AB Rockwell ControlLogix Simulator Connector

A built-in ControlLogix controller simulator for testing solutions that use the AB Rockwell ControlLogix Connector without a PLC.

- Name: **ControlLogixSimulator**
- Version 1.0.0.0
- Protocol: CIP over EtherNet/IP (TCP port 44818)
- Interface: TCP/IP
- Runtime: Windows (.NET 4.8 executable); serves clients on any platform
- Configuration:
  - Devices / Channels
- Available from FrameworX 10.1.6

Overview
Starting the Simulator
Command Line Parameters
Configuration File
Limitations

Overview

The ControlLogix Simulator is a built-in tool that behaves like a Rockwell ControlLogix controller on the network. It answers the EtherNet/IP and CIP requests of the AB Rockwell ControlLogix Connector, so a solution can read and write controller tags, show moving values and react to operator commands with no PLC connected. Typical uses are developing and testing displays, alarms and scripts, training, demonstrations, and verifying a migrated application before the real controller is available.

The tags the simulator serves come from a JSON configuration file. Each tag has a Logix name and data type, an optional value generator that keeps it moving, and optional rules that give buttons a visible effect, for example a Start command that sets a Running feedback tag.

From the connector's point of view the simulator is a controller at an IP address. No change to the channel, node or points is needed beyond pointing the node at the simulator.

Starting the Simulator

From the Designer

1. Go to **Devices / Channels** and select a channel whose protocol is **ControlLogix**.
2. Click **Run ControlLogix Simulator**. The button appears for ControlLogix channels, in the same place as *Run Modbus Simulator* for Modbus channels.

The simulator opens in a console window and starts with the tag list in `Documents\FrameworX\ToolsSettings\ControlLogixSimulator.config.json`. The first launch creates that file from the sample configuration installed with the product; edit it to serve your own tags, then restart the simulator.

Only one simulator is started from the Designer at a time. To simulate several controllers, run the executable directly as described below.

Connecting a Channel

Configure the channel and node as for a real controller:

- **Channel**: protocol ControlLogix. ProtocolOptions model **Others** or **1756-L8X**, for example `Others;3;0;0`.
- **Node**: PrimaryStation `127.0.0.1;44818;0`, that is, the address the simulator listens on, port 44818 and the slot from the configuration file.
- **Points**: the usual addresses, for example `REAL:Tank1_Level:true`, `BOOL:Agitator_Running:true` or, for a program-scope tag, `BOOL:Program:MainProgram.Pump1_Start:true`.

> By default the simulator listens only on this computer (127.0.0.1). It accepts reads and writes from any client without authentication, so make it reachable from the network only on a test network, with the `--bind` option.

Command Line Parameters

The executable is installed in the product's `Protocols` folder:

```
ControlLogixSimulator.exe [--config <tags.json>] [--port 44818] [--bind 127.0.0.1] [--log <file>] [--verbose]
```

Parameter	Default	Description

--config	ControlLogixSimulator.config.json next to the executable	Full path of the tag configuration file.
--port	44818	TCP port.
--bind	127.0.0.1	Local address to listen on. Use 0.0.0.0 or a network card address to accept other computers. To simulate several controllers on one computer, run one simulator per controller, each on its own loopback address (for example 127.0.0.10 and 127.0.0.20; every 127.x.x.x address is local on Windows).
--log	off	Appends one line per request to the file: time, client, command, service, tag path and status. Useful to find a tag that the solution asks for but the configuration does not contain.
--verbose	off	Also prints every request in the console window.

Example with two controllers:

```
ControlLogixSimulator.exe --config "C:\Sim\Line1.json" --bind 127.0.0.10 --log "C:\Sim\Line1.log"
ControlLogixSimulator.exe --config "C:\Sim\Line2.json" --bind 127.0.0.20 --log "C:\Sim\Line2.log"
```

Press Ctrl+C to stop the simulator. On exit it lists the tag names that clients requested but the configuration does not contain.

Configuration File

```
{
  "controller": { "name": "DemoLine", "slot": 0 },
  "tags": [
    { "name": "Tank1_Level", "type": "REAL", "initial": 50,
      "gen": { "kind": "sine", "min": 10, "max": 90, "periodSec": 60 } },
    { "name": "Agitator_Running", "type": "BOOL", "initial": 0,
      "gen": { "kind": "square", "periodSec": 20 } },
    { "name": "Product_Name", "type": "STRING", "initial": "Demo Product A", "gen": null },
    { "name": "Program:MainProgram.Pump1_Start",   "type": "BOOL", "initial": 0, "gen": null },
    { "name": "Program:MainProgram.Pump1_Running", "type": "BOOL", "initial": 0, "gen": null }
  ],
  "rules": [
    { "onWrite": "Program:MainProgram.Pump1_Start", "equals": 1,
      "set": [ { "tag": "Program:MainProgram.Pump1_Running", "value": 1 } ] }
  ]
}
```

Tags

- **name**: the Logix tag name, matched without regard to case. Program-scope tags use `Program:<program>.<tag>`. A structure member or array element can be given by its full path, for example `Motor1.Speed` or `Temps[3]`.
- **type**: BOOL, SINT, INT, DINT, REAL or STRING (the standard 82-character Logix STRING).
- **initial**: start value.
- **gen**: optional value generator, updated every 250 ms.

kind	Parameters	Behavior
sine	min, max, periodSec	Sine wave between min and max.
ramp	min, max, periodSec	Rises from min to max, then starts again.
random	min, max, stepSec	A new random value between min and max every stepSec.
square	periodSec (min, max for numbers)	A BOOL toggles every half period; a number switches between min and max.
counter	stepSec	Adds 1 every stepSec.

When a client writes a tag that has a generator, the generator pauses for 30 seconds so the written value stays visible, then resumes.

Arrays

Elements of one array are configured as separate tags with the same base name, for example `Temps[0]` and `Temps[2]`. All elements of an array must have the same type. An element that is not listed, up to the highest listed index, reads as zero or an empty string. BOOL arrays are not supported; use individual BOOL tags.

Rules

A rule runs after a client writes the **onWrite** tag with a value equal to **equals**, and sets each listed **tag** to its **value**. Use rules to give momentary buttons a visible result, such as Start setting a Running feedback to 1 and Stop setting it to 0.

Unknown Tags

A tag that is not in the configuration file is answered as a real controller answers an unknown tag, so the point shows bad quality in the solution. The console window names each unknown tag once, and the log file records every request for it.

Limitations

- User-defined types (UDTs) are not simulated as structures; serve their members as individual tags with full paths, for example `Motor1.Speed`.
- BOOL arrays and string types other than the standard Logix STRING are not supported.
- For the Micro850 model, string tags are not supported; use the Others or 1756-L8X model with the simulator.
- Browsing the simulator's tags lists controller-scope tags with BOOL, SINT, INT, DINT, REAL and STRING types. Program-scope tags are read and written by address.
- Use the simulator for testing and demonstrations only. It is not a replacement for a controller and does not run any control logic beyond its generators and rules.

In this section...