innon
WWW.INNON.COM

Innon Niagara Essential Toolkit

User Manual

TABLE OF CONTENTS

INTRODUCTION

Licensing

The Innon Essential Niagara Toolkit product line comprises of two Niagara 4 modules we have developed to allow you to exchange data between a web application and a Niagara 4 station. The two modules are –

- HTTP module

1. JSON module

The HTTP module provides –

1. A servlet component to create a URL (endpoint) in the Niagara 4 station for a web API client to POST data to
2. A client component to allow the Niagara 4 station to construct either a
    a. POST message to transfer data from the station to a web server API OR
    b. GET message to request data from a web server API

JSON module components can be used with the above components to –

1. Decode JSON data received using the HTTP servlet component into various component slots
2. Decode JSON data received using the HTTP client component using the GET method into various component slots

The JSON component can also be used to decode any other JSON payload received by other means, e.g. MQTT subscriptions as a client.

GENERAL

Licensing

Each host is required to be licensed by Innon otherwise a warning will be displayed at start-up and the components will not function.
Licensing is split into two main areas, json and http, so each can be licensed separately.
There are no limits to the number of components that can be created within a station running on a licensed host.

Examples

Please visit our knowledge base website to find articles and examples about our HTTP and JSON modules, like this one where we use an HTTP servlet and the JSON blocks to get LoRa WAN data from a local gateway.

https://know.innon.com/how-to-read-lorawan-on-niagara-withinnon

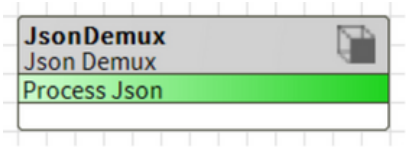
JSON DECODING TOOLKIT

Innon provides some simple components to decode JSON objects into Niagara slot values.

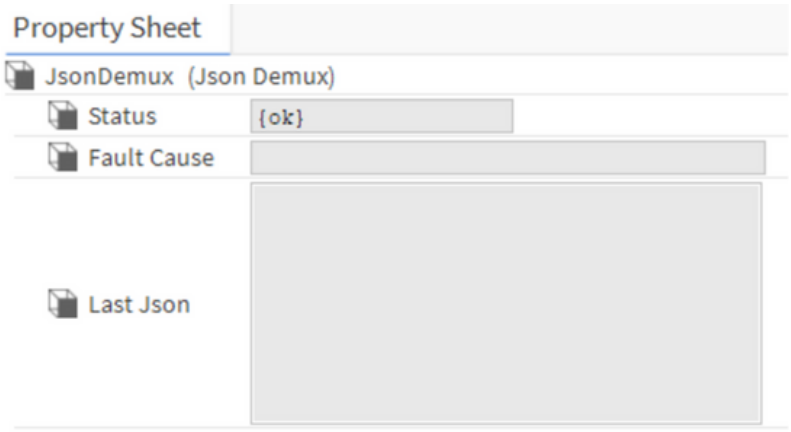
The toolkit resides in the innonJson module and palette, and consists of the following components:

- BJsonDemux
- BJsonExtractor
- BJsonArrayIndexExtractor
- BJsonQueue
- BJsonRouter

BJsonDemux



Wiresheet view (use the “pin slots” function to expose more slots)



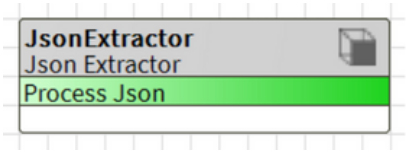
Accepts a root JSON object and outputs each element as a slot.

Element values are converted to an equivalent simple Niagara slot value type e.g. number as either BInteger or BDouble, Boolean as BBoolean etc. Nested JSON objects or arrays are output in their JSON string form to a further demux for further processing.

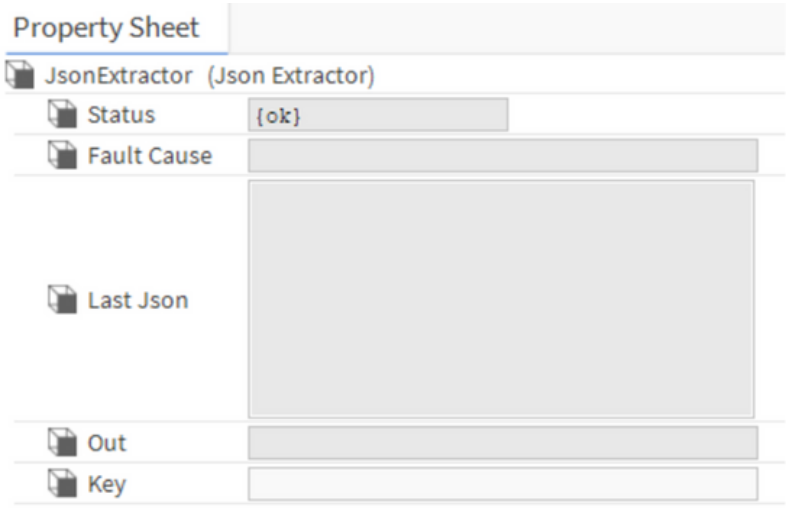
The component provides the following Actions:

- removeSlots – removes all auto-generated slots
- resetSlots – sets all auto-generated slots to their default values
- reset – invokes removeSlots and resets the lastJson to an empty string
- reprocessJson – processes the last received JSON value. Useful to ‘force’ the reprocessing of downstream components

BJsonExtractor



Wiresheet view (use the “pin slots” function to expose more slots)



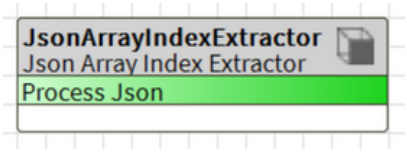
Accepts a root JSON object, finds a specific element by it’s key and outputs the String value to a slot.

The extractor will recursively search all properties of the root object until it finds the first matching key.

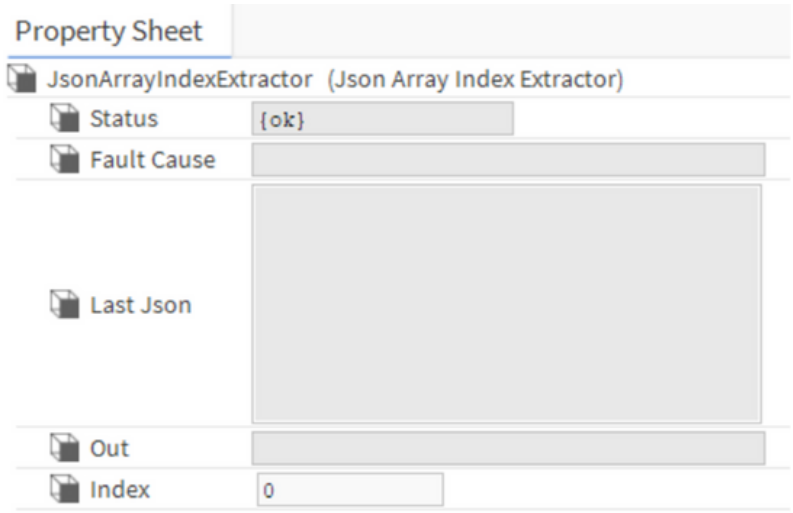
The component provides the following Actions:

- reprocessJson – processes the last received JSON value. Useful to ‘force’ the reprocessing of downstream components

BJsonArrayIndexExtractor



Wiresheet view (use the “pin slots” function to expose more slots)

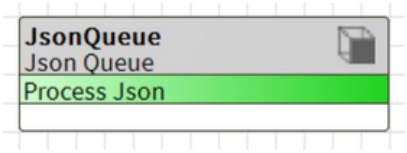


Accepts a root JSON array and finds a specific element by index and outputs the String value to a slot.

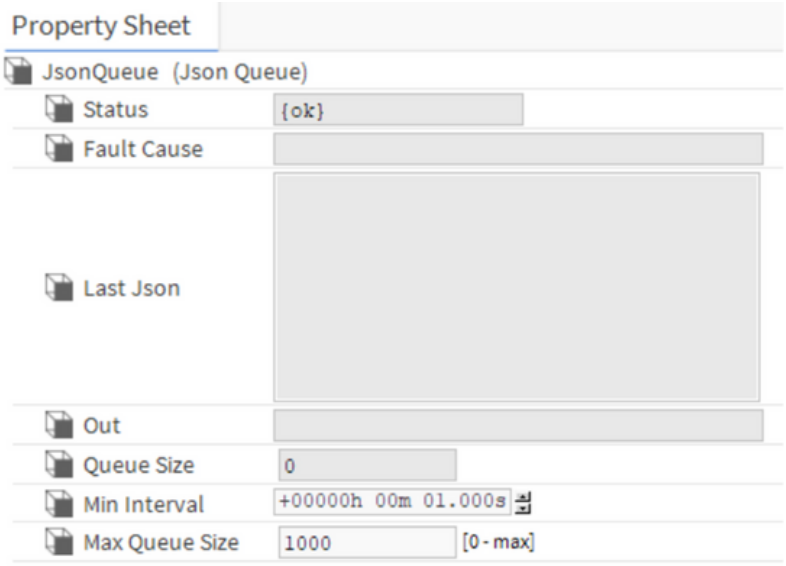
The component provides the following Actions:

- reprocessJson – processes the last received JSON value. Useful to ‘force’ the reprocessing of downstream components

BJsonQueue



Wiresheet view (use the “pin slots” function to expose more slots)



Accepts a JSON string, waiting if necessary and then writes the string back to the output slot. This is to smooth out spikes in JSON updates by waiting for a specific minimum update interval (default 1 sec), meaning the output will not update faster than specified.

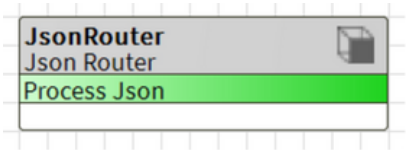
Received JSON updates are held in memory in a bounded queue (default size 1000) and processed asynchronously using the defined update interval.

If updates are received faster than they can be output, the queue will reach max capacity and the oldest updates will be discarded.

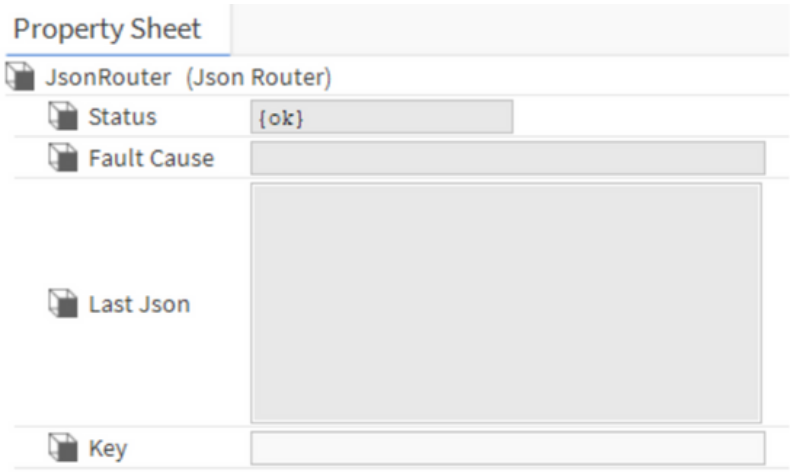
The component provides the following Actions:

- reprocessJson – processes the last received JSON value. Useful to ‘force’ the reprocessing of downstream components

BJsonRouter



Wiresheet view (use the “pin slots” function to expose more slots)



Accepts a root JSON object, finds a specific element by key and writes the entire root object to a slot with a slot name matching the element value. For each update, when a new value is detected, the entire root object is written to a new slot for the new value. The root object is output as a JSON string for further processing.

The component provides the following Actions:

- removeSlots – removes all auto-generated slots
- resetSlots – sets all auto-generated slots to their default values
- reset – invokes removeSlots and resets the lastJson to an empty string
- reprocessJson – processes the last received JSON value. Useful to ‘force’ the reprocessing of downstream components

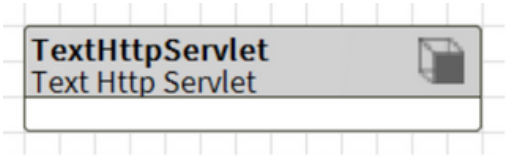
HTTP RESTFUL SERVLET AND CLIENT

Innon provides some simple RESTful components that handle JSON over HTTP.

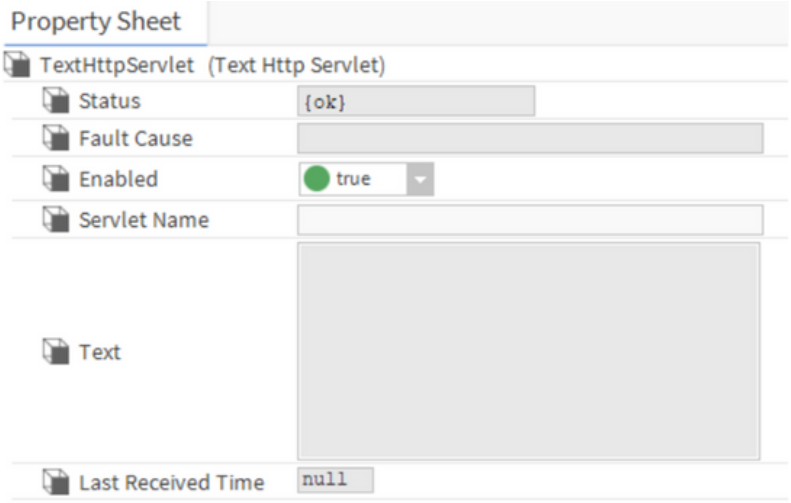
The components reside in the innonHttp module and consist of the following:

- BTextHttpServlet
- BTextHttpClient

BTextHttpServlet



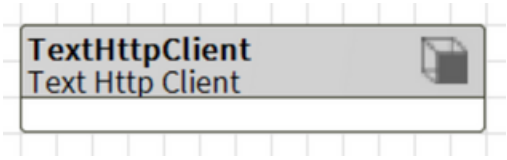
Wiresheet view (use the “pin slots” function to expose more slots)



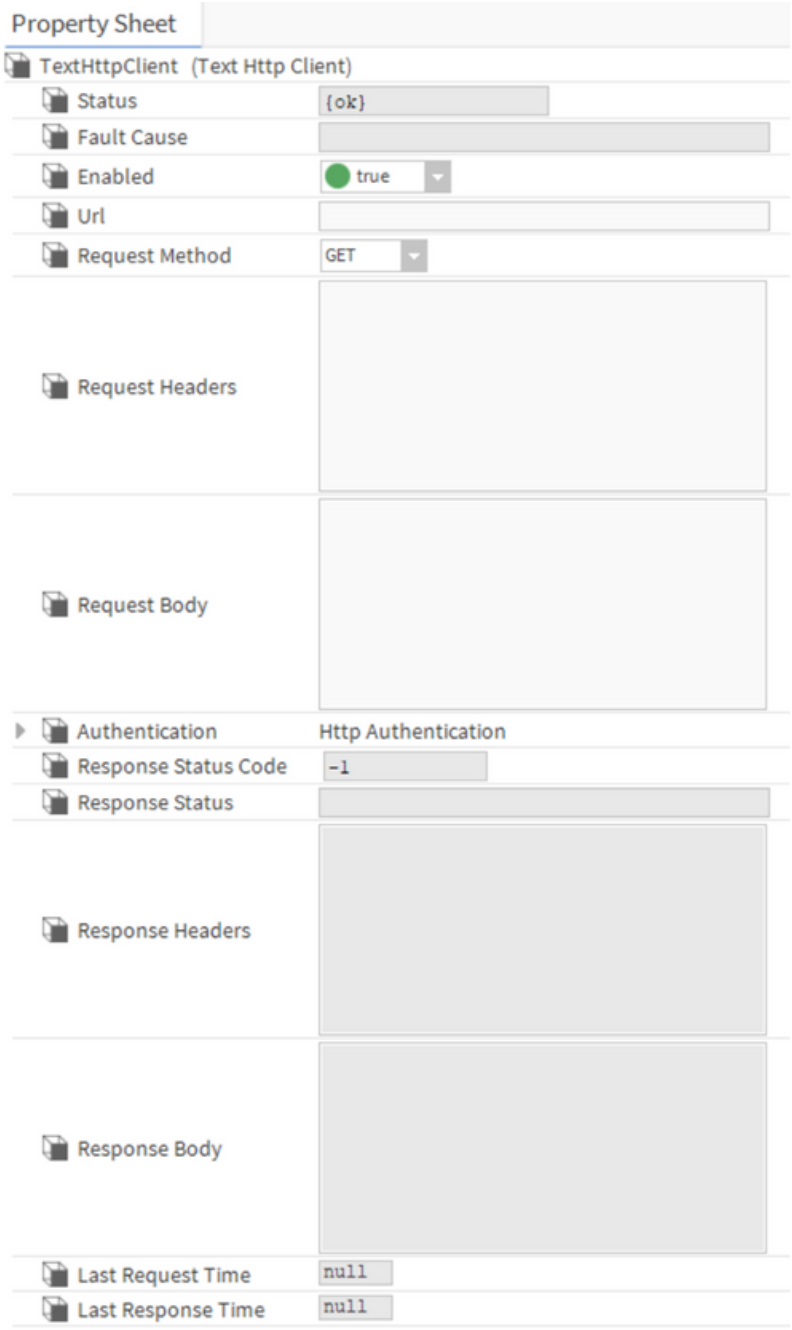
This component acts as a RESTful server and is an extension of the built-in standard web server and registers with a specific servlet name. It accepts an HTTP POST request with a JSON body and writes the JSON value to a slot.

If the servlet is configured with a servlet name of “testServlet”, it is reached by sending a request to https://<hostname>/testServlet. Changing servlet name takes effect immediately.

BTextHttpClient



Wiresheet view (use the “pin slots” function to expose more slots)



This component acts as a RESTful client and sends either an HTTP GET or POST request to a remote RESTful server. The request headers can be specified. For POST requests, a String body can be provided.

The following authentication options are supported:

- Bearer – Encrypted token
- Basic – Base64 encoded username and password
- None

The component will automatically append the necessary headers for the authentication method chosen so avoid the need to display credentials in plain text.

Following sending of a request, a String response body will be output if received along with the response headers.