**Loading Files to the WinMax Control using REST API**

Hurco Companies, Inc.

**Record of Changes**

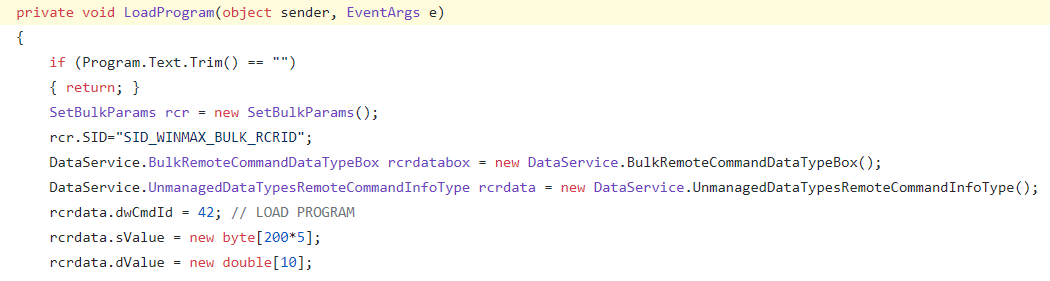
|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Revision Description | Rev By | Date |
| 1 | Original Release | Robert Gorgol | 2017.10.19 |
| 2 | New remote program load argument to allow loading to be skipped when file is already in memory. | Robert Gorgol | 2018.01.08 |
| 3 | Add support for program run that does not automatically queue to repeat upon completion. | Robert Gorgol | 2018.03.09 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Introduction**

This document covers the process for remotely loading part programs and other proprietary files to the WinMax control from a connected application using REST API. The document refers to the software provided in the sample integration package and requires an established connection following the procedure outlined in the document **Connecting to the WinMax Control using a VendorID and REST API.docx**. This connection is only available on Mill machines running v9 or higher control software and lathes running v11.01.311.54 or higher.

**Remote Command Request API**

The remote file loading process uses a generic remote command request API that sends a command index to the WinMax control (42 for the load command) with an array of 5 string arguments and 10 double arguments.



**Loading All File Types**

- 1st string argument (rcrData.sValue[0]) – Full file path and name in the form of an ASCII string that is fewer than 200 characters. The other string arguments are unused.



* All other string arguments are reserved for potential future use.

**Loading Part Programs**

* The file can have any extension that is not reserved for another known file type used by the WinMax system. If it a known conversational program extension (.HWM), WinMax will attempt to load the file as a conversational program. Otherwise, it will attempt to load the file as an NC program.
* 1st double argument (rcrdata.dValue[0]) – Close all other loaded programs before loading the specified program.

0 = no

1 = yes

If WinMax identifies any loaded program as ‘in-use’ in a state where it cannot be safely closed, the remote request as a whole will be rejected. None of the loaded programs will be closed, and the requested new program will not be loaded in this case.

* 2nd double argument (rcrdata.dValue[1]) – Automatically set the program up to run after loading, requiring only a Start Cycle command to initiate the cycle. The screen will automatically switch to the auto run DRO screen. This is limited to running the full program in normal program run mode. Partial program runs and special modes like Dry Run and Speed/Feed Capture are not supported.

0 = Do not queue program to run.

1 = Queue to run and automatically reload for next run upon completion.

(Same behavior as a normal program run initiated manually.)

2 = Queue to run once. Do not automatically reload upon completion.

Minimum supported WinMax versions for a value of ‘2’: 09.01.359.44, 10.01.230.64. Prior versions only support 0 and 1.

‘1’ is recommended for turn-key user interfaces, specialized loading utilities (ex: barcode scanner applications), and applications where a human operator starts each program run cycle, the active program is typically run multiple times in a row, and the utility is only used to simplify switching to a different job. **Not recommended for automation cells.**

‘2’ is recommended for automation cells where a remote client manages distribution of jobs and has full control over which programs to load and run. This optimizes the transition between programs because there is no need for the automation application to wait for the pending repeat run to be aborted and reset to a ready state before commanding the new program to run.

* 3rd double argument (rcrdata.dValue[2]) - Allow program loading step to be skipped if a program with the same full file path and name is already loaded in memory.

Minimum supported WinMax versions: 09.01.359.37, 10.01.230.53, 11.01.311.54

0 = Force file to be reloaded even if the same file is already loaded.

1 = Skip file load and retain current version of file if already loaded.

This argument can be combined with other arguments. For instance, the combination of arguments 2 and 3 can remotely queue a program that is already loaded as the next program to run without requiring it to be reloaded.

* All other arguments are reserved for potential future use. Default values of 0 are recommended to avoid unexpected changes in behavior in the event of future WinMax software upgrades.

**Common Scenarios / Recommended Argument Combinations**

1. Automation cells that repeatedly run a single program or consistent sequence of programs that can all be loaded in memory at the same time.

rcrdata.dValue[0] = 0 // Do not close other programs.

rcrdata.dValue[1] = 2 // Run without automatically queuing to run again upon completion.

rcrdata.dValue[2] = 1 // Skip reloading if already in memory.

This allows the fastest possible transition between runs by minimizing file reloading time and waiting for an additional abort of the pending cycle that is set to run again by the automatic reload.

1. Automation cells that frequently update programs offline and must reload the program to replace the old one in memory.

rcrdata.dValue[0] = 0 or 1 // Close other programs as needed.

rcrdata.dValue[1] = 2 // Run without automatically queuing to run again upon completion.

rcrdata.dValue[2] = 0 // Always load the file. Close and replace if already loaded.

This guarantees that the newest available version of the file is used every time the program is run.

1. Automation cells using large programs that take a significant amount of time to load.

Optimize with multiple calls triggered at different times.

* The initial call will load the first program and queue it to run.

rcrdata.dValue[0] = 0 or 1 // Close other programs as needed.

rcrdata.dValue[1] = 2 // Run without automatically queuing to run again upon completion.

rcrdata.dValue[2] = 0 or 1 // Force reload or use the program in memory as needed.

* At some point during the program run, send a second request to load the next program, but not to run it. This could be triggered in a variety of ways, depending on whether the cell can load the program as soon as it is available (ex: if it is being generated or updated offline in parallel with the program run) or whether jobs are being distributed based on machine availability and the program should only go to a machine when it is known to be almost done with the current job.

rcrdata.dValue[0] = 0 // Do not close other programs. This would cause the load request to fail because the file that is running is still in use and cannot be closed.

rcrdata.dValue[1] = 0 // Do not set the program to run after loading.

rcrdata.dValue[2] = 0 or 1 // Force reload or use the program in memory as needed. Note that if set to 1 and the program is already in memory, this call won’t do anything.

Note that while loading the file at this time will not interfere with the running program, loading a program will cause the screen to switch from the Auto Run screen to the Program Manager screen, where it will remain until the next run is triggered.

* When the client receives notification that the program has completed, send a third command to run the program that was loaded by the second command.

rcrdata.dValue[0] = 0 or 1 // Close other programs as needed.

rcrdata.dValue[1] = 2 // Run without automatically queuing to run again upon completion.

rcrdata.dValue[2] = 1 // This should always skip the reloading step, assuming command #2 successfully loaded the program.

* Repeat the 2nd and 3rd commands in sequence as needed.

If the setup needs to provide a newer version of the same program, this sequence can still be used as long as the newer version of the program is given a different name than the one that is currently running. Attempting to reload a program while it is running will fail because the file will be in use.

1. Jobs that require multiple program files. (Conversational programs that call NC programs using NC merge or NC programs that call subprograms from other files.)

* Load the supporting NC files first using a single command for each required file. These commands can be issued back-to-back without waiting for the previous load request to complete. WinMax will accept and queue multiple file load requests as long as it does not receive a request to load and run a program while another load-and-run request is already in progress.

rcrdata.dValue[0] = 0 or 1 on first load, 0 on others. // only close other programs on the first load.

rcrdata.dValue[1] = 0 // Do not run.

rcrdata.dValue[2] = 0 or 1 // Force reload or reuse program in memory as needed.

* Load and run the main program file.

rcrdata.dValue[0] = 0 // Do not close other programs.

rcrdata.dValue[1] = 2 // Run without automatically queuing to run again upon completion.

rcrdata.dValue[2] = 0 or 1 // Force reload or use the program in memory as needed.

**Loading Tool Library Backup Files**

* Minimum supported WinMax versions: 09.01.359.25 , 10.01.230.36 (unsupported on lathe)
* The file must have a .TCM extension in order to be loaded as a tool library backup file. Otherwise, the system will attempt to load it as a part program.
* 1st double argument (rcrdata.dValue[0]) - Specifies how to combine the tool list from the backup file with the current tools in the library.

0 = Merge/Replace

If a tool in the backup file has the same number as a tool already in the library, the tool from the file replaces the tool in the library. If any loaded conversational programs were using the tool that was replaced, it will be retained in the system as an unmatched tool. The user can then match it to another tool in the library or add it as a new tool, in which case it will be given the next available tool number.

1 = Append

If a tool in the backup file has the same number as a tool already in the library, the tool in the library remains untouched, and the tool from the backup file is given the next available tool number.

* All other arguments are reserved for potential future use. Default values of 0 are recommended to avoid unexpected changes in behavior in the event of future WinMax software upgrades.

**Loading NC State Backup Files**

* Minimum supported WinMax versions: 09.01.371.00, 10.01.247.00, 11.01.311.54
* The file must have a .NCSX extension in order to be loaded as an NC state backup file. Otherwise, the system will attempt to load it as a part program.
* 1st double argument (rcrdata.dValue[0]) - Specifies the components of the NC state file that can be individually loaded. An NC state file contains part setup / work offsets, tool setup, tool offsets, program parameters (shared by conversational and NC), NC-specific parameters, NC program variables, and NC user preferences.

0 = Load all components

Nonzero values are interpreted as a bitmask, allowing restoration of a combination of individual sections.

0x0001 – Part Setup / Work Offsets

0x0002 – Tool Setup

0x0004 – Tool Offsets

0x0008 – Program Parameters

0x0010 – NC Parameters

0x0020 – NC Program Variables

0x0040 – NC User Preferences

Ex: rcrdata.dValue[0] = 0x23; // import part setup, tool setup, and NC program variables

* 2nd double argument (rcrdata.dValue[1]) – Specifies the target of the NC state import.

0 = Replace the global NC state shared by all NC programs.

1 = Import the NC state into the active editing part program.

* + - If the active program is NC, the global NC state is updated.
    - If the active program is conversational, the NC state components that are relevant for a conversational program are imported into that program only. (Part Setup, Tool Setup, and Program Parameters are imported. The other sections are ignored in this case.)
* All other arguments are reserved for potential future use. Default values of 0 are recommended to avoid unexpected changes in behavior in the event of future WinMax software upgrades.