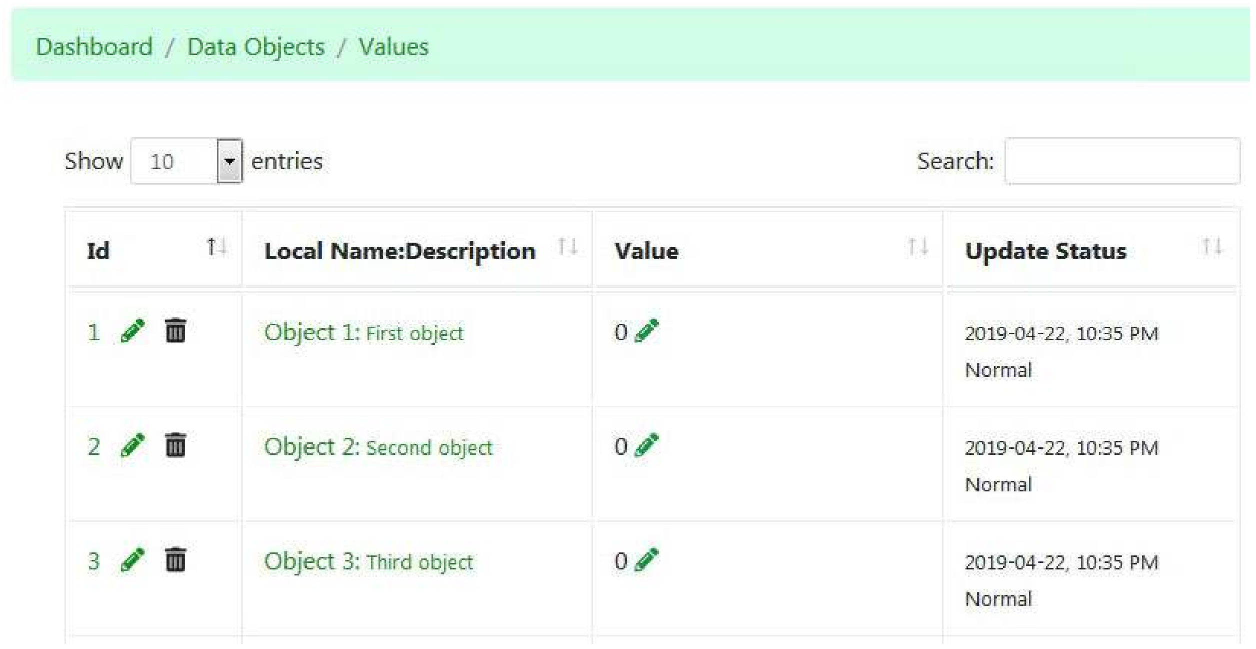
BB4-8422 Data Objects

The IoTServer is, at its core, protocol agnostic. Its internal data objects are generic in nature. These local objects may be represented in various ways that are specific to various protocols. But the content is universal and can be shared across all protocols supported by the IoTServer device. The "Data Engine" pages are where you view the present values of the objects, and specify the internal format for each object. Data objects may be natively integer, floating point, or character string. When accessed by various protocols, data type conversions are made automatically.

Global Values

The terms "global values" and "local objects" are used interchangeably in the IoTServer. The values are "global" in the sense that they are shared among all protocols and applications known to the IoTServer. The objects are "local" in the sense that they are stored and maintained within the IoTServer itself. The objects themselves are protocol agnostic, but may be presented to the outside world using protocol specific representations.

Global Object Values

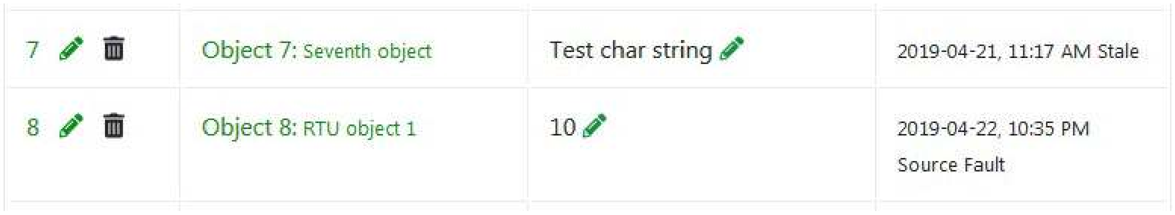


Id	Local Name:Description	Value	Update Status
1	Object 1: First object	0	2019-04-22, 10:35 PM Normal
2	Object 2: Second object	0	2019-04-22, 10:35 PM Normal
3	Object 3: Third object	0	2019-04-22, 10:35 PM Normal

The Global Values page shows the present values of all of the data objects in the system. The time at which the object was last updated is displayed along with the object's present status. Status indications may be any of the following:

- “Normal” = activity producing data for this object is proceeding normally
- “Default” = as a result of some error, the object has assumed its default value
- “Database-Value” = object is persistent and value was retrieved from values database
- “Inactive” = object is not configured

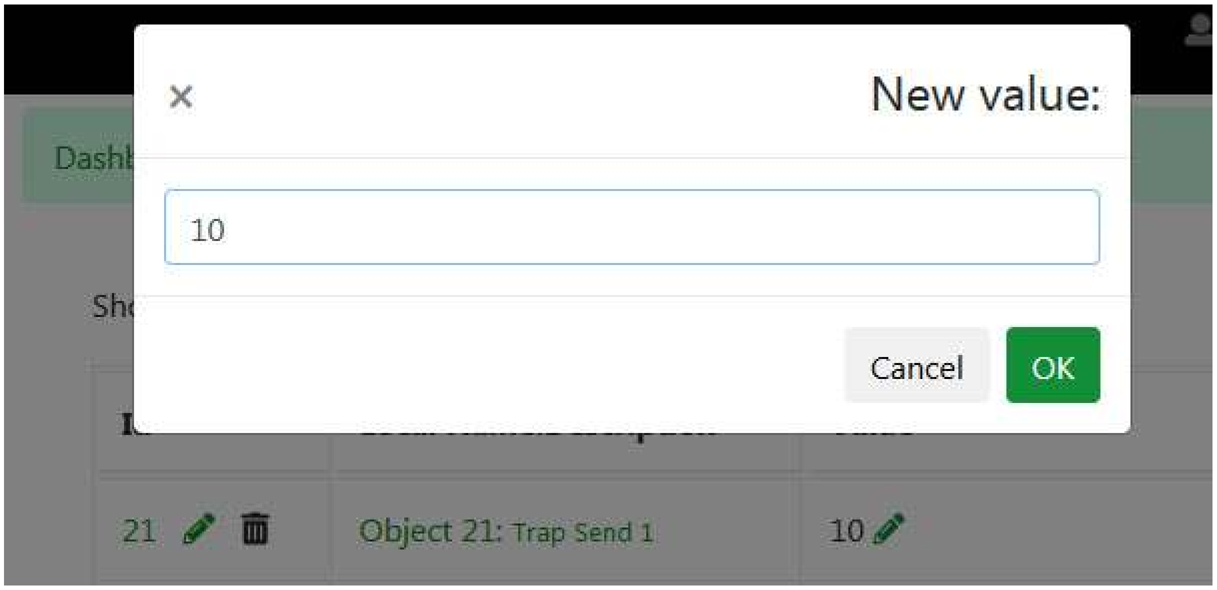
- "Stale" = refresh time exceeded since last update
- "Zero-Uninitialized" = object is configured but not yet processing data
- "Overridden" = normal data activity has been overridden
- "Source Fault" = a communication client has reported an error in attempting to provide data
- "Unknown" = a fault code not recognized by the Json API is present



7	Object 7: Seventh object	Test char string	2019-04-21, 11:17 AM Stale
8	Object 8: RTU object 1	10	2019-04-22, 10:35 PM Source Fault

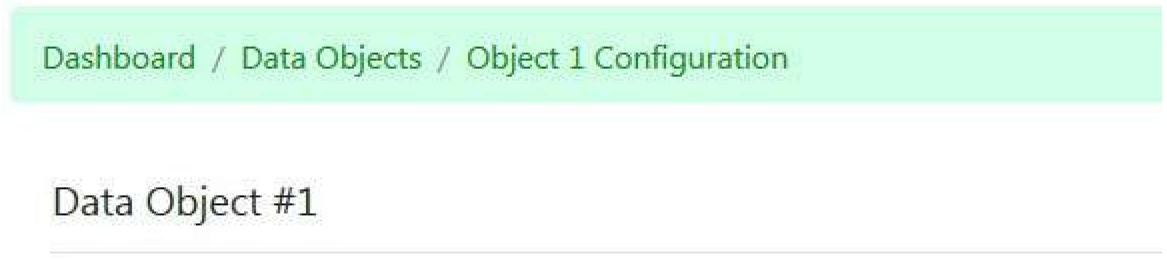
You may modify the configuration (or definition) of a data object by clicking the object number or pencil icon next to the object number. You can also add new objects - start by clicking the pencil icon of an object near the point where you want to add objects (normally the end of the list).

Changing Object Value



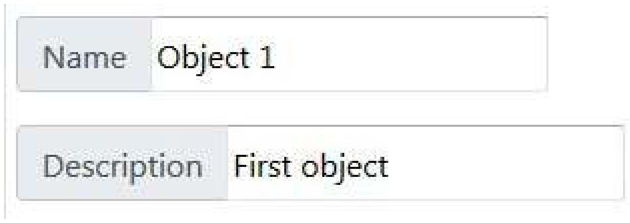
Change an object's value by clicking the pencil icon next to the present value. A dialog will appear with the present value. Modify the value as desired and click Ok. Note that any values entered may be overwritten by the next value read from a device connected via the network and polled by one of the protocols in the IoTServer.

Global Object Edit


Dashboard / Data Objects / Object 1 Configuration

Data Object #1

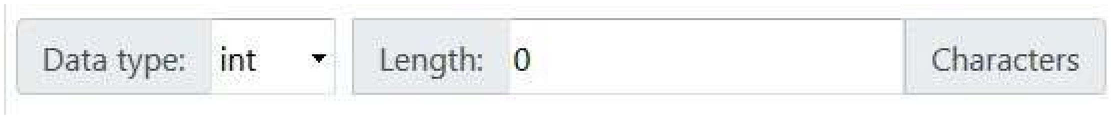
Number – The local object number being defined (first object is 1).


Name: Object 1

Description: First object

Name – Arbitrary name for the data object, generally visible in web UI's. CSV import will insert "Object name N" if omitted.

Description – Arbitrary description of the object provided by the user. CSV import will insert "Object N description" if omitted.

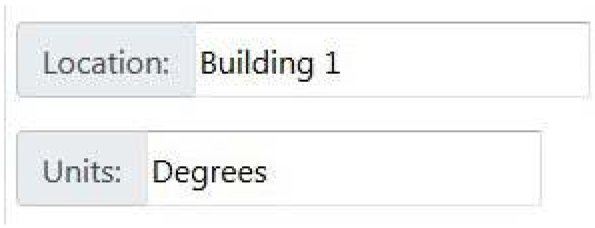

Data type: int

Length: 0 Characters

Type – Valid parameters for type, referring to native data type, are as follows (CSV will assume integer if omitted):

Type label	Description
Int	32-bit integer
Int64	64-bit integer
Real	Floating point (double precision)
Char	ASCII character string

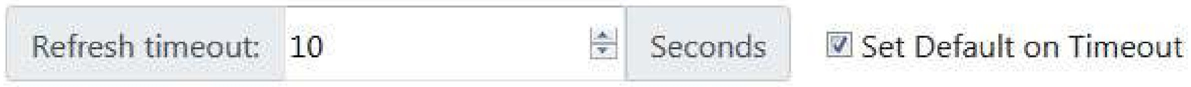
Length – If type is "char", then this specifies how many character long the string is. Not used for any type other than char. Length should be an integer. CSV import will assume maximum permitted value if omitted.


Location: Building 1

Units: Degrees

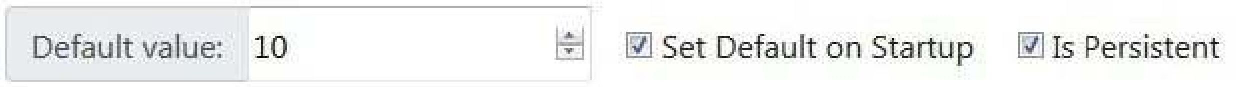
Location – Arbitrary location string further documenting the object. CSV import will insert "Location N" if omitted.

Units – Arbitrary units string further documenting the object. CSV import will insert "No units" if omitted.


Refresh timeout: 10 Seconds [✓ Set Default on Timeout]

Refresh Timeout – A timeout in seconds, if nonzero. If the object is not updated within this amount of time, its status will be changed to "stale". If zero, the timeout feature is disabled and the object will never become stale. Timeout should be an integer. Will default to zero.

Default on Timeout - Check to set object to default value upon refresh timeout (in addition to setting status to "stale"). This is especially useful for SNMP table walks of alarm tables to cause object value to return to zero when alarm is no longer found to exist.

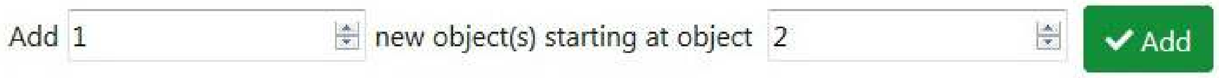

Default value: 10 [✓ Set Default on Startup] [✓ Is Persistent]

Default Value – Default value to be applied upon startup if requested, may be integer or real (but not character). Will default to zero.

Default On Startup – Check to indicate that this object should be set to the default value at startup.

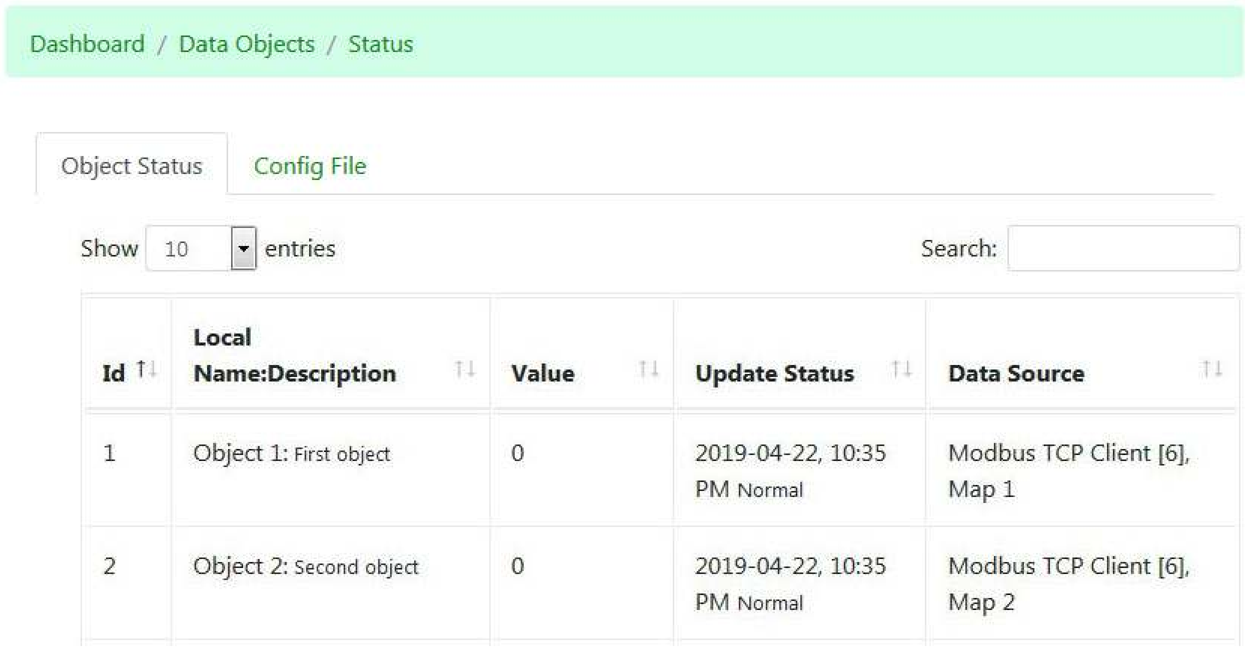
Is Persistent – Check to indicate this object is "persistent", which means saved in non-volatile memory. This non-volatile memory is implemented as a database stored in Flash memory. Upon startup, the object will assume its last known value as found in the database. Any changes to the object's value will be recorded in the database for future use. If not persistent, objects will start up with a value of zero, unless the "default on start" has been selected.

NOTE: You would not normally select both "default on startup" and "is persistent" because these will contradict each other.


Add 1 new object(s) starting at object 2

To add or insert new objects, enter a number of objects to add, and select a starting point. Then click the Add button.

Object Status

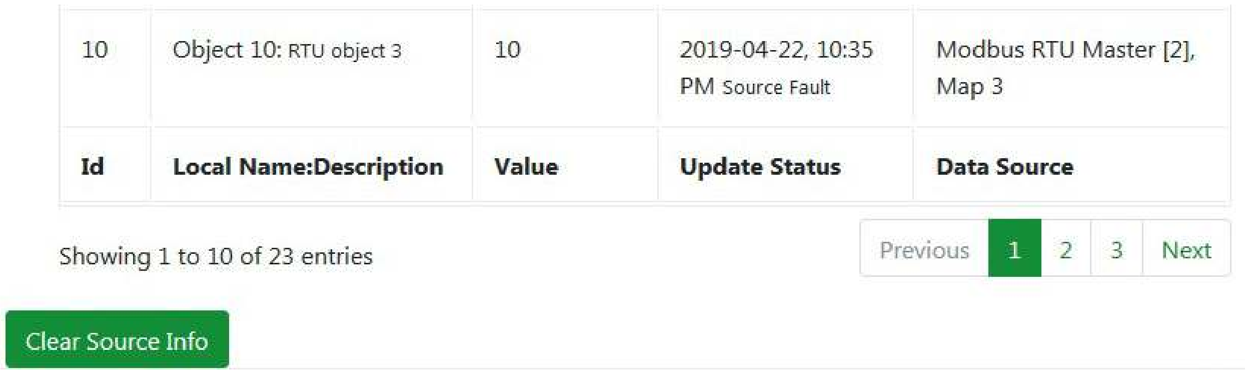


Id	Local Name:Description	Value	Update Status	Data Source
1	Object 1: First object	0	2019-04-22, 10:35 PM Normal	Modbus TCP Client [6], Map 1
2	Object 2: Second object	0	2019-04-22, 10:35 PM Normal	Modbus TCP Client [6], Map 2

The Object Status page is a mirror image of the Global Values page, but with additional diagnostic information. If the object's value is being updated by a protocol function somewhere in the system, that protocol mapping will be displayed here.

The Data Source tells you where values in this object are coming from, e.g. Modbus TCP Client. The number in brackets is the task number (from task status page). The second item in the data source is the specific map number or rule number. The map or rule will most often be a "read map".

Note that only data sources known to the IoTServer will be listed here, namely client functions that are actively polling other devices to collect data. When the IoTServer is functioning as a server or slave, and other remote clients or masters are writing data to the IoTServer, the IoTServer has no way of knowing what these sources are. Therefore, there can be sources of data not explicitly listed here.



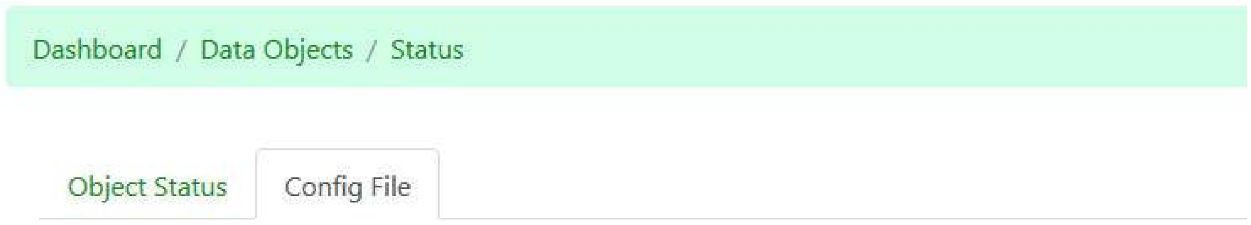
Id	Local Name:Description	Value	Update Status	Data Source
10	Object 10: RTU object 3	10	2019-04-22, 10:35 PM Source Fault	Modbus RTU Master [2], Map 3
Id	**Local Name:Description**	**Value**	**Update Status**	**Data Source**

Showing 1 to 10 of 23 entries

The maps or rules for any protocol client that is actively polling other devices for data will have a "Set Source Info" button to "claim" objects for that protocol. That button must be clicked before any information will show up in the Data Source column here.

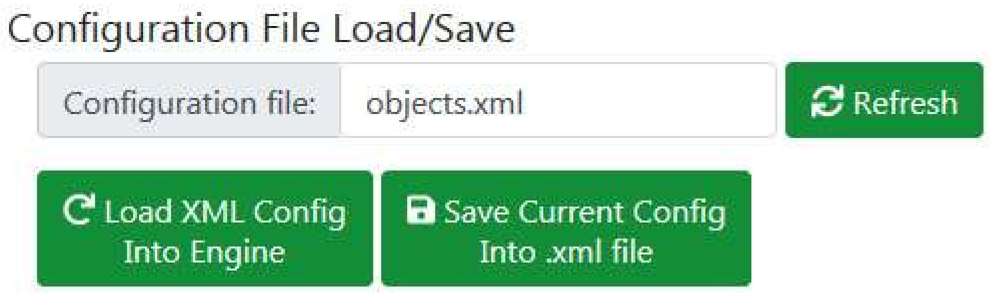
If there are conflicts, i.e. two different clients trying to provide data to the same object, you will see error messages when the Set Source Info button is clicked. To clear the source info and start over on the claim process, click the Clear Source Info button here.

Object Config File



All of your configuration information is stored in an internal database each time you click the Save button on any page where configuration entries may be made. To make configuration portable from one device to another, and for purposes of retaining a backup copy, the configuration information may be exported and imported as XML or CSV files. This page is where your configuration file management takes place.

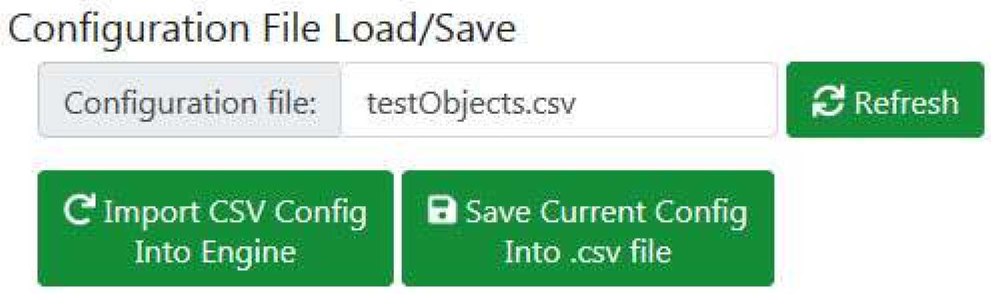
It is important to note that the XML file saved within any one system function will contain the configuration information for only that function. Depending on overall system configuration, a complete backup may involve more than one XML or CSV file.


Configuration File Load/Save
Configuration file:
objects.xml
Refresh
Load XML Config Into Engine
Save Current Config Into .xml file

XML Files: When an XML file has been selected, click the Load button to clear the configuration database and reload configuration from the given XML file.

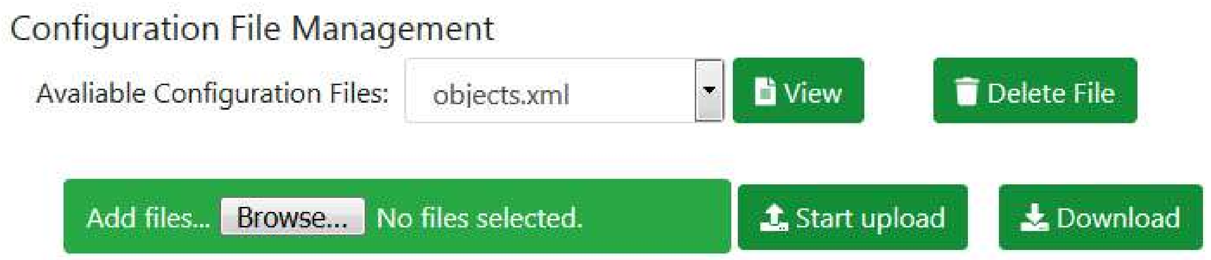
Select an existing name to overwrite or enter a new file name, and then click Save to write the current configuration to the file in XML format.

You may type in a new name in the file name window for purposes of saving a new file. If you click the Refresh button, the file name will be restored to the name currently loaded into the client. The name could have been changed by selecting a file from the list below, or by typing in a new name. If the displayed name has not yet been used, then Refresh will restore the file name to what was most recently loaded.


Configuration File Load/Save
Configuration file:
testObjects.csv
Refresh
Import CSV Config Into Engine
Save Current Config Into .csv file

CSV Files: If a CSV file is selected, the Load and Save buttons will change into load/save CSV buttons. When a CSV file has been selected, click the Load button to clear the configuration database and reload configuration from the given CSV file.

Select an existing name to overwrite or enter a new file name, and then click Save to write the current configuration to the file in CSV format.

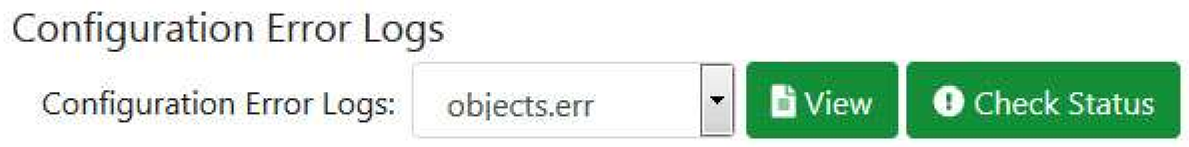


The drop-down list will show a list of all configuration files currently found in the device's configuration folder. When you select an XML or CSV file from this list, the name will be copied to the Load/Save section of this page for pending load or save.

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

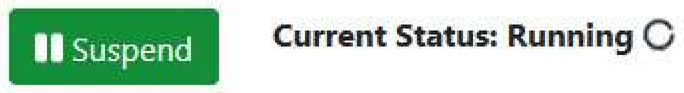
You may upload files to the IoTServer from your PC. Start by clicking Browse, and then use the browser's file dialog to locate the file on your PC. Once a file is selected on your PC, click the "Start upload" button to initiate the transfer.

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



Any time an XML or CSV file is loaded, an error log file is generated. The error log file will be given the same name as the loaded file, but with ".err" as the suffix instead of ".xml" or ".csv". You may view the error log by selecting it from the list and clicking View.

Status is normally displayed in a message box at the top of the screen when the load or save operation is complete. But if you want to double check the status of the previous file operation, click Check Status.



The task needs to be suspended while a file load operation is in progress to prevent acting on any partial configurations. This suspend/resume operation will normally happen automatically as part of the sequence invoked by the Load button. The task can be explicitly suspended here. The Suspend button will become a Resume button when the task is suspended. Click Resume to continue operation. The current status is always displayed here.

Global Object XML Files

Example XML File

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<configuration>
<data_objects>
<obj id="1">
<set type="int" name="Object 1" desc="First object"/>
<set location="Building 1" units="Degrees"/>
<set refresh="600"/>
<set defvalue="10" defonstart="1" persistent="1"/>
</obj>
<obj id="2">
<set type="int" name="Object 2" desc="Second object"/>
<set location="Building 2" units="Degrees"/>
<set refresh="600"/>
<set defvalue="20" defonstart="1" persistent="1"/>
</obj>
<obj id="3">
<set type="int64" name="Object 3" desc="Third object"/>
<set location="Building 3" units="Degrees"/>
<set refresh="600"/>
<set defvalue="0" defonstart="1" persistent="1"/>
</obj>
</data_objects>
</configuration>
```

Global Object <data_objects> Section

Each XML "paragraph" about an object starts with an ID number, which will be the object number that is universally accessible to all applications in the system.

For each object:

type="xxx" - Valid parameters for type, referring to native data type, are as follows:

Type label	**Description**
int	32-bit integer
int64	64-bit integer
real	Floating point (double precision)
char	ASCII character string

length="n" – If type is “char”, then this specifies how many character long the string is. Not used for any type other than char. Length should be an integer.

name="ObjectName" – arbitrary name for the data object, generally visible in web UI’s.

desc="description" – arbitrary description of the object provided by the user.

location="location" – arbitrary location string further documenting the object.

units="unitsName" – arbitrary units string further documenting the object.

refresh="n" – A timeout in seconds, if nonzero. If the object is not updated within this amount of time, its state will be changed to “stale”. If zero, the timeout feature is disabled and the object will never become stale. Timeout should be an integer.

defvalue="n.nn" – Default value to be applied upon startup if requested.

defontimeout="1" - Flag set to "1" to indicate that this object should be set to the default value upon refresh timeout.

defonstart="1" – Flag set to “1” to indicate that this object should be set to the default value at startup, or zero otherwise (zero assumed if omitted).

persistent="1" – Flag set to “1” to indicate this object is “persistent”, which means saved in non-volatile memory. This non-volatile memory is implemented as a database stored in Flash memory. Upon startup, the object will assume its last known value as found in the database. Any changes to the object’s value will be recorded in the database for future use. If not persistent, set flag to zero (zero assumed if omitted).

Global Object CSV Files

A CSV file may be imported to configure various aspects of the IoTServer (or Babel Buster gateway). A single CSV file may contain multiple sections. When a file including an “Objects” section is imported by the Data Engine, local objects will be configured. When a file including one or more “Modbus” sections is imported by an instance of the Modbus Engine, Modbus gateway functionality will be configured. The same Modbus file may be imported by a Modbus Client or Modbus Server, and either RTU or TCP, and only those sections of interest to that Modbus function will be imported. The CSV file may also contain one or more SNMP sections, and so forth.

A section begins when the word “Begin” appears in the first column of a line. All lines up to and including a line that begins with the word “End” will be taken to be part of that section.

The line immediately following the “Begin” line must be a header line. A Header line is one which labels the columns of data that will follow the Header line.

All lines following the Header line are data lines that are expected to contain the same number of columns as the Header line, and whose contents are defined by the labels found in each column of the Header line.

Labels in the section Begin and End lines, and labels in the Header line are NOT case sensitive and will be interpreted equally whether upper case, lower case, or some combination of both (for readability).

Labels may NOT contain embedded spaces. A label is terminated by a comma, line-end, or space. Labels may not be encapsulated in quote characters; however, data content in data lines may be encapsulated in quote characters and may contain embedded spaces or blanks if quoted.

Some labels in the Header line may be considered optional. The minimum required columns are indicated in the definition of each data section.

Columns in the Header line do not have to follow any particular order. They may be rearranged to the user's liking. The only restriction is that data in subsequent data lines must match up with the labels placed in the Header line. Data lines may contain fewer columns than the Header line, but may not contain more. Data columns that the user wishes to deliberately omit, but omit between included columns, should be indicated by place holder commas (which will simply appear as blank cells in a spread sheet program).

A Begin line will contain three columns:
Column 1: BEGIN
Column 2: Function as noted below
Column 3: Sub-function as noted in definition of the Function.

Functions may be any of the following (with this listed expanded from time to time):

- LOCALDATA
- MODBUS
- SNMP

Sub-functions:

MODBUS

- DEVICES
- READMAPS
- WRITEMAPS
- SERVERMAPS
- SERVERREMAPS

NOTE: A Modbus CSV file may contain both client and server sections as the respective functionality will select sections relevant to its purpose. It should also be noted that the same application, csiModbusEngine, can function as both client and server (master and slave).

Global Objects CSV Example

The following illustrates a valid local object configuration CSV file. Columns omitted in any given line will assume default values.

Note: Pay special attention to quote characters. It is not necessary to explicitly include them when entering labels in the spread sheet program. The CSV export (file save as) will do that for you. More importantly, if you type them in manually, they will be the "fancy" version of quote characters and will be explicitly included in the resulting object name. Only the plain quotes inserted by the spread sheet program will be recognized as escape characters for embedded commas.

```
BEGIN,LOCALDATA,OBJECTS
NUMBER,TYPE,LENGTH,NAME,DESC,LOCATION,UNITS,REFRESH,DEFVALUE,DEFONTIMEOUT,DEFONSTART,PERSISTENT
1,INT,0,My object 1,Object 1 description,Location 1,No units,600,-99.000000,N,N,N
2,INT,0,My object 2,Object 2 description,Location 2,No units,600,-99.000000,N,N,N
3,INT,0,My object 3,Object 3 description,Location 3,No units,600,-99.000000,N,N,N
4,INT,0,"Thing 1,2,3",Object 4 description,Location 4,No units,600,-99.000000,N,N,N
11,REAL,0,Object name 11,Object 11 description,Location 11,No units,600,-99.000000,N,N,N
12,REAL,0,Object name 12,Object 12 description,Location 12,No units,600,-99.000000,N,N,N
21,CHAR,80,Object name 21,Object 21 description,Location 21,No units,0,0.000000,N,N,N
22,CHAR,80,Object name 22,Object 22 description,Location 22,No units,0,0.000000,N,N,N
END
```

LOCALDATA OBJECTS Data Section

For each object, the following data columns are recognized, with minimum required columns denoted "REQUIRED":

Number – (REQUIRED) The local object number being defined (first object is 1).

Type – Valid parameters for type, referring to native data type, are as follows (CSV will assume integer if omitted):

Type label	Description
Int	32-bit integer
Int64	64-bit integer
Real	Floating point (double precision)
Char	ASCII character string

Length – If type is "char", then this specifies how many character long the string is. Not used for any type other than char. Length should be an integer. CSV import will assume maximum permitted value if omitted.

Name – Arbitrary name for the data object, generally visible in web UI's. CSV import will insert "Object name N" if omitted.

Desc – Arbitrary description of the object provided by the user. CSV import will insert "Object N description" if omitted.

Location – Arbitrary location string further documenting the object. CSV import will insert "Location N" if omitted.

Units – Arbitrary units string further documenting the object. CSV import will insert "No units" if omitted.

Refresh – A timeout in seconds, if nonzero. If the object is not updated within this amount of time, its state will be changed to "stale". If zero, the timeout feature is disabled and the object will never become stale. Timeout should be an integer. Will default to zero.

DefValue – Default value to be applied upon startup if requested, may be integer or real (but not character). Will default to zero.

DefOnTimeout - Flag set to "Y" to indicate that this object should be set to the default value upon refresh timeout.

DefOnStart – Flag set to "Y" to indicate that this object should be set to the default value at startup, or "N" otherwise ("N" assumed if omitted).

Persistent – Flag set to "Y" to indicate this object is "persistent", which means saved in non-volatile memory. This non-volatile memory is implemented as a database stored in Flash memory. Upon startup, the object will assume its last known value as found in the database. Any changes to the object's value will be recorded in the database for future use. If not persistent, set flag to "N" ("N" assumed if omitted).