# FMW System Requirements

How to create automation scripts in order to test limits of a system to decide system readiness for installation of fusion middleware products on a fusion middleware environment.

## Preparation

In the example, we will be using the following software:

* [Python 2.7.16](https://www.python.org/ftp/python/2.7.16/python-2.7.16.amd64.msi)[[1]](#footnote-1)

### System Requirements

#### OS Type/Version:

RED HAT version 6 or 7 with latest patch level

#### Required packages minimum version:

|  |
| --- |
| binutils-2.20.51.0.2-5.28.el6 compat-libcap1-1.10-1 compat-libstdc++-33-3.2.3-69.el6 for x86\_64 compat-libstdc++-33-3.2.3-69.el6 for i686 gcc-4.4.4-13.el6 gcc-c++-4.4.4-13.el6 glibc-2.12-1.7.el6 for x86\_64 glibc-2.12-1.7.el6 for i686 glibc-devel-2.12-1.7.el6 for i686 libaio-0.3.107-10.el6 libaio-devel-0.3.107-10.el6 libgcc-4.4.4-13.el6 for x86\_64  libgcc-4.4.4-13.el6 for i686 libstdc++-4.4.4-13.el6 for x86\_64 libstdc++-4.4.4-13.el6 for i686 libstdc++-devel-4.4.4-13.el6 libXext for x86\_64 libXext for i686 libXtst for x86\_64 libXtst for i686 openmotif-2.2.3 for x86\_64 openmotif22-2.2.3 for x86\_64 sysstat-9.0.4-11.el6  zlib-1.2.3-29.el6.x86\_64  zlib-1.2.3-29.el6.i686  ksh  make-3.82  numactl-2.0.9 for x86\_64  numactl-devel-2.0.9 for x86\_64  motif-2.3.4-7 for x86\_64  motif-devel-2.3.4-7 for x86\_64 |

#### Open file limits requirement:

|  |
| --- |
| sed -i '/.\*End of file/d' /etc/security/limits.conf  echo "\* soft nofile 65536" >> /etc/security/limits.conf  echo "\* hard nofile 65536" >> /etc/security/limits.conf  echo "\* soft nproc 32768" >> /etc/security/limits.conf  echo "\* hard nproc 32768" >> /etc/security/limits.conf  echo "# End of file"  >> /etc/security/limits.conf |

#### Kernel parameters:

|  |
| --- |
| echo "net.core.rmem\_max=4192608" >> /etc/sysctl.conf  echo "net.core.wmem\_max=4192608" >> /etc/sysctl.conf  echo "kernel.shmall = 4294967295" >> /etc/sysctl.conf  echo "kernel.shmmax = 68719476736" >> /etc/sysctl.conf  echo "kernel.msgmax = 65536" >> /etc/sysctl.conf  echo "kernel.msgmnb = 65536" >> /etc/sysctl.conf  echo "vm.swappiness = 10" >> /etc/sysctl.conf |

#### User and group requirements:

|  |
| --- |
| groupadd dba [with GID 500] [if not already exist]  useradd -g dba -G dba,users oracle [with UID 500]  useradd -g dba -G dba,users orafmw [with UID 550] |

#### Directories and Space requirements:

|  |
| --- |
| Mount / 100GB  Local Mount /u01/apps/middleware 100GB – FMW binaries  NFS-Mount /u02 50GB – WEBLOGIC Domain Configuration  NFS-Mount /u03 100GB – WEBLOGIC Application and Domain Logs  NFS-Mount /media 100GB – Software/Patch staging area  NFS-Mount /apps 50GB – Application package staging  NOTE:  NFS mounts shared between the servers in the cluster environment |

#### Umask requirement for oracle:

|  |
| --- |
| echo "umask 027" >> .bash\_profile |

#### Enabling Unicode Support in the .bash\_profile for oracle.

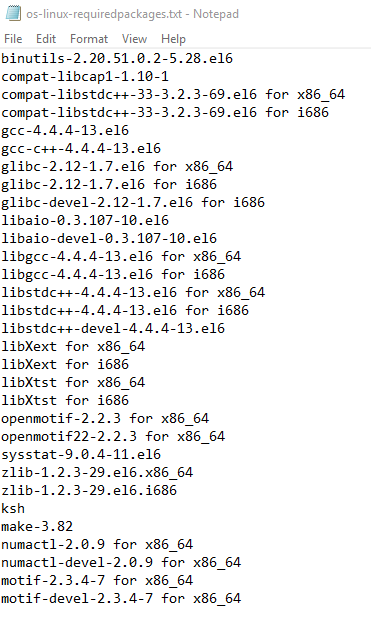
|  |
| --- |
| echo "export LANG=\"en\_US.UTF-8\"" >> .bash\_profile |

#### Memory and CPU requirement:

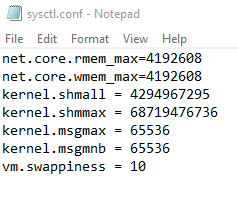
|  |
| --- |
| **Memory:**  **PRD -** 32GB Minimum  NON-PRD - 16GB Minimum  **Swap:**  PRD - 20GB  ACC - 16GB  **CPU:**  PRD – 2 [Minimum Xeon 2.30 GHz with 2 Cores]  ACC - 2 [Minimum Xeon 2.30 GHz with 2 Cores] |

### Directory Structure

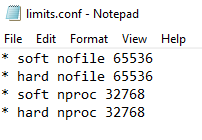
First, we have to set up the directory structure. For machines/ servers (must be operating on Linux) we must create a file to be stored in the same location as the automation script, to store the required packages. From the above requirements list, below is an example text file containing the required packages:



For testing on Windows, I also created a temporary configuration file containing system controls with corresponding values:



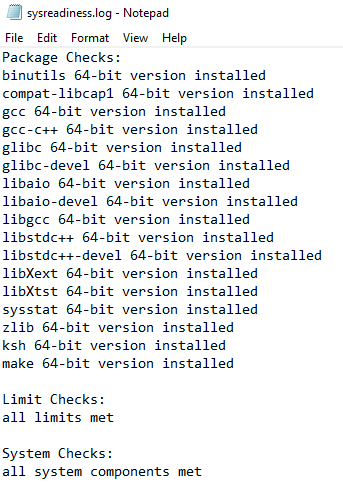
I also created another configuration file containing the limits for number of files and number of processes:



You can use file handling to grab the appropriate text and corresponding values for both system controls and limits configuration files and compare them with the above limits when testing on Windows.

### Response Files

After the script has finished executing, up to two files are stored. During the script, many temporary log and text files are created to store outputs from OS commands in a log directory but are deleted once the required information has been recalled from those files. The two potential files are the system readiness log file and system readiness error file. The log file when all system readiness checks pass should look as follows:



However, if any checks are failed, they will be stored in a system readiness error file which is only created on the encounter of an error. The script will determine is the file has been created at the end and display the appropriate output.

1. Versions of Python from 3.0 and above can be used to create automation scripts however the latest version installed on the test Linux servers used for development and testing was Python 2.7.16. [↑](#footnote-ref-1)