Load Manager  
Version 5.5  
  
User Guide

Table of Contents

Contents

[Table of Contents 1](#_Toc455008564)

[1 Load Manager Overview 3](#_Toc455008565)

[1.1 Introduction 3](#_Toc455008566)

[1.2 Platform and Development Language 3](#_Toc455008567)

[1.3 What’s New in Version 5.x 3](#_Toc455008568)

[1.4 Features 4](#_Toc455008569)

[1.5 Additional Functionality 5](#_Toc455008570)

[1.6 How Does LoadMgr Log Job Status 6](#_Toc455008571)

[1.7 LoadMgr SAS Log Analyzer 7](#_Toc455008572)

[1.7.1 Development Background 7](#_Toc455008573)

[1.7.2 Log Analyzer Functions 7](#_Toc455008574)

[1.7.3 How does log analyzer work? 7](#_Toc455008575)

[1.7.4 Log message knowledge file structure 8](#_Toc455008576)

[1.7.5 Set up suggested action for known messages 9](#_Toc455008577)

[2 Installation and Setup 10](#_Toc455008578)

[2.1 Installation Instructions 10](#_Toc455008579)

[2.1.1 Download LoadMgr 10](#_Toc455008580)

[2.1.2 Edit LoadMgr Profile 10](#_Toc455008581)

[3 Using LoadMgr 13](#_Toc455008582)

[3.1 Create LoadMgr Configuration File 13](#_Toc455008583)

[3.1.1 Job List Section 14](#_Toc455008584)

[3.1.2 Job Parameter Section 15](#_Toc455008585)

[3.1.3 Local Setting Section 15](#_Toc455008586)

[3.1.4 Log Message Pattern Sections 16](#_Toc455008587)

[3.2 Validating a LoadMgr Configuration File 16](#_Toc455008588)

[3.3 Executing a Process Using LoadMgr 16](#_Toc455008589)

[3.4 Terminating LoadMgr Processes 17](#_Toc455008590)

[3.5 Monitoring LoadMgr Processes 17](#_Toc455008591)

[3.5.1 Check LoadMgr Process Status Log File 17](#_Toc455008592)

[3.5.2 Check LoadMgr Stand Out and Stand Error 17](#_Toc455008593)

[3.6 Forcing a Fresh Restart after a Job Failure 18](#_Toc455008594)

[3.7 Re-run a Failed Job 18](#_Toc455008595)

[3.8 Log File Archival 19](#_Toc455008596)

[3.9 Automate LoadMgr Log Clean-Up Process 19](#_Toc455008597)

[3.10 Running nmon within LoadMgr 19](#_Toc455008598)

[3.11 Running a Backup Utility during a Load 20](#_Toc455008599)

[3.12 Using FULLSTIMER Log Parser 21](#_Toc455008600)

# Load Manager Overview­

## Introduction

Load Manager (***LoadMgr***) is a UNIX/Linux shell script application that automates both parallel and sequential batch SAS and shell jobs.

## Platform and Development Language

***LoadMgr*** was developed originally using Korn shell 88 (ksh 88) on the AIX operating system for SAS applications. Recently, many SAS applications are running on Linux Red Hat Enterprise Linux Server release 5.x, which comes with newer Korn Shell, ***LoadMgr 5.x*** is re-written in Korn Shell 93(ksh 93).

***LoadMgr 4.x***  runs also on Windows Subsystem for Unix-based Applications (SUA) 3.5 Microsoft’s Windows Services for UNIX SUA comes with Windows OS (Windows 2003 Server and XP), and it is officially supported by Microsoft. Since KSH on SUA is a public domain KSH, slightly different from KSH88 running on most UNIX operating systems, the Windows version of ***LoadMgr*** is a separate package. ***LoadMgr 5.5*** is not available on Windows SUA.

## What’s New in Version 5.x

***LoadMgr*** ***5.5*** has significant enhancements as compared to previous releases of ***LoadMgr***. The following changes are included with ***LoadMgr 5.5:***

* + - * It is common that a LoadMgr process needs to finish within a time window to meeting SLA (Service Level Agreement). Therefore, when a job fails, you may want to restart it while the process is still running. In previous releases, you have to wait for the LoadMgr process to stop before resubmitting it. ***LoadMgr 5.5*** allows the Admin to re-run the failed process when it is running.
      * All Configuration files that define a LoadMgr package are now in a single configuration file. The configuration file will have a .conf extension. This replaces the .meta, .param, and .email files in LoadMgr 4.x.
      * You can now use a SAS program more than once in a package. Log file names will include the SAS program name plus the Task ID, which is unique in each package. This way, you can simply define different parameter values for the same programs.
      * You can now use fully qualified SAS program names in a configuration (.conf) file. This removes the requirement of having soft links to program directories, and allow you to have programs from different directories in the same configuration file. You can still use soft links, if you wish. You can override the default source code directory in your package configuration file.
      * You can now use a “testing” area for dry runs of production programs. Us the following syntax to use this feature:

$LOADMGR/bin/controller.sh –d [testing\_dir] [Package\_name]

* + - * The Log Analysis Report now has a separate e-mail distribution list. This will allow remote consultant to receive LoadMgr notification on SMS based devices
      * Warning notification messages can now be disabled. When disabled, you will no longer receive e-mail messages for SAS Warnings.
      * ***LoadMgr 5.5*** archives logs like its predecessors. However, the backup directory is now time-stamped using its last update time. This will help you find the correct log files when troubleshooting
* ***LoadMgr 5.5*** integrates a log parser utility to provide FULLSTIMER statistics for performance analyzing. You need to set FULLSTIMER\_PARSE=1 in .LoadMgr\_profile to enable this function.
* ***LoadMgr 5.5*** will enforce the order of jobs defined in the configuration file.
* ***LoadMgr 5.5*** support extensions by defining multiple function paths in FPATH parameter in .LoadMgr\_profile
* ***LoadMgr 5.5*** stops a package when it runs more than MAX\_RUN\_TIME, which is configurable in .LoadMgr profile.

## Features

***LoadMgr*** manages a list of SAS jobs and UNIX/Linux shell scripts as an overall process, the entire process is known as a package. The following is a list of features supported by ***LoadMgr***:

* Configuration data-driven process automation:
  + ***LoadMgr*** reads a configuration file that lists jobs and their predecessors,
  + ***LoadMgr*** then executes the jobs according their dependencies and defined concurrency level.
* Status and Re-Startability:
  + ***LoadMgr*** logs execution status and run time for each jobs as well as the entire process. If the process failed on particular jobs, other independent jobs will run to their completion. You can overwrite the default restart by issue –f at command line like: *controller.sh -f sample*
  + When an entire process stops running due to a job failure, the application’s administrator will need to address the root cause of the job failure. ***LoadMgr*** will resume the process at the point of failure with the same ***LoadMgr*** command.
* Parameter Driven SAS Application:
  + ***LoadMgr*** supports a job parameter that can define or change job parameters easily.
* SAS log management:
  + ***LoadMgr*** backs-up job historical log files with date/time stamps for each process. The backup process includes SAS logs, list files, nmon output, and html reports.
  + ***LoadMgr*** also provides a configurable utility to clean up historical log files based on their age. This feature is helpful considering that some weekly log file sizes can grow to GBs.
* Email notification:
  + ***LoadMgr*** can send out email notification messages automatically whenever a job fails
  + ***LoadMgr*** can also send notification when an entire package of jobs finishes successfully.
* SAS Log Analysis:
  + ***LoadMgr*** comes with a SAS log analyzer utility.
* Enterprise Scheduling:
  + ***LoadMgr*** can be easily integrated with job schedulers such as CRON, ESP, and AutoSys.
* Performance Monitor:
  + ***LoadMgr*** is integrated with NMON and VMstat on Unix/Linux
  + It can automatically start and stop NMON/VMstat to provide runtime performance data collection.

## Additional Functionality

* + - * To prevent the predecessor job from becoming invalid during the update process, the checkconf.sh utility now validates the predecessors to ensure a configuration file is valid.
      * The controller.sh script accepts the –f option to force a fresh restart.
      * Version 2.4 and above comes with a robust full and incremental SAS datamart backup utility using rsync.
      * Starting from 4.0, multiple instances of the ***LoadMgr*** installation are supported and can function independently.
      * ***LoadMgr*** stdout and stderr are automatically output to the following log file:

$LOG\_ROOT/${package\_name}/loadmgr\_{timestampe}.log.

* + - * All log files for a package are stored under $LOG\_ROOT/ {package\_name}.

## How Does LoadMgr Log Job Status

***LoadMgr*** updates a status file while jobs are running in real-time. The file is stored in the $LOG\_ROOT/status directory when jobs are executing and the moved to the $LOG\_ROOT/ directory upon successful completion of a package. The naming convention of the status file uses the filename of the current configuration file, a date/time stamp, and the file extension .log.

Therefore, if you started the SAS process initial load by typing:

controller.sh ro52\_src2stg

A status file called ro52\_src2stg.log will be created under $LOG\_ROOT/status directory. After all jobs run successfully, the status file will be updated and moved to the $LOG\_ROOT/ro52\_src2stg directory.

The log file name contains three parts:

*{package***\_***name}\_date\_timestamp***.log**

***LoadMgr 5.5*** comes with sample configuration files and dummy programs for install test. The configuration file sample.conf is at $LOADMGR\_HOME/etc. If the package is executed at 9:17pm of May 8, 2012, the log file will be named as sample\_2012\_05\_08\_211745.log.

Figure below shows a sample status file for package sample.

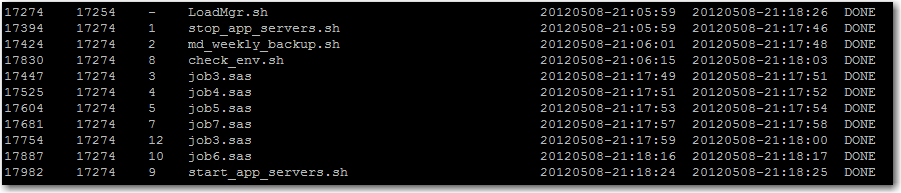
 **DO NOT manually remove status file**

Figure LoadMgr Status file

## LoadMgr SAS Log Analyzer

### Development Background

One of common issues of SAS deployment is how to monitor and handle warning and error messages in SAS log files. Often supporting staff at client companies is new to SAS, and does know how to deal with warning and error messages. Due to the complexity of the solution and the sensitivity of various processes to source data and host environment, SAS consultants have to monitor errors, warning, and sometime even notes that can indicate fatal failures for important routine processes. Furthermore, batch processes mostly run during off-hours and some of them run well through midnight. So monitoring and diagnosing large amount of log files becomes a costly task; the SAS Log Analyzer addresses this need.

### Log Analyzer Functions

* The log analyzer accumulates knowledge about process errors and warnings either from run-time or from manual inputs;
* It has the capability of applying its knowledge and reporting for errors and warnings with suggested action;
* It is highly efficient to analyze a large number of log files, which can be up to GBs in size;
* It can email log analysis report

### How does log analyzer work?

In runs time, the analyzer will read in the message strings and patterns defined in configuration files to parse all log files in a given directory (LoadMgr package log directory), see Section 1.6 for details. It then sends captures the messages to a SAS program for analysis and reporting. Once it finishes the analysis, it will create a report file, and update the knowledge base for the process. The knowledge base file name is the package name and it will have extension .msg. For example, if the directory name is ro52\_src2stg, then the knowledge base file is ro52\_src2stg.msg. When it finishes report writing, it will call LoadMgr email notification module and send the reports to the application administrator(s). The report will look like:

*Known Log Message Summary Report By Suggested Action*

*job3.log*

*-------------------------------------------*

*Suggested Action - REVIEW: 1*

*Suggested Action - FIX: 1*

*-------------------------------------------*

*job4.log*

*-------------------------------------------*

*Suggested Action - IGNORE: 1*

*-------------------------------------------*

*Total Log Message Summary Report By Message Type*

*job3.log*

*-------------------------------------------*

*total number of ERROR message: 1*

*total number of WARNING message: 1*

*-------------------------------------------*

*job4.log*

*-------------------------------------------*

*total number of WARNING message: 1*

*-------------------------------------------*

*Detail Message and Suggested Action Repo****rt***

*job3.log*

*-------------------------------------------*

*FIX ERROR: A lock is not available for DI\_MON.JOB\_PARAM.DATA*

*REVIEW WARNING: DMS bold font metrics fail to match DMS font.*

*-------------------------------------------*

*job4.log*

*-------------------------------------------*

*IGNORE WARNING: Statement terminated early due to OUTOBS=100 option.*

The above report indicates:

* There is one ERROR in job3.log, which needs to be fixed;
* There is one WARNING message in job3.log, which needs review;
* In addition, there is WARNING message in job4, which should be ignored.

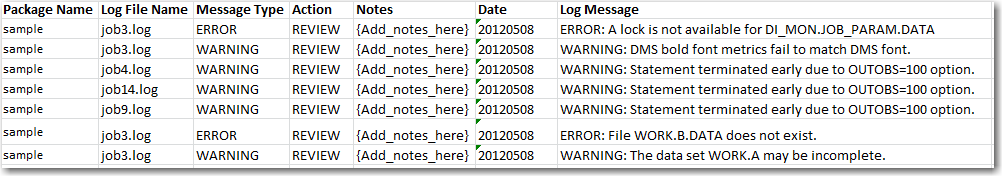
### Log message knowledge file structure

The structure of the knowledge base file is shown in Figure 2 below:

Figure Log Message Knowledge File Structure

It is a pipe delimited flat file in the directory that contains the log files for the LoadMgr package.

### Set up suggested action for known messages

Since the analyzer retains the histories of log message, it provides a simple way for solution administrator to set up suggested action for a known message. The default suggested action for a new message captured is REVIEW. You can edit the knowledge base file and change according to the severity of the log message such as setting action as IGNORE, RERUN, DQ\_CHECK, etc. You can either edit the file directly using UNIX edit like Vi, Emac/s, or download the file and edit it in Excel and then save it pipe delimited flat file.

# Installation and Setup

## Installation Instructions

### Download LoadMgr

Download the current version of ***LoadMgr*** from the Prism directory:

\vertical\Retail\optimization\tools\LoadMgr::Main.

Untar the software from the directory under which you wish to install ***LoadMgr***. The untar command is:

tar xvf LoadMgr5.5.tar

You should see the following directory structure upon completion of the untar command:

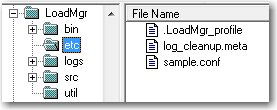


Figure LoadMgr Directory Structure

### Edit LoadMgr Profile

Edit the lines in the **.loadMgr\_profile** file to reflect your deployment environment. **You must configure the Section I parameter**, and replace the autoexec.sas and sasv9.cfg files with ones from your systems, if you want to define explicitly.

The following is an example of a .loadMgr\_profile:

*## Copyright Â©, 2012 by SAS Institute*

*## $Id: .LoadMgr\_profile,v 1.18 2014/03/13 17:54:30 ranliu Exp $*

*## Usage: This profile is sourced at LoadMgr execution time to define its environment*

*## Date By Comment*

*## 2012/03/04 Jozhao This is a part of LoadMgr*

*##*

*## Section I - LoadMgr environment variable*

*## please change before using LoadMgr*

*##*

*export LOADMGR\_HOME=*

*export SRC\_ROOT=$LOADMGR\_HOME/src ## backend program and script home, MAKE SURE YOU CREATE IT*

*export LOG\_ROOT=$LOADMGR\_HOME/logs ## MAKE SURE YOU CREATE the directory*

*export LOG\_BACKUP\_ROOT=$LOG\_ROOT/backup ## batch process log archive home directory, MAKE SURE YOU CREATE IT*

*export ENV\_ID=INTERNAL-DEV ## the ENV\_ID will be include in the LoadMgr email subject line*

*export CONCURRENCY\_LVL=2*

*export CONCURRENCY\_SCOPE=0 ## 0 - only check concurrency within the package, 1- check the concurrency across the OS server*

*export NOTIFICATION\_ADDRESS= # use double quote if more than one email addresses*

*export CC\_NOTIFICATION\_ADDRESSES= # use double quote if more than one email addresses*

*export REPLY\_TO= # set reply-to email address*

*export AWS\_SMTP\_SERVER=0 # if LoadMgr runs on AWS, set to 1*

*export SAS\_WARNING\_NOTIFICATION=0 # 0 - no notification email when sas program finishes with warning*

*## SAS system settings*

*export SASROOT= #enter path for SASROOT*

*export SASCONFIG=$LOADMGR\_HOME/etc/sasv9.cfg #if not set, SAS will pick up configurations based on the Order of Precedence for Processing SAS Configuration Files*

*export SASAUTOEXEC=$LOADMGR\_HOME/etc/autoexec.sas*

*export MEMSIZE=1G*

*export SORTSIZE=512M*

*export REALMEMSIZE=512M*

*##*

*## Section II - Additional LoadMgr environment variable*

*## Please DO NOT make any changes unless you completely understand the impact*

*##*

*## Extend LoadMgr by developing packages of scripts/funcs, i.e. package extension, by set FPATH accordingly*

*export FPATH=$LOADMGR\_HOME/bin/funcs*

*export REST\_TIME=0.5 ## number of seconds to sleep between checking next job to run*

*export LOADMGR\_TMP\_DIR=/tmp ## temp directory that LoadMgr uses for internal temporary files*

*export SYSTEM\_MONITOR=0 ## turn on trace or truss if set to 1 IT WILL GENERATE LOG SYSTEM FILES. make sure trace or truss is installed*

*##*

*## Section III - LoadMgr SAS Log Analyzer Integration Setting*

*##*

*export SAS\_LOG\_ANALYZER=1 # 0-> don't use Analyzer, 1-> use the analyzer. If set to 1, must define [[LOG\_MESSAGE\_INCLUDED]] and [[LOG\_MESSAGE\_EXCLUDED]] sections in*

*# package configuration file*

*export LOG\_REPORT\_LVL=2 # 1-> summary, 2-> summary + detail log message*

*export MAIL\_REPORT=1 # 0-> don't email report, 1-> email report. be aware of potentially email large report file!!!!*

*export LOG\_MAIL\_ADDRESS= # addresses defined here will receive log report, use double quote if more than one email addresses*

*export CC\_LOG\_MAIL\_ADDRESS= # use double quote if more than one email addresses*

*##*

*## Section IV -- Performance and resource monitoring*

*##*

*export MONITOR\_RESOURCE=0 # 0-> don't monitor resource, 1-> use the monitor*

*export MAX\_RUN\_TIME=max # max - no limit on how long a package can run, {n} - max run time is {n} secords*

*export NMON\_CMD=/usr/bin/nmon*

*export NMON\_OPTION\_S=300 #seconds between snap shots*

*export NMON\_OPTION\_C=864 #number of refreshes, $NMON\_OPTION\_S x $NMON\_OPTION\_C should be longer than*

*#any LoadMgr job package since LoadMgr will end the NMON upon the complete of a package*

*##*

*## Section V -- SAS solution sas\_dm backup settings*

*## The online backup is only recommended during deployment. It uses rsync to archive quick incremental backup*

*## for fast recovery IF tape restore time is NOT adequate or operational feasible for the deployment.*

*## Consideration of disk failure should be take into account of the choice of backup file system.*

*export BACKUP\_DIR=*

*export SAS\_DM\_ROOT=*

*export MAX\_INCREMANTAL\_BACKUP=2*

*export DI\_MON\_ROOT=*

*export DI\_SRC\_ROOT=/var/ftp/source\_files/di\_src*

*##*

*## Section VI - LoadMgr event utility and source file management*

*##*

*##*

*export EVENT\_LOG\_ROOT=$LOG\_ROOT/event\_logs*

*export EVENT\_LOG=$LOG\_ROOT/loadmgr\_events.log*

*export CHECK\_EVENT\_TIME=60*

*export EVENT\_PROCESS\_MODE=0 # 0 - sequential 1 - parallel*

*export SRC\_FILE\_LANDING\_DIR=/var/ftp/source\_files*

*export SRC\_FILE\_ARCHIVE\_ROOT=/var/ftp/source\_files/di\_src\_archive*

*export SRC\_FILE\_ENCRYPTION=0 # 0 - no encryption 1 = encrypted*

*export ENCRYPTION\_PASSPHRASE=*

*export SRC\_FILE\_COMPRESS=0 # 0 - no gzip 1 = gzipped*

*##*

*## Section VII Customization -- add any shell profile below to support your jobs*

*##*

To validate your settings, navigate to the bin directory and type:

***checkconf.sh sample***

# Using LoadMgr

## Create LoadMgr Configuration File

The ***LoadMgr*** configuration file defines what and how ***LoadMgr*** executes for a job package. A configuration has five sections, each starts with reserved names in double square brackets. They are [[JOB\_LIST]], [[JOB\_PARAMETER]], [[LOCAL\_SETTING]], [[LOG\_MESSAGE\_INCLUDED]] and [[LOG\_MESSAGE\_EXCLUDED]]. The configuration file supports comment lines, whose first non-black character must be ‘#’.

The following is the content of sample.conf, which comes with ***LoadMgr 5.5***

*[[JOB\_LIST]]*

*1 stop\_app\_servers.sh*

*2 md\_weekly\_backup.sh*

*3 job3.sas 1*

*4 job4.sas 3*

*5 job5.sas 3,4*

*10 job6.sas 7*

*7 job7.sas 4*

*12 job3.sas 1*

*8 check\_env.sh*

*9 start\_app\_servers.sh ALL*

*[[JOB\_PARAMETER]]*

*1|parm value 1*

*3|parm value 2*

*4|12012,02323, 12121*

*12|parm value 3*

*[[LOCAL\_SETTING]]*

*##LoadMgr seeting*

*LOCAL\_CONCURRENCY\_LVL=*

*LOCAL\_SRC\_ROOT=*

*##email address*

*LOCAL\_NOTIFICATION\_ADDRESS=*

*LOCAL\_NOTIFICATION\_CC\_ADDRESS=*

*LOCAL\_REPLY\_TO=*

*##log analyzer setting*

*LOCAL\_SAS\_LOG\_ANALYZER=1 #0-> not run analyzer, 1->run analyzer. If set to 1, [LOG\_MSG\_INCLUDE] and [LOG\_MSG\_EXCLUDE] sections must be defined.*

*LOCAL\_LOG\_REPORT\_LVL=2*

*LOCAL\_LOG\_MAIL\_REPORT=1*

*LOCAL\_LOG\_MAIL\_ADDRESS=john.zhao@sas.com #only email addresses defined here will receive log analysis report, avoid sending large email to SMS address*

*##SAS settings*

*LOCAL\_SASCONFIG=/home/jozhao/sasv9.cfg*

*LOCAL\_SASAUTOEXEC=*

*LOCAL\_MEMSIZE=*

*LOCAL\_SORTSIZE=*

*LOCAL\_REALMEMSIZE=*

*##Other settings*

*LOCAL\_RESOURCE\_MONITOR=1*

*## the string patters will be used to grep text line in SAS log files*

*[[LOG\_MESSAGE\_INCLUDED]]*

*ERROR:*

*WARNING:*

*^NOTES: testing log analyzer*

*## the string patters will be used by grep exclude text line in SAS log files*

*[[LOG\_MESSAGE\_EXCLUDED]]*

*Unable to copy SASUSER*

*Errors printed on page*

*[ ]\*put[ ]+"[ ]\*(ERROR:|WARNING:|NOTE:)[ ]+.+"*

*WARNING: DMS bold font metrics fail to match DMS font*

### Job List Section

The Job List section starts with [[JOB\_LIST]]. This section contains tab-delimited job definition with three columns:

* + - * Job ID: Each job must have a unique ID, which should be a whole number. The Job ID value has nothing to do with the execution sequence. Instead, the jobs are executed based on their dependencies, defined concurrence level, and the order in which the jobs are listed.
      * Job File Physical Name: The file can be either a SAS program or a UNIX/Linux shell script. SAS programs must have the extension ‘.sas’. The shell script must have permission for execution with extension of .sh. There are two ways of creating jobs in .conf file.

a. Job name only without fully qualified path, for example, *job1.sas*. To use this method, you need to create a subfolder under **$LOADMGR\_HOME/src**, using the same name as your package name. Then put all jobs for this package under this folder. For example, you create *sample.conf* in **$LOADMGR\_HOME/etc**, you need to create subfolder named sample under **$LOADMGR\_HOME/src**, then put all jobs for the package *sample.*

b. Specify the job name with fully qualified path. For example

*/data/LoadMgr/src/sample/job1.sas*

* + - * Predecessor Job IDs: This column contains a comma-delimited string that contains predecessor job IDs of the current job. If the current job has no predecessor, leave it blank. If a job depends on all jobs above, you can use the word 'ALL'. If the predecessor Job ID has not been defined in the configuration file, it is an invalid predecessor. When ***LoadMgr*** checks the status of predecessors, it will send out a Warning message and terminate the execution if the predecessor’s Job ID is invalid. Always run the ***checkconf.sh*** utility against a package before running ***LoadMgr***.

### Job Parameter Section

The Job Parameter section starts with [[JOB\_PARAMETER]], and it defines parameter for jobs defined in Job List section. Each line in this section has two columns with the pipe character (|) as its delimiter. The first column is the name of the job file. The names must follow the same rules as described in previous section for a configuration file. The second column is the value of the parameter string.

A package must have jobs defined in Job List section, but the Job Parameter is optional, and it can be empty. Only jobs that have parameters need records in the parameter file. The checkmeta.sh also checks parameter definitions for a package.

### Local Setting Section

The Local Setting for a package starts with [[LOCAL\_SETTING]]. You can overwrite certain ***LoadMgr*** settings defined in .LoadMgr\_Profile. For example, you can specify email addresses that are particular to a package. Please see Local Setting section in the sample.conf for the entire list of parameters you can specify in a .conf file.

List of variables that can be overridden by [[LOCAL\_SETTING]]:

*## define local overwritten env vars*

*export MONITOR\_RESOURCE=${LOCAL\_RESOURCE\_MONITOR:-${MONITOR\_RESOURCE}}*

*export SRC\_ROOT=${LOCAL\_SRC\_ROOT:-${SRC\_ROOT}}*

*export LOG\_ROOT=${LOCAL\_LOG\_ROOT:-${LOG\_ROOT}}*

*export SAS\_LOG\_ANALYZER=${LOCAL\_SAS\_LOG\_ANALYZER:-${SAS\_LOG\_ANALYZER}}*

*export REPORT\_LVL=${LOCAL\_REPORT\_LVL:-${REPORT\_LVL}}*

*export LOCAL\_REPLY\_TO=${LOCAL\_LOCAL\_REPLY\_TO:-${LOCAL\_REPLY\_TO}}*

*export CONCURRENCY\_LVL=${LOCAL\_CONCURRENCY\_LVL:-${CONCURRENCY\_LVL}}*

*export CONCURRENCY\_SCOPE=${LOCAL\_CONCURRENCY\_SCOPE:-${CONCURRENCY\_SCOPE}}*

*export SASROOT=${LOCAL\_SASROOT:-${SASROOT}}*

*export SASCONFIG=${LOCAL\_SASCONFIG:-${SASCONFIG}}*

*export MEMSIZE=${LOCAL\_MEMSIZE:-${MEMSIZE}}*

*export SORTSIZE=${LOCAL\_SORTSIZE:-${SORTSIZE}}*

*export REALMEMSIZE=${LOCAL\_REALMEMSIZE:-${REALMEMSIZE}}*

*export SAS\_LOG\_ANALYZER=${LOCAL\_SAS\_LOG\_ANALYZER:-${SAS\_LOG\_ANALYZER}}*

*export LOG\_REPORT\_LVL=${LOCAL\_LOG\_REPORT\_LVL:-${LOG\_REPORT\_LVL}}*

*export MAIL\_REPORT=${LOCAL\_MAIL\_REPORT:-${MAIL\_REPORT}}*

*export LOG\_MAIL\_ADDRESS=${LOCAL\_LOG\_MAIL\_ADDRESS:-${LOG\_MAIL\_ADDRESS}}*

*export CC\_LOG\_MAIL\_ADDRESS=${LOCAL\_CC\_LOG\_MAIL\_ADDRESS:-${CC\_LOG\_MAIL\_ADDRESS}}*

*export NOTIFICATION\_ADDRESS=${LOCAL\_NOTIFICATION\_ADDRESS:-${NOTIFICATION\_ADDRESS}}*

*export CC\_NOTIFICATION\_ADDRESSES=${LOCAL\_CC\_NOTIFICATION\_ADDRESS:-${CC\_NOTIFICATION\_ADDRESSES}}*

*export REPLY\_TO=${LOCAL\_REPLY\_TO:-${REPLY\_TO}}*

*export SASAUTOEXEC=${LOCAL\_SASAUTOEXEC:-${SASAUTOEXEC}}*

*export MAX\_RUN\_TIME=${LOCAL\_MAX\_RUN\_TIME:-${MAX\_RUN\_TIME}}*

### Log Message Pattern Sections

The entries in section [[LOG\_MESSAGE\_INCLUDED]] and [[LOG\_MESSAGE\_EXCLUDED]] are UNIX pattern strings. The log analyzer will grep patterns defined in LOG\_MESSAGE\_INCLUDED section and excludes ones in the LOG\_MESSAGE\_EXCLUDED section. There is a good book talking about the pattern (A.K.A. Regular Expression), [Mastering Regular Expressions](http://shop.oreilly.com/product/9780596528126.do?green=705BE22B-5FD3-5D1D-A7A5-3366CAAB0A49&intcmp=af-mybuy-9780596528126.IP).

## Validating a LoadMgr Configuration File

To validate the configuration file, type:

***checkconf.sh sample***

If there are any errors in the configuration file such as missing SAS programs or incorrect Job IDs, the ***checkconf.sh*** script will print them to the screen.

**You must validate a configuration file before running a package.**

If you receive the following error message:

Fail to run LoadMgr with error message like "./controller.sh: line 21: function: not found”

The error is most likely due to wrong value for environment variable LOADMGR\_HOME. You need to correct it in etc/.loadMgr\_profile. If you are using the Windows version of ***LoadMgr***, you will need to add the shell variable LOADMGR\_HOME to your .profile, and source the .loadMgr\_profile by adding this line:

$LOADMGR\_HOME/etc/.loadMgr\_profile.

## Executing a Process Using LoadMgr

If there are no errors in the checkconf screen, you can execute the package of jobs in the configuration file by typing:

***controller.sh [Package\_name]***

You can run **controller.sh** from any directory if you fully qualify the path to it.

## Terminating LoadMgr Processes

To terminate a running LoadMgr process, type:

$LOADMGR\_HOME/util/stop\_loadmgr.sh <package name>

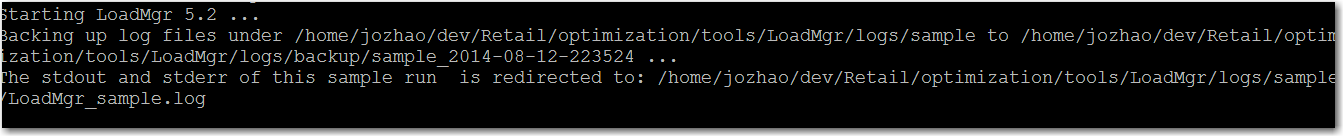
## Monitoring LoadMgr Processes

### Check LoadMgr Process Status Log File

***LoadMgr*** provides a safe way to view the status log file while a ***LoadMgr*** job package is still running. To use checkstatus.sh, type:

**$LOADMGR\_HOME/util/check\_status.sh <package name>**

### Check LoadMgr Stand Out and Stand Error

When you kick off a LoadMgr package, you will see the message on screen like the one below:

To monitor LoadMgr process, you can issue a tail –f command to monitor the progress.

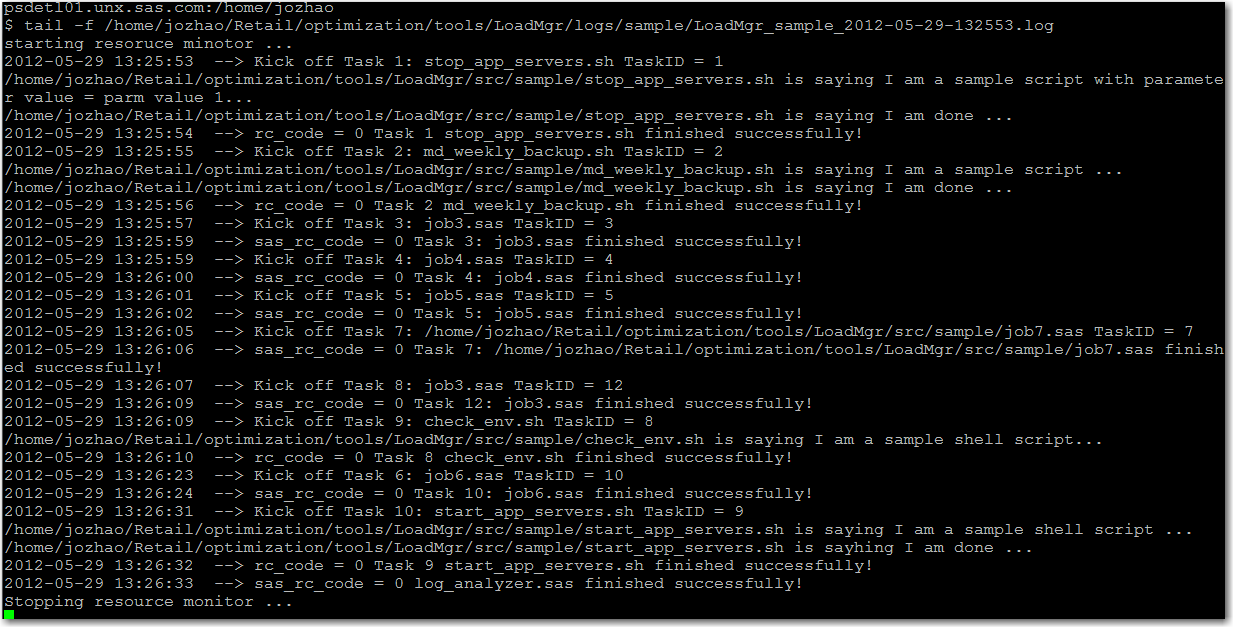


Figure Controller Stdout and Stderror

## Forcing a Fresh Restart after a Job Failure

By default, controller.sh will restart a previously failed session in resume mode. However, there are times when you may need to force a fresh restart. For example, if a job failed after you submitted:

$LOADMGR\_HOME/bin/controller.sh ro52\_src2stg

You could then force a fresh restart by typing:

$LOADMGR\_HOME/bin/controller.sh –f ro52\_src2stg

Always ensure you wait till the entire package is terminated before the restart.

## Re-run a Failed Job

When a job fails, you will need to find out and address the root cause, and typically, you will resume the process after LoadMgr terminates it. However, if you want to re-run the failed job, you can use the re-run failed job utility tool by typing:

$LOADMGR\_HOME/util/rerun\_failed\_job.sh <package name> <Job ID>

The tool will re-run the failed job as a part of running LoadMgr process. It will update LoadMgr process status log file, and sends stdout from the job to ${LOG\_ROOT}/${PACKAGE\_NAME}/external\_rerun.log.

If LoadMgr process finishes before a re-run of failed job, the tool will stop and report the LoadMgr is no longer running.

## Log File Archival

Log files from the most recent execution of a process are located under $LOG\_ROOT/ {package\_name}. Before ***LoadMgr*** executes a package, it moves \*.lst, \*.log, \*.nmon and \*.html files under $LOG\_ROOT/{package\_name} to $LOG\_BACKUP\_ROOT/{package\_name}\_{date}\_{timestamp}.

## Automate LoadMgr Log Clean-Up Process

To automate log clean up for a particular ***LoadMgr*** package, add the package name to **$LOADMGR\_HOME/etc/log\_cleanup.meta** and the number of days to keep the log file for the package.

For example,

sample 2

ro52\_src2stg 90

To clean up the log for all packages, type:

**$LOADMGR\_HOME/util/log\_cleanup.sh**

The clean-up process will remove the archived log files in $LOG\_ROOT/backup directory for the ro52\_src2stg package that are older than 90 days. However, it will retain log files from the most recent execution of the ro52\_src2stg package.

## Running nmon within LoadMgr

You need to ensure nmon is installed on your server, and if not, download and install nmon first. The version tested with ***LoadMgr*** is nmon 11e. Prior to running nmon within ***LoadMgr***, run it from a command line to ensure it works. To activate nmon in LoadMgr, edit $LOADMGR\_HOME/etc/.loadMgr\_profile, and set **MONITOR\_RESOURCE=1**, or set **LOCAL\_RESOURCE\_MONITOR=1** in conf file for the package. Adjust the values of the nmon options **-s** and **-c** to fit your needs.

export MONITOR\_RESOURCE=0 # 0-> don't monitor resource, 1-> use the monitor

export NMON\_CMD=/usr/bin/nmon

export NMON\_OPTION\_S=300 #seconds between snap shots

export NMON\_OPTION\_C=864 #number of refreshes, $NMON\_OPTION\_S x

$NMON\_OPTION\_C should be longer than

#any LoadMgr job package since LoadMgr will

end the NMON upon the complete of a package

## Running a Backup Utility during a Load

***LoadMgr*** uses the populate [rsync](http://samba.anu.edu.au/rsync/) to get snapshots of SAS solution datamart. The idea of using replication to create a snapshot-like system was designed by [Mike Rubel](http://www.mikerubel.org/computers/rsync_snapshots/) in his online article about rsync with snapshots. You can read his article at <http://www.mikerubel.org/computers/rsync_snapshots>. The script **sas\_dm\_backup.sh** is an enhanced version of Mike’s original work that allows you to define the numbers of snapshots. Since the tool uses UNIX hard link and rsync fast incremental file transfer, it provides robust backup and quick restore. The versions of snapshot will be named as:

$BACKUP\_DIR/sas\_dm.{version}

Where sas\_dm.0 is the most recent backup.

To use the utility, you simple define the parameters in .loadMgr\_profile, and then run:

$LOADMGR/util/sas\_dm\_backup.sh

To restore, issue the following command:

rsync -av --delete $BACKUP\_DIR/sas\_dm.{version}/. $SAS\_DM\_ROOT/

**Caution!** If you specify –delete options, any files that do not exist under $BACKUP\_DIR/sas\_dm.{version}, but are currently under $SAS\_DM\_ROOT/ will be removed.

## Section V -- SAS Solution sas\_dm backup settings

## The online backup is only recommended during deployment. It uses rsync to archive quick incremental backup

## for fast recovery IF tape restore time is NOT adequate or operational feasible for the deployment.

## Consideration of disk failure should be take into account of the choice of backup file system.

export BACKUP\_DIR=

export SAS\_DM\_RO

export MAX\_INCREMANTAL\_BACKUP=4

It is recommended that you read the articles about rsync and rsync snapshot before using the tool. In addition, the backup utility is NOT a replacement for enterprise backup system.

## Using FULLSTIMER Log Parser

The FULLSTIMER Log Parser is developed by Michael A. Raithel, see SAS Notes “[Sample 34301: Parse SAS® Logs to Extract Performance and Timing Information.”](http://support.sas.com/kb/34/301.html) It is enhanced and integrated in LoadMgr. To enable the feature, you need enable SAS system option FULLSTIMER and set FULLSTIMER\_PARSE to 1 in **.LoadMgr\_profile**. To retain fullstimer statistics for the LoadMgr package, set RETAIN\_FULLSTIMER\_HIST to YES.

export FULLSTIMER\_PARSE=1 # 1-> to run the parser to capture fullstimer stats into SAS dataset for performance analysis. Make sure FULLSTIMER is on before turn on this parser.

export RETAIN\_FULLSTIMER\_HIST=NO # YES -> to keep fullstimer stats history, be mindful of the size of the SAS dataset

The FULLSTIMER statistics extracted from SAS logs for LoadMgr package will be saved in a SAS database that is collocated in LoadMgr log directory. In addition to variables defined by Michael original code, realtime\_vs\_cputime is calculated for performance analysis. Please read Michael’s article “[Programmatically Measure SAS® Application Performance On Any Computer Platform With the New LOGPARSE SAS Macro](http://www2.sas.com/proceedings/sugi30/219-30.pdf).”