



## CONTACT SUPPORT:

COMPANY NAME:	Control Concepts Inc.
SUPPORT CONTACT:	Elizabeth Scozzari
EMAIL ADDRESS:	<a href="mailto:elizabeths@controlconcepts.net">elizabeths@controlconcepts.net</a>
PHONE:	(201) 797-7900
ADDRESS:	336 Route 46, Fairfield, NJ 07004



## GENERAL INFORMATION

<b>SIMPLWINDOWS NAME:</b>	JSON SerialWrite v1.0.0
<b>CATEGORY:</b>	Utility
<b>VERSION:</b>	1.0.0
<b>SUMMARY:</b>	Module provides a method by which to write data to a corresponding tokens in a Json file provided by an associated FileProcessor Module.
<b>GENERAL NOTES:</b>	<p>Each JSON SerialWrite (Writer) module must be associated with a JSON FileProcessor (Processor) module via the Processor ID. Each individual Writer may only write up to 10 values, but multiple Writers may be associated with each Processor module, allowing for large amounts of data to be processed.</p> <p>The notation for digital request is dot based in a format known as Json Path. Each dot requests a more specific level. Arrays are supported using bracketed notation. Arrays are zero indexed. "Clients[3].cellPhone[0].Number" would write data to the the "Number" value of the first array index of the cellphone data of the fourth array indexed client in an appropriate Json file. An example file is provided with the demo program. All writes are case sensitive. A malformed or failed request results in an empty field. Any exception in the code is written to the error log and if the debug flag for the associated Processor is 'high' it will also print to the trace window.</p> <p>'Storing' data using this module only provides for temporary storage. It must still be written to the file using the 'Write_File' signal on the FileProcessor. In this way, data can be stored until a reboot. It also allows for large amounts of data to be stored and a single 'Write_File' command to process all the data at once, prolonging the life of the storage media.</p> <p>This module has support for all JSON datatypes. There is no protection included to prevent a user from writing over a Boolean with a string, for example.</p> <p>This module can also be used to store data arbitrarily. If no data exists for a token, that token will be created in the file.</p>
<b>CRESTRON HARDWARE REQUIRED:</b>	3-Series Processors <b>only</b> .
<b>SETUP OF CRESTRON HARDWARE:</b>	N/A
<b>VENDOR FIRMWARE:</b>	N/A
<b>VENDOR SETUP:</b>	N/A
<b>CABLE DIAGRAM:</b>	N/A



**CONTROL:**

<u>Signal/Function Name</u>	<u>D.S.A</u>	<u>Digital, Serial, Analog signal property definition.</u>
<b>Store_Values</b>	D	Stores all Serial_Values to the corresponding tokens on the rising edge of this signal. Values stored by this module are NOT persistent. All stored values must be written using the FileProcessor Module.
<b>Serial_Value&lt;01 – 10&gt;</b>	S	Writes data to the token in the corresponding parameter field as a String datatype. See "General Notes" for details.



## PARAMETERS:

<b>Json Path &lt;01-10&gt;</b>	S	String representing the token you wish to save with a Serial value. Writes are formatted with dot notation, with each subgroup represented by a dot. See "General Notes" for a more detailed description. Sample Json files and appropriate requests for each field are included in the demo program.
<b>Processor ID</b>	A	Setting to indicate the ID for a specific Processor Module. Each Requester must be paired with an appropriate Processor via matching Processor ID fields.



<b>TESTING:</b>	
<b>OPS USED FOR TESTING:</b>	RMC3: 1.502.3078.29223 CP3: 1.502.3079.17847
<b>SIMPL WINDOWS USED FOR TESTING:</b>	4.07.03
<b>DEVICE DB USED FOR TESTING:</b>	87.00.003.00
<b>CRES DB USED FOR TESTING:</b>	63.07.003.00
<b>SYMBOL LIBRARY USED FOR TESTING:</b>	1039
<b>REVISION HISTORY:</b>	1.0.0 – Initial Release