

# MPExec

*Multi-Processing Software Providing Parallel Processing  
of WPS Programs and Applications*

***MPExec Users Guide***

June 2009

---

**MineQuest, LLC**  
1939 Queensbridge Drive  
Columbus, OH 43235-2018  
(614) 457-3714  
[www.MineQuest.com](http://www.MineQuest.com)

## MPExec

MPExec allows you, the WPS Developer, to implement a method to run WPS applications and tasks concurrently. MPExec relies on the strength of the underlying SMP (symmetric multiprocessing) hardware to process more than one task at a time. By utilizing multiple cores or Logical CPU's on multi-core processors, MPExec can dramatically reduce the amount of processing time by running independent WPS code segments concurrently.

With MPExec, you can now exploit a local desktop or server's SMP capabilities by creating self contained tasks that allow for the parallel processing of your WPS code and the automatic coordination of the concurrent sessions log and listing files back into the parent session. You also have access to all the temporary data sets that have been created in each parallel thread by the parent session.

## Benefits

- Achieve maximum scalability for your WPS applications.
- Reduce costs by utilizing your desktop or server's SMP hardware to gain maximum processing efficiency.
- Speed up WPS application and processing performance and lessen the time it takes to process, prepare, transform and report on your data.
- Run multiple WPS tasks concurrently and automatically retrieve the results back into your parent WPS session.
- Both sequential and parallel tasks are supported allowing you to choose which type of processing is best for your applications performance.
- Supports macro variable resolution from parent to threads.
- Automatically matches LineSize and PageSize parameters across threads.

## Statements

MPExec introduces four new statements that allow you to run concurrent processes using WPS.

**%MPEXEC** - which initializes the threading library.

**%Thread()** - is the point where the code you want to run in a thread begins. Optionally, you can provide a Job Identifier with the %Thread() statement. The Job Identifier is used in the WPS log to help you identify which code segments belong to a specific thread. If you do not use a Job Identifier in the %Thread statement, a generic one is created for you.

**%EndThread** - is the point where the code ends for the particular thread.

**%WaitForThreads** - waits for all the concurrent processing to complete and assembles the log and list files for each concurrent process into the parent's log and list file.

## Programming MPExec

For the WPS developer who wants to utilize MPExec, it's important to understand the design considerations and how MPExec is intended to be implemented. MPExec allows for concurrent (also called *Threading* or *Parallel Processing*) segments of WPS code. Identifying which code segments are good candidates for threading is important to its success.

MPExec is designed so that WPS code segments that are placed between %Thread() and %EndThread keywords must be independent of other threads. Functionally, the restriction is that you cannot run multiple threads that:

- create a data set of the same name (in either a permanent or temporary library).
- does not use a permanent or temporary dataset that was created in one %Thread block while simultaneously trying to create the data set of the same name in another %Thread block that is running concurrently.

MPExec threads are independent of each other. Threads do not know what might be contained or created in other concurrent threads. It is up to you as the programmer to make sure that you do not step on, delete or overwrite other files created across thread blocks.

Ultimately, the success of threading your code is the reduction of time it takes to run your WPS programs. This is dependent on the SMP hardware you are running WPS on and how it is configured. The greatest impediment to dramatically reducing your run times will be the I/O system of your hardware. Running multiple drives in a RAID-0 array for your work files will often be the major factor in successfully reducing your execution times.

Threading is also dependent on the number of cores or Logical CPU's that your hardware has in place. Creating more than two threads per Logical CPU often will not show much improvement because your programs tend to become CPU bound. As the developer, you will probably have to run a series of test programs that contain multiple threads to find the optimum number for best performance.

The structure of an MPExec enhanced WPS program is best illustrated with some pseudo code. Note the ';;;' just before each %EndThread statement. This is required.

```
%MPExec;          /* Initialise the threading library */

    %Thread(Job A); /* Start a thread with a thread id of "Job A" */
        ...
        Your WPS Code for Job A here
        ';;;'
    %EndThread; /* End the thread for Job A */

    %Thread(Job B); /* Start a thread with a thread id of "Job B" */
        ...
        Your WPS code for Job B here
        ';;;'
    %EndThread; /* End the thread for Job B */

%WaitForThreads; /* wait for all the threads to complete before further processing */
```

Below is an example of running two threads, creating two different temporary data sets, with each creating a PROC MEANS, a PROC FREQ and a PROC SORT to sort the data. After the threads have completed, the program merges the two data sets created in each thread together.

### Example 1. Running two concurrent tasks.

```
%let Iter = 2500000;

%MPExec;

  %Thread(Job A);

    data a;
      do ii=1 to &iter;
        a=ranuni(0); b=ranuni(0); c=ranuni(0); d=ranuni(0);
        e=ranuni(0); f=ranuni(0); g=ranuni(0); aa=round(c*10,1);
        output;
      end;
    run;

    proc means data=a; run;

    Proc sort data=a; by ii; run;

    Proc freq data=a;
      tables aa;
      Title 'Proc Freq Run for Data Set A';
    run;

  ;;;
  %EndThread;

  %Thread(Job B);

    data b;
      do ii=1 to &iter;
        j=ranuni(0); k=ranuni(0); l=ranuni(0); m=ranuni(0);
        n=ranuni(0); o=ranuni(0); p=ranuni(0); ab=round(j*10,1);
        output;
      end;
    run;

    Proc Means data=b; run;

    Proc Freq data=b;
      tables ab;
      Title 'Proc Freq Run for Data Set B';
    run;

    Proc Sort data=b; by ii; run;

  ;;;
  %EndThread;

%WaitForThreads;

*--> Once threads A and B have completed, merge the two data sets together;
Data all;
  merge a(in=in1) b(in=in2);
  by ii;
run;
```

## WPS Log for Example 1.

Below is the last part of the log that was generated by MPEExec. MPEExec provides a summary as well as some details on the concurrent processing that took place.

```
1481 %MPEExec;
NOTE: MPEExec threading modules initialised.
NOTE: Multi-threaded processing commencing at: 11:04:02.67

      ...      ...      ...
      log statements deleted for brevity
      ...      ...      ...

NOTE: Parallel Threading Completed. 2 Parallel Threads Were Executed.
NOTE: Thread Summary:
      Start Time: 11:04:02.67 End Time: 11:04:15.88 Elapsed Time: 0:00:13.214 (hh:mm:ss:hs)

NOTE: Parallel Task Execution Detail (in order of submission)

      Thread 1 cpu time : 00:00:12.339 real time : 00:00:12.092 Task ID: Job A
      Thread 2 cpu time : 00:00:11.731 real time : 00:00:13.015 Task ID: Job B

      All thread totals : 0:00:24.070 0:00:25.107

NOTE: Calculated minimum threading efficiency: 1.9x (0:00:25.107 / 0:00:13.214)

NOTE: MPEEXEC copyright © 2009 by MineQuest, LLC. and is licensed software.
NOTE: MPEEXEC v1.0 All rights reserved.

1553 *--> Once threads A and B have completed, merge the two data sets together;
1554 Data all;
1555 merge a(in=in1)
1556 b(in=in2);
1557 by ii;
1558 run;

NOTE: 2500000 observations were read from "WORK.a"
NOTE: 2500000 observations were read from "WORK.b"
NOTE: Data set "WORK.all" has 2500000 observation(s) and 21 variable(s)
NOTE: The data step took :
      real time : 00:00:06.317
      cpu time : 00:00:06.442
```

MPEExec provides the time that the concurrent processing started, when it ended, and the elapsed time. It also provides information on each thread with the amount of CPU time and REAL TIME that each thread took to execute. Finally, an efficiency statistic is provided that compares the elapsed time with the total real time of all threads so you have some basis to evaluate how well your parallel processing is performing.

## MPEExec Restrictions

- MPEExec does not execute threads across different network machines.
- There is a limitation of a maximum of 256 threads.
- Macro variables inside comment blocks `/* &mymacrovar */` are intentionally not resolved.
- Four semi-colons (`;;;;`) are required before the `%EndThread` statement.

## Installation

Installing MPExec requires that you place the macrolib (SASMACR.WPCCAT) into a folder on your machines hard drive. If you unpack the MPEXEC.ZIP file and select the default directory for uncompressing, it will create a folder called MPExec. You will need to add the following lines of code to your autoexec.sas file. If you don't have an autoexec.sas file, you need to create one and then add the following three statements to it.

```
libname mpexec 'c:\MPExec'; *--> the location of the MPEXEC macrolib;
options mstored sasmstore=mpexec ;
%let _WPSLoc = c:\Program Files\World Programming WPS 2;
```

Modify the first line the statements above to point to the folder where you copied the MPExec macrolib. You also need to modify the last line to point to the folder where the program WPSI.EXE resides. If you are using Vista 64, be sure to check to see where WPS was installed. You may have to modify the last line to read:

```
%let _WPSLoc = c:\Program Files (x86)\World Programming WPS 2;
```

## **Copyright Notice**

This MPEXec Users Guide is copyrighted © 2009 by MineQuest, LLC. Columbus, OH 43235. All rights reserved.

The macro programs, %MPEXEC, %Thread, %EndThread and %WaitForThreads are copyrighted © 2009 by MineQuest, LLC. Columbus, OH 43235. All rights reserved.

## **Warranty**

This software and manual are offered “AS IS” and without warranties as to performance or merchantability that seller’s and/or redistributors may have made statements about this software. Any such statements do not constitute warranties and shall not be relied on by the user in deciding whether to use this program.

This program is offered without any express or implied warranties whatsoever, because of the diversity of conditions and hardware under which this program may be used, no warranty of fitness for a particular purpose is offered. The users of this software are advised to test the program thoroughly before relying on it. The users must assume the entire risk of using the program; any liability of seller, provider or manufacturer will be limited exclusively to product replacement.

In no event shall MineQuest be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential or other damages in the use, installation and application of MPEXec.

## **Governing Law**

This statement shall be construed, interpreted, and governed by the laws of the state of Ohio, United States of America.

## **Acknowledgements**

MineQuest, LLC is a registered corporation that specializes in WPS and SAS Software development and consulting. Web: [www.minequest.com](http://www.minequest.com)

World Programming Company. World Programming Company develops and sells WPS, a SAS/Base alternative that runs on Windows and z/OS. Web: [www.teamwpc.co.uk/home](http://www.teamwpc.co.uk/home)

## **Trademark Notices**

WPS and ORA are the trademarks of World Programming Limited.

## MineQuest Commercial Software License

Please read this license carefully before using the software. By using the software, you are agreeing to be bound by the terms of this license.

### END USER LICENSE AGREEMENT

This software, MPExec is protected by both United States law and international treaty provisions. MineQuest retains all rights to this software including intellectual property rights and distribution rights. MPExec is commercial software and requires an annual purchased license for each desktop or server that this software is to be used on.

MineQuest provides the computer software, documentation, and source code in this product ('Licensed Software') and licenses its use to the party ("You") who purchases and uses this product, as follows:

**BINARY LICENSE.** You have the non-transferable right to use one (1) copy, on one (1) single computer system or network, with the number of seats specified on the original invoice, of Licensed Software internally for Your own research and development or other business purposes. You must purchase a license for each User that will be using the Licensed Software. Licenses are non-transferable.

You may make one machine readable copy of the Licensed Software for backup purposes. You may not otherwise copy the Licensed Software or any of its documentation in printed form. You must, as a condition of Your license under this Agreement, reproduce and include all copyright and other proprietary notices on the backup copy.

You may not transfer possession of Licensed Software, whether in part or in whole, to other parties. You may not copy enclosed documentation or distribute them to other parties. You may not modify Licensed Software or disassemble it for the purpose of obtaining the source code. You agree not to "reverse engineer" the Licensed Software or any portion of it or to permit such reverse engineering or otherwise reduce the Software to a human-perceivable form.

**SOURCE CODE.** At MineQuest's sole discretion, a portion of the this software may be provided in source code form, constituting proprietary information of MineQuest. If provided, You may use these sources to customize the software for **Your** particular needs, or simply as supplementary documentation, but You may not distribute the sources in either their original or modified form.

**RESTRICTIONS ON DISCLOSURE.** You agree to use reasonable efforts, and to take the same precautions that You take to protect your own computer programs and other information of similar value that You do not wish to have disclosed or disseminated to others, to avoid disclosure or dissemination of any part of the Licensed Software, its documentation, or its source, to any third party by You and Your employees, agents or consultants (if any). These restrictions will not apply to any information that becomes publicly known, through no fault or breach on Your part, that MineQuest provides third parties without restrictions on disclosure or that you receive from a third party rightfully and without restriction on disclosure.

**OWNERSHIP.** You agree that MineQuest has and will retain all ownership rights in the Licensed Software and its documentation and all related patents, copyrights, trademarks, service marks, and other

proprietary rights, and that You will have no interest in the Licensed Software and its documentation except the limited license provided under this Agreement.

**NO OTHER LICENSE.** No rights or licenses are granted by MineQuest under this License, expressly or by implication, with respect to any proprietary information or patent, copyright, trade secret, or other intellectual property right owned or controlled by MineQuest, except as expressly provided in this License.

**TERMINATION.** This License is effective until for one year from date of purchase. You may terminate this License at any time by destroying the Software, related documentation and all copies thereof. This License will terminate immediately without notice from MineQuest if You fail to comply with any provision of this License or fail to renew the License. Upon termination You must destroy the Software, related documentation and all copies thereof which are in your possession or under Your control.