JetTrac JobController Guide

Purpose:

The information found in this document will outline the functionality found within JetTrac JobController (aka JobController and JBC) as well as tips and tricks for troubleshooting some of the common issues.

How JobController Runs:

JobController is an executable program that typically runs as a service on a Windows server (or Windows desktop OS), however, can also be run through a batch file (.bat). While running. JobController acts as a shell to process jobs specified within the JMD (Job Management Database). *Please see documentation on JetTrac JobConfig for information on setting up Jobs.*

As JobController is running, it will scan a watched folder every specified number of seconds for a new file to appear (this is by default every 5 seconds, however, both the number of seconds and which folder should be watched are configurable in the jfserver.ini file). Once one or more new files are found within the watched folder, JobController will make an internal list of the files in order of their date modified and then process them in the order in which they are in the list. Once all of the files in the list are processed, JobController begins to scan the watched folder again for any new files.

JobController and Job Names:

When a file found in the watched folder is being processed, JobController first looks to see what file type it is and then looks into the jfserver.ini configuration file to find out what job to run to process that extension. A file type can be set to always process the same job, or it can be set to look within the file to find a Job Card which contains the job name to be run for that file. The following are the file types that can have a Job Card:

- .xml - When JobController is set up to read the job card from an XML file, it looks in the first line of the XML for this:

  *<?xml version="1.0" encoding="UTF-8"?><?JtJob **JobNameHere** ?>*

  The first half of the line is standard XML encoding, but you will notice that after JtJob in the second half of the line is the place which specifies the job name for JobController to

run. **Note: there cannot be a CRLF before the JtJob portion - it needs to be on the first record.**

- .pdf - This file type will almost always be set to run the same job, PDFJOB. This is a standard job within the JMD that allows JobController to look within the PDF field data to find the job name. The job name within a PDF should always be in a field named Job_Name. The field can be hidden so it is not visible, but it has to exist with that name and a text value of the job name to process the PDF file.

- .dat - The job card data within this file type,which tells JobController which job to run, is found on the first line of the .dat file and looks like this:

  *^job **JobNameHere***

  The job name found in this line will be the job that processes this input .dat file.

Archiving Processed Files:

If a file is run through a job without any errors, the file is placed into the archive folder which is also configurable within the jfserver.ini configuration file. If a file being copied to the archive folder has the same name as another file within the archive folder already, the new file will be renamed with a date/time stamp suffix in order to not overwrite the previous file and avoid collision between them.

Error Handling:

When an error occurs while processing a file through JobController the file is copied to the error folder, specified within the jfserver.ini configuration file similar to the archive folder. Along with this file a .err stub file is also copied to the error folder. This stub file contains the portion of the JobController log with the actual error message. If a file exists already in the error folder with the same name, a suffix will be added to the end of the new file name to differentiate between the two.

Along with copying the file to the error folder, you are also able to specify a job name within the jfserver.ini configuration file to run. This job is typically used to email a notification of the error to whoever needs to be notified.

One thing of note, is that when a PDF file is picked up by JobController, typically PDFJOB will run to extract the job name from the PDF, and then that extracted file will be run through the job specified. This makes it so that if a PDF file were to error in JobController, the PDF itself will

actually be in the archive folder. Within the error folder you will only find the .err stub file as well as a stub file containing the location of the PDF.

Restarting or Reloading JobController:

When JobController first starts on a server, both the JMD with the job tasks as well as the jfserver.ini configuration file are loaded. If any changes are made to either of these files JobController needs to either be restarted or reloaded in order for the changes to take effect.

To reload the JMD and jfserver.ini configuration files, you can simply drop a file with the extension of .msg into the watched folder. This reloads both files without fully stopping the JobController service.

To restart JobController, we provide a .bat file which stops the service and any running java.exe processes, and then starts the service again. It is important when restarting the service to stop the instances of java.exe as JobContoller uses this process and starts it when JobController starts. If you do not stop the java.exe process, you run the risk of having duplicate processes running and thus duplicate files being processed.

Sometimes when a person is testing JetTrac they may prefer to start JobController from a bat file. When this is done, a CMD (Command) Window appears that shows the processing happening within JobController. To restart JobController when you are running it through a .bat file and not a service, simply close the CMD prompt window and rerun the StartServer.bat file.

Editing the jfserver.ini Configuration File:

The jfserver.ini file is split into four sections: Required, UserVars, GlobalVars, and TimedJobs. The following will go through each section and explain how to adjust them. Here is a sample of the configuration file:

[Required]
PauseTime=5
FileSpec=.pdf:PDFJOB|*.dat:*|*.htm:HTMLJOB|*.html:HTMLJOB|*.xml:*|*.txt:NOJOB|*.lic:JTLicenseUpdate*
DataPath=C:\JetTrac\JobController\Server\Data
BackupPath=C:\JetTrac\JobController\Server\Archive
ErrorPath=C:\JetTrac\JobController\Server\Error
LogFile=C:\JetTrac\JobController\Server\Log\JobController.log
BinPath=C:\JetTrac\JobController\Bin
JMDFileName=C:\JetTrac\JobController\Server\Config\jfserver.jmd
JMDFileName2=C:\JetTrac\JobController\Server\Config\jfserver2.jmd

JasperPath=C:\JetTrac\JobController\Jasper
ErrorJobName=JBCError
UseLicenseLog=C:\JetTrac\JobController\Server\Log\JTTransactionHistory.log
PrefixJobs=GDU_:JTGoogleDriveUpload

[UserVars]
LicenceFile=C:\JetTrac\JobController\License\jettrac.lic
FormPath=C:\JetTrac\JobController\Server\Forms
PdfPath=C:\JetTrac\JobController\Server\PDF
JasperPath=C:\JetTrac\JobController\Jasper
CmdExe=C:\Windows\SysWOW64\CMD.EXE

[GlobalVars]
 # WARNING: THE JETTRAC BYOD INSTALLER MSI USES THE FOLLOWING LINES IN THE INSTALLATION PROCESS. DO NOT MODIFY THESE LINES.
 # NOTE: At signs "@" must have a tilde "~" in front of it since @ is a reserved character
#JetTracCompanyName=
#CompanyBusinessContactNameEmail= <jettracsandbox~@gmail.com>
#CompanyTechnicalContactNameEmail= <support~@protechinc.com>
#JTErrorNameEmail=ProTechnology Support <support~@protechinc.com>
#VendorSupportNameEmail=support@YourVendorHere.com <Your Vendor Here>
#ProTechnologySupportNameEmail=ProTechnology Support <support~@protechinc.com>
#===
==================================
JetTracCompanyName=ProTechnology
CompanyBusinessContactNameEmail= <jettracsandbox~@gmail.com>
CompanyTechnicalContactNameEmail= <support~@protechinc.com>
JTErrorNameEmail=ProTechnology Support <support~@protechinc.com>
VendorSupportNameEmail=support@YourVendorHere.com <Your Vendor Here>
ProTechnologySupportNameEmail=ProTechnology Support <support~@protechinc.com>
 # END JETTRAC BYOD INSTALLER RESERVED LINES
ProTechnologyTicketNumber=(PTT#7695)
ContractStartDate=6/1/2017
ContractEndDate=5/31/2018
ContractVolume=20000
#===
==================================

[TimedJobs]
RunMonthly=1-JTLogArchiveJob
RunMonthly=1-JTTransactionSummaryJob
RunDaily=23-JTDailyLicenseCheck
RunWeekly=3-JTUpdateJetTracBYOD

- Required - This section contains the required key values for JobController to run appropriately. Each of these key values must exist within the file and have valid values.
    - PauseTime - This value should be a number. The number specifies how many seconds JobController will wait between each scan of the watched data folder. The default is 5.
    - FileSpec - This is a pipe-delimited list of extensions and which job to process for each file type. There can be any number of file types listed here. The syntax is

      .pdf**:**PDFJOB***

      After the asterisk and period, you specify the file type, then a colon, and the job to run. If it is a .dat or .xml and you wish to have JobController pull the job from the job card within the file, simply use as asterisk in place of the job name as such:

      .xml:

    - DataPath - This is the fully qualified file path to the watched data folder. This folder is where JobController watches for files.
    - BackupPath - This is the fully qualified file path to the archive folder.
    - ErrorPath - This is the fully qualified file path to the error folder.
    - LogFile - This is the fully qualified file path to the output log file.
    - BinPath - This is the fully qualified file path to the bin folder.
    - JMDFileName - This is the fully qualified file path to the primary job management database.
    - JMDFileName2 - This is OPTIONAL and does not need to exist within the configuration file. This is the fully qualified file path to the secondary job management database. When JobController starts, if this field is present with a file path, not only will the primary JMD be loaded but the job steps from this secondary JMD will also be loaded. The purpose of this is to allow for a standard primary JMD for clients while allowing you to put custom jobs in a separate file. This allows for better version control of the primary JMD. Please note that while the job steps are copied over from the secondary JMD, the task lines for each module are ignored. This means that in order to use a module in a job in the secondary JMD, the task line for that module must exist in the primary.
    - JasperPath - This is an old variable that is not used, however, it needs to be within the jfserver.ini configuration file with a value. We recommend not editing it.
    - ErrorJobName - This is the name of the job within the JMD that will be run when an error occurs in JobController.
    - UseLicenseLog - This is the fully qualified file path to the transaction log file. This file is used along with the JTTransactionSummary module to allow you to keep

track of transactions processed in a single instance of JobController. Please refer to the module guide for JTTransactionSummary for more information.
    - PrefixJobs - This line is pipe-delimited and contains two parts per section separated by a colon. The first part is a prefix and the second part is a job name. Whenever JobController scans the watched data folder and finds a file with the specified prefix at the beginning of the file name, it will automatically run the job specified after the colon. The purpose of this is if there is a file type that will only ever process a single job. Instead of using a job card to specify the job name within the file, JobController will automatically run the job specified within the jfserver.ini whenever a file of that type is found within the watched folder.

- UserVars - This section will rarely be edited and simply sets the locations for files required by JobController.
    - LicenseFile - This is the fully qualified file path to the license file.
    - FormPath - This is an old variable that is not used, however, it needs to be within the jfserver.ini configuration file with a value. We recommend not editing it.
    - PdfPath - This is an old variable that is not used, however, it needs to be within the jfserver.ini configuration file with a value. We recommend not editing it.
    - JasperPath - This is an old variable that is not used, however, it needs to be within the jfserver.ini configuration file with a value. We recommend not editing it.
    - CmdExe - This is the fully qualified file path to the command prompt executable.

- GlobalVars - Within this section you can specify variables that will globally apply and can be used during job steps. These types of variables will typically be company information, email addresses for error notifications, etc. These variables should not be used for anything that will change on a job by job basis.
    - Any of the lines commented out (they begin with a #) should not be edited. These lines contain notes, visual separations, and data required by our JetTrac installer.
    - The first six lines that are not commented out in this section should be filled in with the company name for who is running this install as well as valid email addresses for the rest of the lines. These are used by some of our default jobs, specifically the JBCError job. If you use the JetTrac installer, these lines will be set up through that GUI. Please note that within these lines wherever there is an @ symbol, it must be preceded by a tilde. This is to prevent the configuration file from treating everything after the @ as a variable.
    - The last four lines can be modified to be any variable you wish. In the example, we have more company data. Any of these lines, as well as the six above can be called in the command line of a module as variables.

- TimedJobs - This section allows you to specify any number of job names that should be run at either a monthly, weekly, daily or “RunEvery" interval. Please note that only one job can run at a time this way. For example, on daily jobs you can’t have multiple jobs set for 11:00am. On weekly, you cannot have multiple jobs run on Tuesday, and on monthly

you cannot have multiple jobs run on the 15th. Also note that you can only have one line per Run command. In order to specify more than one of a single run command, RunDaily for example, you would need to put it in the same line separated by a comma. For example, “RunDaily=1-Job_Name1,13-Job_Name2”.

- RunMonthly - In the example provided above we have two monthly jobs. The number after the equal sign is the day of the month to run on. As these are set to 1, every month on the 1st at 12:00am the job specified after the dash will run.
- RunWeekly - This is set up the same way as RunMonthly, except that the number refers to the day of the week in which the job will run. The numbers can go from 1 - 7, 1 being Sunday at 12:00am , and 7 being Saturday at 12:00am.
- RunDaily works similar to the other two, with the exception of the number referring to which hour of the day to run the job. In this example, the job will run at 11:00pm every night.
- RunEvery - runs a job every n seconds, e.g.
  RunEvery=3600-JobName1,120-JobName2
  Runs JobName1 every 1 hour (60 seconds x 60 minutes)
  Runs JobName2 every 2 minutes (60 seconds x 2 minutes)

Built in Variables:

The following is a list of variables that do not need to be set prior to using them, they are always available within the JobController service:

- DataFile - the input data file name with the fully qualified file path.
- DataName - the input data file name without the fully qualified file path.
- DataPath - the input data fully qualified file path without the file name.
- JobTimeYear - The date year when the job started processing (2018)
- JobTimeMonth - The date month when the job started processing (09)
- JobTimeDay - The date day when the job started processing (29)
- JobTimeHour - The hour when the job started processing (23)
- JobTimeMin - The minute when the job started processing (59)
- JobTimeSec - The second when the job started processing (09)

Setting Variables:

Variables are able to be set in JobController from a data file. These variables can then be used to pull dynamic files, name files dynamically, etc. While a part of JobController, the functionality that allows this is called SetVariables and is treated as a separate module within jobs. As such, please refer to our SetVariables module guide in order to learn how to set variables within a JobController job.

Setting Up Jobs and the JMD:

When JobController processes a file, it runs them through a set of steps in a specified job. These jobs are set up in a file called the JMD or Job Management Database. This file is typically found at *C:\JetTrac\JobController\Server\Config\jfserver.jmd.*

It is highly recommended that you set up, edit, and manage these jobs through our JobConfig GUI. To find more information on this, please refer to our training guide on JobConfig.

You can, however, modify directly through the jfserver.jmd file itself if absolutely needed. The only time this should be necessary is if a bad !x task line is added from the JobConfig GUI. This situation occurs if you have an invalid set up for a module in the jobconfig.ini file.

(Please refer to the JobConfig GUI - Adding/Adjusting Modules training document for more information on this.)

When you save changes in the JobConfig GUI, JobConfig checks to see if there are any new modules being used in any of the jobs that do not have task lines in the jfserver.jmd. If there is a new module, the task line will be added from the jobconfig.ini file. If you have a invalid set up for a module in the jobconfig.ini file, add that module to a job in the JobConfig GUI, and then save changes, this invalid task line will be added to the JMD. The only way to rectify this, would be to manually remove the task line from the JMD. You can do this by opening the jfserver.jmd in Notepad++ or any other text editor, scrolling to the bottom, then deleting the entire !x line for the module from the JMD. Once you save changes make sure the jobconfig.ini file now has the correct task line for that module, then open JobConfig GUI, remove that module from the job, re-add it, and save changes. This should add the now correct task line to the JMD.

JetTrac JobController Version History:

- 1.0 - Original release
- 1.2 - Handles XML files as well as reads XML Job Cards.
- 1.3 - Removes job timeout.
- 1.31 - If a scan of the active folder has at least one file, then don't wait before doing next scan. Only wait if the previous scan returned 0 files.
- 1.32 - Handles OtherJobTokens in ^JOB jobname and printer switch on job card (-zhplj or -z"hp laser 2033").

- 1.36 - Adds the @DataFile variable and supports XML job cards.
- 2.0 - Allow for multiple types of extensions to be processed with different job handling for each one. Now supports PDF input file type.
- 2.01 - Renames processed files going to the Archive folder if another file of the same name already exists. Prevents collision.
- 2.02 - Added capability to pick a file up on a subsequent scan if the file is locked initially.
- 3.0 - Adds user defined variables through a vars.ini file.
- 4.0 - Transaction log writting. New key value in jfserver.ini for UseLicenseLog. Every transaction run through JobController will be written to the log. If anything in the log is manually changed or removed we will be able to detect it.
- 4.01 - Fix for instances where muliple SetVariables steps are run.
- 4.02 - JBC.VARS error fix.
- 4.21 - Added the ability to specify a job name to run when an error occurs.
- 4.3 - Additional key values can be specified in the jfserver.ini to run scheduled jobs.
- 4.4 - Scheduled jobs can now be specified as Periodic, Daily, Weekly, or Monthly.
- 4.5 - Changes error reporting to also allow for attaching the original error file.
- 4.6 - Additional section added to the jfserver.ini for GlobalVars
- 4.7 - Email addresses in the jfserver.ini were being treated as variables to be substituted due to having an @ in them. This version adds an escape character ~ before a @ so that it is not treated like a variable.
- 4.8 - JBC.VARS will scrub values replacing \ / : * ? " < > | with _
- 4.9 - New optional key value in the jfserver.ini that specifies a prefix to a file name found in the watched folder. If the specified prefix is found, the specified job is run.
- 4.10 - JBC.VARS no longer automatically scrubs all variable values. Instead it will only scrub when a dash is found immediately after the @ when calling the variable.
- 4.11 - This adds the trim functionality to JBC.SETVAR.

- 4.12 - Added the <errDesc> tag to the error XML file when JobController errors.
- 4.13 - Will filter # from variable values if scrubbing is specified for that variable.
- 4.14 - Add variable JobTimeYearYY for 2 digit year.
- 4.15 - Fix for 2 digit year variable.
- 4.16 - Fix variable trim for JBC.SETVAR. Also, add a key value in jfserver.ini for secondary JMD. This JMD will append all !f lines to the end of the !f lines found in the primary JMD. The !x lines are ignored in the secondary JMD.
- 4.17 - No longer runs timed jobs (daily, monthly, weekly, periodic) when JobController restarts.
- 4.18 - Restarting the JobController service will now also reload the jfserver.ini. Also, allows for a .msg file type to be dropped into the watched folder to reload the JMD and jfserver.ini without fully restarting the service.

Troubleshooting Tips and Tricks:

- When using a secondary JMD, it is assumed that the job names within the file will be unique from the job names in the primary JMD. If these job names are not unique, the job steps will not function properly as all steps from both JMDs will run from top to bottom (starting with the primary JMD) as every step with the same job name is treated as being part of a single job.