**Accessing Mill NC Variables through REST API**

Hurco Companies, Inc.

**Record of Changes**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Revision Description | Rev By | Date |
| 1 | Original Release | Jesse Satterwhite | 2018.09.05 |
| 2 | Revision | Jesse Satterwhite | 2022.11.16 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Introduction**

This document refers to the software provided in the sample integration package. Accessing mill NC variables is only available on Mills running v10.02.xx or higher. Mill NC variables cannot be accessed on Lathes.

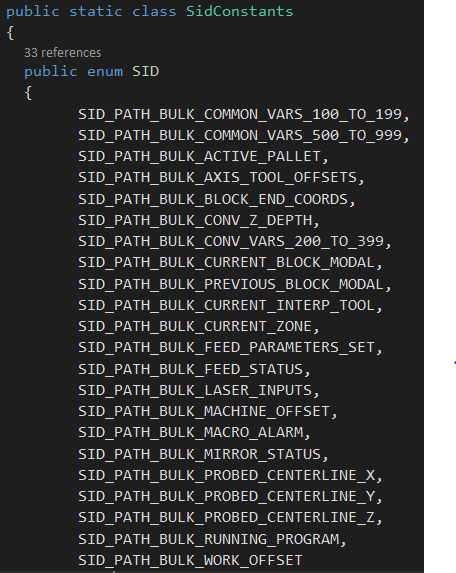
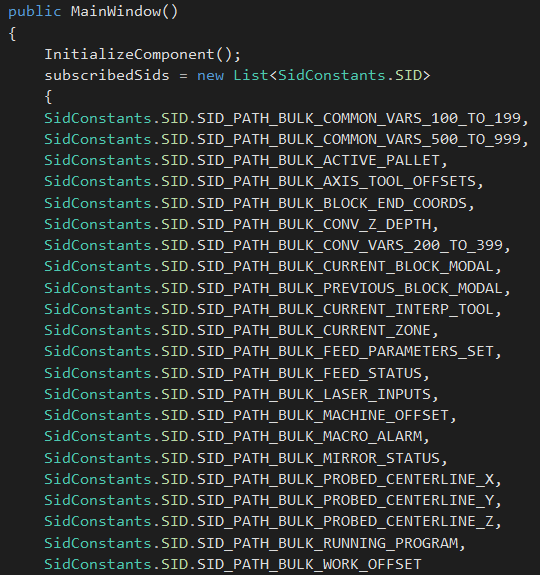
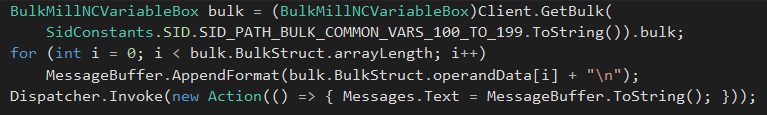
Information provided in this document assumes that a RESTful connection has already been established with the Mill’s Winmax Control. If this is not the case, refer to the steps provided in the sample integration package to connect to the Winmax Control.

In the current Winmax release, there are a total of 22 SIDs, or Source IDs, that can be used to read and write Mill NC variables during program runtime. Using REST API, the variables associated with these SIDs can be altered remotely. These altered values can be read by the Mill immediately after writing, affecting the program that is currently running.

**Available Mill NC Variable SIDs**

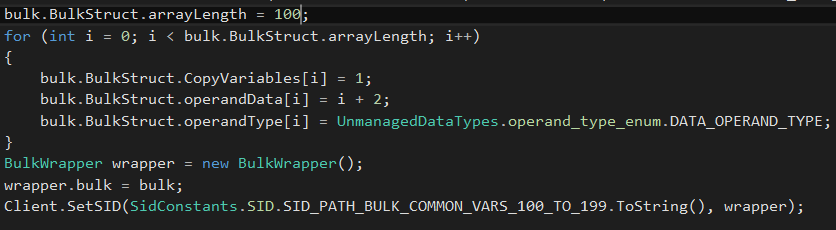
|  |  |  |
| --- | --- | --- |
| SID | arrayLength | Read Only |
| SID\_PATH\_BULK\_COMMON\_VARS\_100\_TO\_199 | 100 | False |
| SID\_PATH\_BULK\_COMMON\_VARS\_500\_TO\_999 | 500 | False |
| SID\_PATH\_BULK\_ACTIVE\_PALLET | 1 | True |
| SID\_PATH\_BULK\_AXIS\_TOOL\_OFFSETS | 3 | False |
| SID\_PATH\_BULK\_BLOCK\_END\_COORDS | 6 | True |
| SID\_PATH\_BULK\_CONV\_Z\_DEPTH | 1 | True |
| SID\_PATH\_BULK\_CONV\_VARS\_200\_TO\_399 | 200 | False |
| SID\_PATH\_BULK\_CURRENT\_BLOCK\_MODAL | 130 | False |
| SID\_PATH\_BULK\_PREVIOUS\_BLOCK\_MODAL | 130 | False |
| SID\_PATH\_BULK\_CURRENT\_INTERP\_TOOL | 1 | True |
| SID\_PATH\_BULK\_CURRENT\_ZONE | 1 | True |
| SID\_PATH\_BULK\_FEED\_PARAMETERS\_SET | 1 | False |
| SID\_PATH\_BULK\_FEED\_STATUS | 1 | True |
| SID\_PATH\_BULK\_LASER\_INPUTS | 1 | True |
| SID\_PATH\_BULK\_MACHINE\_OFFSET | 1 | False |
| SID\_PATH\_BULK\_MACRO\_ALARM | 1 | False |
| SID\_PATH\_BULK\_MIRROR\_STATUS | 1 | True |
| SID\_PATH\_BULK\_PROBED\_CENTERLINE\_X | 1 | False |
| SID\_PATH\_BULK\_PROBED\_CENTERLINE\_Y | 1 | False |
| SID\_PATH\_BULK\_PROBED\_CENTERLINE\_Z | 1 | False |
| SID\_PATH\_BULK\_RUNNING\_PROGRAM | 1 | True |
| SID\_PATH\_BULK\_WORK\_OFFSET | 1 | False |

**Using GetBulk with REST API**

1. **Use REST API to establish a RESTClient.**Refer to the supporting documentation on connecting to the Winmax control for help with this step. The following steps will be additions to the established RESTClient setup.
2. **Add the Mill NC variable SIDs to the list of SidConstants.**
3. **Add the Mill NC variable SIDs the list of SIDs that the application will subscribe to.**
4. **Call the RESTClient’s GetBulk() function to retrieve the desired Mill NC variable.**To successfully get a Mill NC variable, use the BulkMillNCVariableBox type. RESTClient.GetBulk() returns a BulkWrapper. BulkWrapper contains a bulk that can be typecast to BulkMillNCVariableBox. BulkMillNCVariableBox contains a BulkStruct of type BulkMillNCVariable. This contains an arrayLength, an array of data operandData, an array of data types operandType for each operandData, and an array for setting specific variables during write operations CopyVariables. For GetBulk operations, follow the above code example to read all data received from a given SID.

Note that, when utilizing the GetBulk call, a new BulkMillNCVariableBox object will be returned. This means that whatever is in the receiving bulk object will be completely overwritten by the GetBulk method, **even if the GetBulk request is for only one value**. For this reason, it’s recommended that different objects are used for each get or set operation that you intend to carry out.

**Using SetBulk with REST API**

1. **Follow the same steps 1 – 3 from GetBulk.**
2. **Create a BulkWrapper containing the variables to be set.**The above example shows a BulkWrapper being prepared for setting the SID SID\_PATH\_BULK\_COMMON\_VARS\_100\_TO\_199. The arrayLength value should be set according to the value listed with each SID in the “Available Mill NC Variable SIDs” section. Data can be set as desired in the operandData array. Indices in operandData that are assigned should receive the operand\_type\_enum.DATA\_OPERAND\_TYPE value; the rest should be assigned with operand\_type\_enum.VACANT\_OPERAND. Further, indices that are to be set should also be assigned 1 in the CopyVariables array. Once assigned, follow the code example to wrap the bulk and set the SID using the RESTClient’s SetSID() function.

One thing to keep in mind when utilizing SetBulk for NC variables: the bulk data used for the set call is not automatically cleaned up post-set. Any data placed into the structure will still be present, exactly as it was set previously, when making the next call. If not utilized correctly, this can cause one to set variables that were previously written to again, potentially overwriting data unintentionally. For this reason, it’s recommended that the BulkMillNCVariable method “Reset” is called between writes. This will ensure that only the data that is intentionally written to each variable will be set every time. Reset can be called by accessing the underlying BulkStruct inside the BulkMillNCVariableBox object, as such:

bulk.BulkStruct.Reset();