# Load Manager Version 5.5

## User Guide

# Table of Contents

## Contents

# 1 Load Manager Overview

## 1.1 Introduction

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

## 1.2 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.

## 1.3 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.

## 1.4 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.

## 1.5 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}.`

## 1.6 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`.

| | | | | | | |
|---|---|---|---|---|---|---|
| 17274 | 17254 | - | LoadMgr.sh | 20120508-21:05:59 | 20120508-21:18:26 | DONE |
| 17394 | 17274 | 1 | stop_app_servers.sh | 20120508-21:05:59 | 20120508-21:17:46 | DONE |
| 17424 | 17274 | 2 | md_weekly_backup.sh | 20120508-21:06:01 | 20120508-21:17:48 | DONE |
| 17830 | 17274 | 8 | check_env.sh | 20120508-21:06:15 | 20120508-21:18:03 | DONE |
| 17447 | 17274 | 3 | job3.sas | 20120508-21:17:49 | 20120508-21:17:51 | DONE |
| 17525 | 17274 | 4 | job4.sas | 20120508-21:17:51 | 20120508-21:17:52 | DONE |
| 17604 | 17274 | 5 | job5.sas | 20120508-21:17:53 | 20120508-21:17:54 | DONE |
| 17681 | 17274 | 7 | job7.sas | 20120508-21:17:57 | 20120508-21:17:58 | DONE |
| 17754 | 17274 | 12 | job3.sas | 20120508-21:17:59 | 20120508-21:18:00 | DONE |
| 17887 | 17274 | 10 | job6.sas | 20120508-21:18:16 | 20120508-21:18:17 | DONE |
| 17982 | 17274 | 9 | start_app_servers.sh | 20120508-21:18:24 | 20120508-21:18:25 | DONE |

***Figure 1 LoadMgr Status file***

**DO NOT manually remove status file**

# 1.7 LoadMgr SAS Log Analyzer

## 1.7.1 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.

## 1.7.2 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

## 1.7.3 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 Report
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.

## 1.7.4 Log message knowledge file structure

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

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

***Figure 2 Log Message Knowledge File Structure***

## 1.7.5 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.

| Package Name | Log File Name | Message Type | Action | Notes | Date | Log Message |
|---|---|---|---|---|---|---|
| sample | job3.log | ERROR | REVIEW | {Add_notes_here} | 20120508 | ERROR: A lock is not available for DI_MON.JOB_PARAM.DATA |
| sample | job3.log | WARNING | REVIEW | {Add_notes_here} | 20120508 | WARNING: DMS bold font metrics fail to match DMS font. |
| sample | job4.log | WARNING | REVIEW | {Add_notes_here} | 20120508 | WARNING: Statement terminated early due to OUTOBS=100 option. |
| sample | job14.log | WARNING | REVIEW | {Add_notes_here} | 20120508 | WARNING: Statement terminated early due to OUTOBS=100 option. |
| sample | job9.log | WARNING | REVIEW | {Add_notes_here} | 20120508 | WARNING: Statement terminated early due to OUTOBS=100 option. |
| sample | job3.log | ERROR | REVIEW | {Add_notes_here} | 20120508 | ERROR: File WORK.B.DATA does not exist. |
| sample | job3.log | WARNING | REVIEW | {Add_notes_here} | 20120508 | WARNING: The data set WORK.A may be incomplete. |

# 2 Installation and Setup

## 2.1 Installation Instructions

### 2.1.1 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:

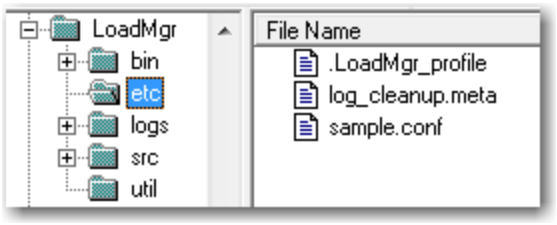


***Figure 3 LoadMgr Directory Structure***

### 2.1.2 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***

# 3 Using LoadMgr

## 3.1 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
```

### 3.1.1 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***.

## 3.1.2 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.

## 3.1.3 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}}
```

### 3.1.4 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.

## 3.2 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.

## 3.3 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.

## 3.4 Terminating LoadMgr Processes

To terminate a running LoadMgr process, type:

$LOADMGR_HOME/util/stop_loadmgr.sh <package name>

## 3.5 Monitoring LoadMgr Processes

### 3.5.1 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>**

### 3.5.2 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:

```
Starting LoadMgr 5.2 ...
Backing up log files under /home/jozhao/dev/Retail/optimization/tools/LoadMgr/logs/sample to /home/jozhao/dev/Retail/optim
ization/tools/LoadMgr/logs/backup/sample_2014-08-12-223524 ...
The stdout and stderr of this sample run  is redirected to: /home/jozhao/dev/Retail/optimization/tools/LoadMgr/logs/sample
/LoadMgr_sample.log
```

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

```
psdetl01.unx.sas.com:/home/jozhao
$ tail -f /home/jozhao/Retail/optimization/tools/LoadMgr/logs/sample/LoadMgr_sample_2012-05-29-132553.log
starting resoruce minotor ...
2012-05-29 13:25:53  --> Kick off Task 1: stop_app_servers.sh TaskID = 1
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/stop_app_servers.sh is saying I am a sample script with paramete
r value = parm value 1...
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/stop_app_servers.sh is saying I am done ...
2012-05-29 13:25:54  --> rc_code = 0 Task 1 stop_app_servers.sh finished successfully!
2012-05-29 13:25:55  --> Kick off Task 2: md_weekly_backup.sh TaskID = 2
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/md_weekly_backup.sh is saying I am a sample script ...
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/md_weekly_backup.sh is saying I am done ...
2012-05-29 13:25:56  --> rc_code = 0 Task 2 md_weekly_backup.sh finished successfully!
2012-05-29 13:25:57  --> Kick off Task 3: job3.sas TaskID = 3
2012-05-29 13:25:59  --> sas_rc_code = 0 Task 3: job3.sas finished successfully!
2012-05-29 13:25:59  --> Kick off Task 4: job4.sas TaskID = 4
2012-05-29 13:26:00  --> sas_rc_code = 0 Task 4: job4.sas finished successfully!
2012-05-29 13:26:01  --> Kick off Task 5: job5.sas TaskID = 5
2012-05-29 13:26:02  --> sas_rc_code = 0 Task 5: job5.sas finished successfully!
2012-05-29 13:26:05  --> Kick off Task 7: /home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/job7.sas TaskID = 7
2012-05-29 13:26:06  --> sas_rc_code = 0 Task 7: /home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/job7.sas finish
ed successfully!
2012-05-29 13:26:07  --> Kick off Task 8: job3.sas TaskID = 12
2012-05-29 13:26:09  --> sas_rc_code = 0 Task 12: job3.sas finished successfully!
2012-05-29 13:26:09  --> Kick off Task 9: check_env.sh TaskID = 8
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/check_env.sh is saying I am a sample shell script...
2012-05-29 13:26:10  --> rc_code = 0 Task 8 check_env.sh finished successfully!
2012-05-29 13:26:23  --> Kick off Task 6: job6.sas TaskID = 10
2012-05-29 13:26:24  --> sas_rc_code = 0 Task 10: job6.sas finished successfully!
2012-05-29 13:26:31  --> Kick off Task 10: start_app_servers.sh TaskID = 9
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/start_app_servers.sh is saying I am a sample shell script ...
/home/jozhao/Retail/optimization/tools/LoadMgr/src/sample/start_app_servers.sh is sayhing I am done ...
2012-05-29 13:26:32  --> rc_code = 0 Task 9 start_app_servers.sh finished successfully!
2012-05-29 13:26:33  --> sas_rc_code = 0 log_analyzer.sas finished successfully!
Stopping resource monitor ...
```

***Figure 4 Controller Stdout and Stderror***

## 3.6 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.

## 3.7 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.

## 3.8 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}`.

## 3.9 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.

## 3.10 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
```

## 3.11 Running a Backup Utility during a Load

***LoadMgr*** uses the populate rsync to get snapshots of SAS solution datamart. The idea of using replication to create a snapshot-like system was designed by Mike Rubel 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.

## 3.12 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." 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."