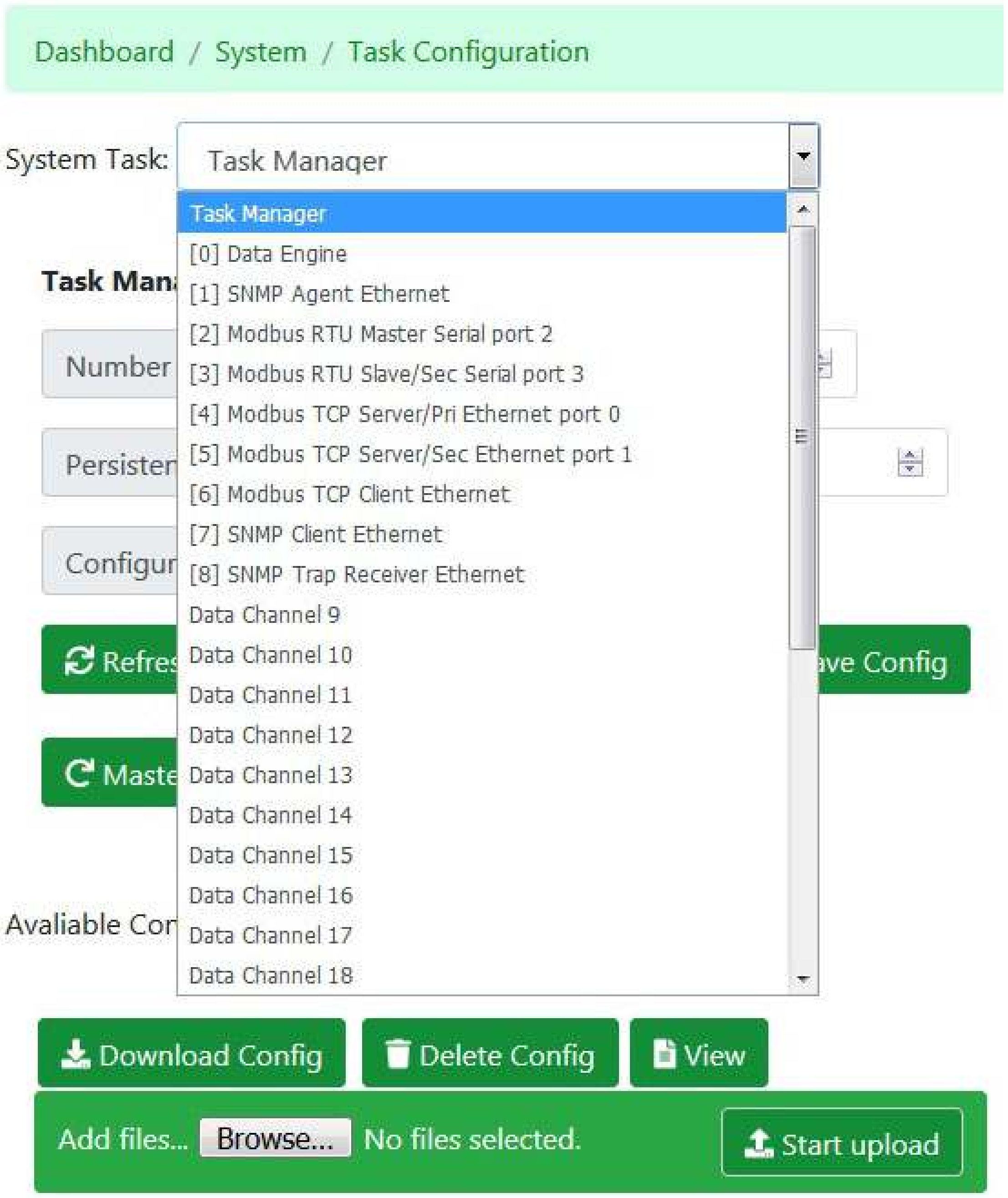
The master load reverses the process, reading the master file and creating multiple files for the various tasks. Upload the file if it does not already exist on the system. Select the file to process from the drop-down list of available files. The master load does not attempt to reconfigure tasks, it only converts one big file into multiple smaller files. Once master load is complete, there are two ways of proceeding:

(1) Click Force Reconfig, then Reboot Device.

OR (2) Go through each task and selectively load each configuration file one at a time.

Task List and Data Channels

Data channels 0, 1, 2, and 3 have dedicated purposes. After that, any network function can be assigned to any data channel 4 and up.


Dashboard / System / Task Configuration

System Task: Task Manager

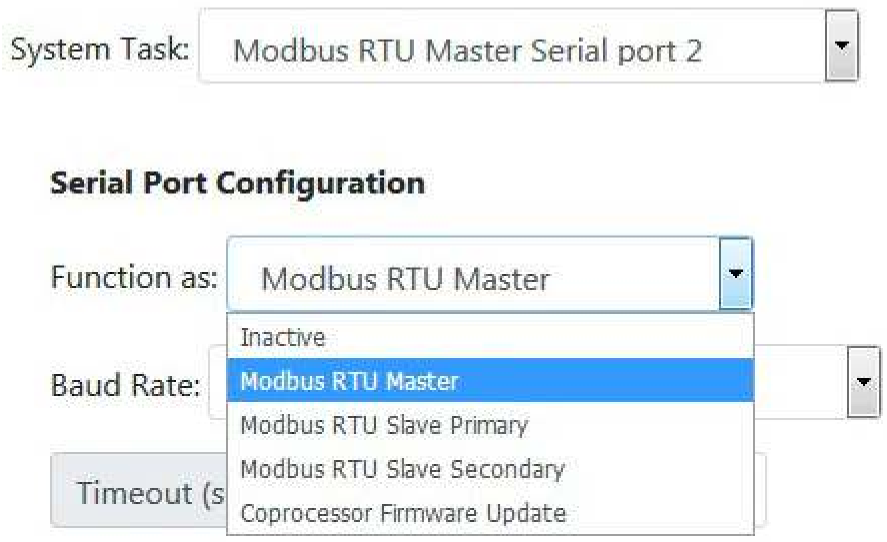
Task Manager
[0] Data Engine
[1] SNMP Agent Ethernet
[2] Modbus RTU Master Serial port 2
[3] Modbus RTU Slave/Sec Serial port 3
[4] Modbus TCP Server/Pri Ethernet port 0
[5] Modbus TCP Server/Sec Ethernet port 1
[6] Modbus TCP Client Ethernet
[7] SNMP Client Ethernet
[8] SNMP Trap Receiver Ethernet
Data Channel 9
Data Channel 10
Data Channel 11
Data Channel 12
Data Channel 13
Data Channel 14
Data Channel 15
Data Channel 16
Data Channel 17
Data Channel 18

An example of a set of configured tasks appears above. There are some restrictions on what can be configured where. The Data Engine (data object manager) will always be task 0. The SNMP Agent will always be task 1. The serial protocol for serial port 2 will always be task 2. The serial protocol for serial port 3 will always be task 3. After that, any number of protocol engines or other applications may be assigned as task 4 and up.

The task number and data channel number are always the same number. The "channel number" has significance in that a set of internal flags is maintained that allows each task to inform other tasks when new data is available in any given global data object.

Port Options

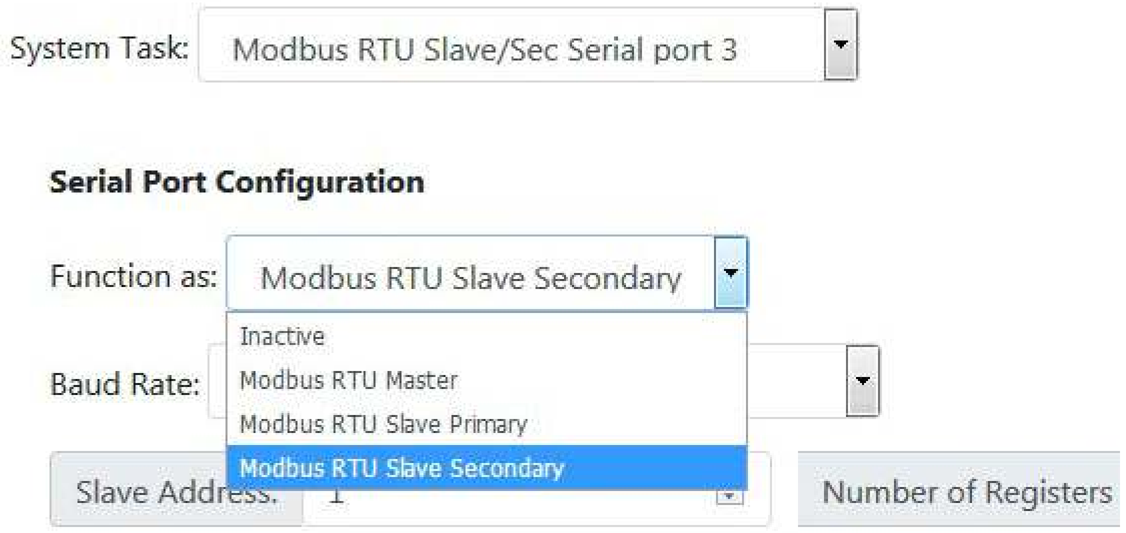
Serial Ports


System Task: Modbus RTU Master Serial port 2

Serial Port Configuration

Function as: Modbus RTU Master

The options currently available for serial port 2, assigned to task 2, is illustrated above.


System Task: Modbus RTU Slave/Sec Serial port 3

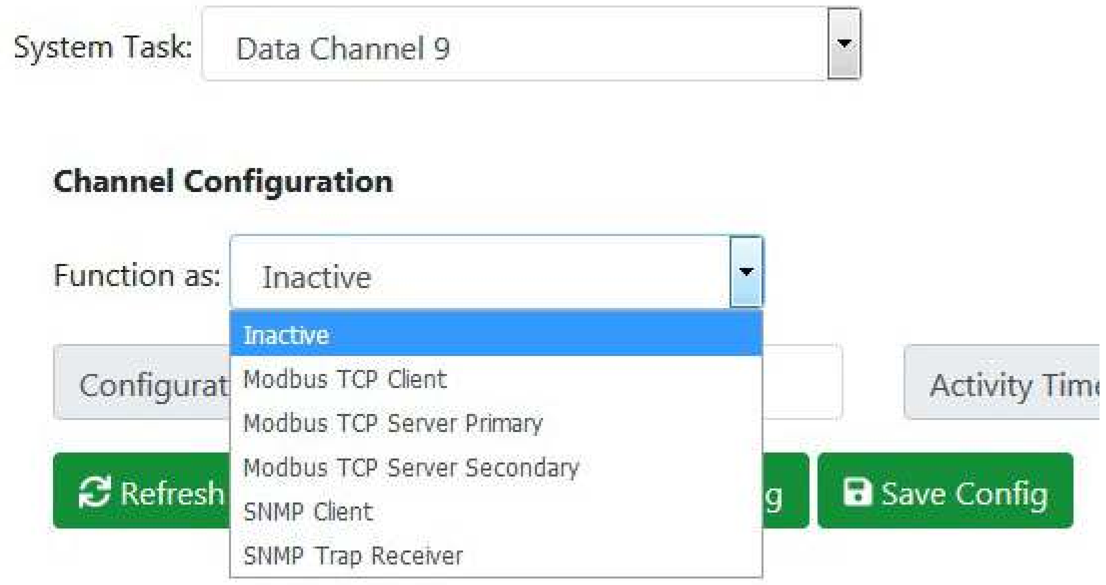
Serial Port Configuration

Function as: Modbus RTU Slave Secondary

Slave Address: 1

The options currently available for serial port 3, assigned to task 3, is illustrated above.

Ethernet Ports



The options currently available for Ethernet applications as tasks 4 and up are illustrated above.

Task Manager XML Files

The default file name for the top level configuration file for the task manager is "global.xml". A minimum global.xml file looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<configuration>

<task_manager>
<app apiPort="42420" maxglobaldata="1000" lockout="15" configFile=”global.xml”/>
</task_manager>

<system_apps>
<app chan="0" name="csiDataEngine" key="2" apiPort="42421" configFile="objects.xml"/>
<app chan="1" name="csiModbusEngine" key="3550" apiPort="42422" portType="1"
portNum="0" cval="1000,502,20,0,0" configFile="server.xml"/>
<app chan="2" name="csiModbusEngine" key="3551" apiPort="42423" portType="1"
portNum="1" cval="1000,502,20,1,0" configFile="server.xml"/>
</system_apps>

<user_apps>
</user_apps>
```

</configuration>

Task Manager Attributes

apiPort="n" - The port on which the respective API should listen on is given as apiPort="n". The access is local only in production, but may be set to global as a build option using the switch found in taskmanager.h that will be part of all system application builds.

maxglobaldata="n" - specifies maximum number of data objects that will be configured for use by all applications.

lockout="n" - specifies the number of seconds that must elapse between NAND file writes of persistent data values. This is provided in order to prevent too-frequent writing of NAND Flash, which shortens the life of the memory.

configFile="xxx" – sets the file name that the task manager's load and save requests will act on.

System/User Task Attributes

chan="n" - Channel number is 0..31 where system channel zero is always the data engine. There are 32 system channels and 32 user channels for data processing. The channel number is provided to system and user tasks as argv[1] for use in creating unique error log names when multiple instances of the same executable will be run on the system.

name="xxx" - The executable that should be invoked as a Linux process is given as 'name'.

key="n" - The key is an arbitrary value that is specific to the process, and generally will direct the process to select any applicable execution options or features.

apiPort="n" - Specifies the port that the Json listener is listening to.

portType="n" - Not to be confused with any API usage, this port is provided to communication engines for use as applicable. Refer to communication engine documentation. Port type is 0 for none, 1 for Ethernet, 2 for serial (tty).

portNum="n" - Provided to communication engines for their use. Valid port numbers for BB4-8422 are 0 or 1 for Ethernet, and 2 or 3 for serial (tty) ports.

alivePeriod="n" (Json "alivePeriod", integer) - This time period in seconds is how long the task manager will wait for this task to update its alive counter before killing and restarting this task. If set to zero, the alive check is disabled for this task. If set to a non-zero value, then the application must increment its alive counter within this time period to indicate to the task manager that it is still alive.

configFile="xxx" - This specifies a file name that will be parsed upon the task being issued the ‘load’ request, or saved upon being issued the ‘save’ request. The normal framework for application implementation is to retrieve configuration information from the task’s database, and only (optionally) parse an XML configuration file when requested to do so. This file should NOT be automatically parsed at startup because any changes made via the API following startup are stored in the database automatically, but only stored in the XML file if the user remembered to request saving the file.

cval="n1,n2,n3,n4,n5" - Used optionally for application specific purposes. The data engine does not currently use or require any cvals. Refer to specific applications for information regarding any use of cval’s. The Modbus engine, for example, does make use of the cvals. If provided, the cval attribute must consist of a series of 5 integer values separated by commas.