TimeShiftX® 8.x

Linux - Solaris - HPUX - AIX Manual

Table of Contents

Overview

What is TimeShiftX (TSX)? How does it work?

TimeShiftX (TSX) is a simulation software that lets you time travel software into the future or past for temporal testing of all date sensitive functionality and code. TSX uses virtual clocks to enable time travel inside of Active Directory & Kerberos without changing system clocks, changing code, or isolating servers. TSX works by intercepting the OS date & time calls made by software programs and swapping them with your custom user-created virtual clock, all the while leaving your OS system files, timestamps and security tokens untouched.

System Requirements

Operating System (OS)	
Linux x86-x64	Red Hat Enterprise Linux 5, 6, 7, 8, 9 Oracle Enterprise Linux 5, 6, 7, 8, 9 CentOS Linux 5, 6, 7, 8, 9 SUSE Enterprise Linux 10, 11, 12, 15 Amazon Linux 1, 2, 2023 Ubuntu 14, 15, 16, 18, 20, 21, 22 Debian 7, 8, 9, 10, 11, 12 Alpine Linux 3.x
UNIX	Solaris x86-64 10, 11 Solaris SPARC 10, 11 HPUX 11 Itanium AIX 6, 7
Linux arm64	Use TSX v9+
Disk Space	30 MB
Memory	64 MB RAM
Software	libgcc (both 32 & 64 bit versions) Solaris: libgcc v4+ AIX: libgcc v4.8.5-5+
Access rights	Local administrator rights (root) for install only
Note:	Installation may reveal additional libraries missing on your server. If your OS is not listed above, email support for the latest compatibility list. libgcc not needed for HPUX

Technical Support

For assistance, email support@vornexinc.com or call 408.713.1400 ext. 2

Install TSX

As root or sudo, untar the agent and run the install script. Replace yellow with your version.

```
# tar -xvf tsx-<version>.tar

# cd tsx-<version>

# ./install.sh
```

For Alpine Linux, run this command instead `./install.sh force`
For Solaris, libgcc v4+ must be installed with symlinks setup beforehand.
For AIX, libgcc4.8.5-5+ must be installed with symlinks setup beforehand. Read AIX libgcc Steps
For AIX Java apps based on J9 JVM (IBM Java, WebSphere Java, OpenJDK with OpenJ9), run these steps post install - AIX Java J9 Setup Steps

Install complete!

Restart any running apps or databases one time to initially load the TSX library.

Now License TSX

Install Steps - Solaris 11 Zones

Run the above steps and only install TSX on the exact zone you want to time travel (whether global, local/non-global, or both).

Install Steps - Solaris 10 Zones

For Solaris 10, if you want to time travel on a local (non-global) zone, install TSX on the Global zone, then on the desired local (non-global) zones. If you want to time travel only the Global zone, then only install there. Note: Using TSX time travel in local (non-global) zones will not affect the global zone.

On Global Zone

```
1. # tar -xvf tsx-<version>.tar
2. # cd tsx-<version>
3. # ./install.sh
```

On Local (non-global) Zone

4. Repeat steps 1-3
    a. Depending on your zones, you may see errors after install saying “ln: cannot create ….: Read-only file system”. These can be ignored.
5. Open a new terminal session -> restart any running apps. (or restart server/zone)
6. Install Finished

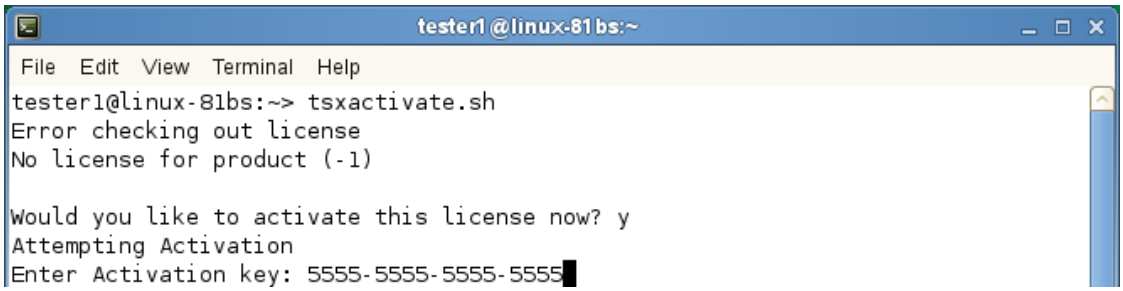
License TSX

Your server needs internet access. If none, skip to step 4 a)
Below is for trial & node-locked licenses. For floating licenses, email support@vornexinc.com

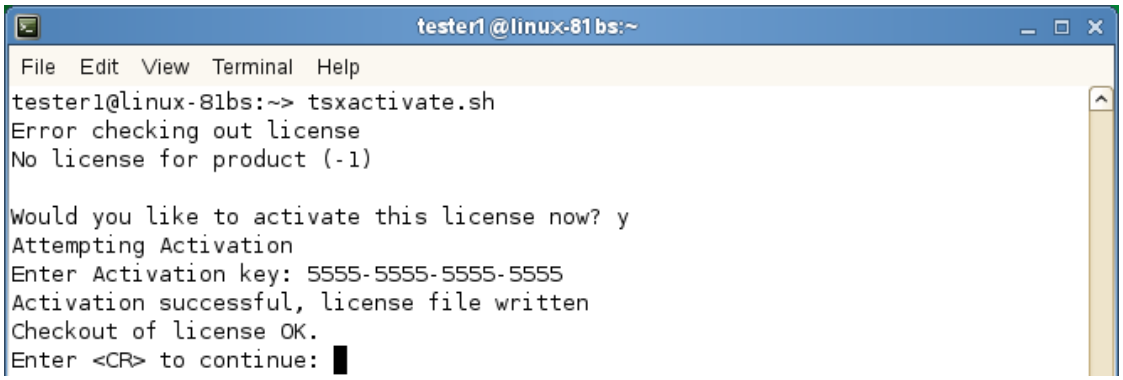
1. As a regular non-root user, run our license utility

```
$ tsxactivate.sh
```

   a. This will license by connecting to http://hostedactivation.com/ on port 80. To use the HTTPS version via port 443 run `$ tsxactivate.sh https`
2. Type **Y** then press **Enter**
3. Paste your 16-digit key then press **Enter**



4. If successful, you will see "Activation Successful". Press **Enter** to finish.
   a. If unsuccessful, send the server output of the below to support@vornexinc.com
      For Linux: `# ifconfig -a` For AIX: `# hostid`
      For Solaris: `# /bin/hostid` For HPUX: `# echo `uname -i` 16op | dc`



5. Run `$ tsx list` You should see " 0 Virtual Clocks set "
   a. If you get an error, fix license permissions via the below, then rerun `$ tsx list`
      `$ sudo chmod 644 /etc/tsx/license/a0.lic`

Note: Do not change the OS system clock as it will break the license.

Licensing complete. Now Smoke Test TSX

Smoke Test TSX

Run the below smoke test to ensure TSX is properly setup. Replace bob with any local non-root user

```
# su - bob

$ tsx set -u bob -d 1

$ date
```

Should see one day in the future.

```
$ tsx reset

$ date
```

Should see the present time

If the above shows time travel on / off, TSX is working!

If the above fails, try the below.

a) Open a new shell / connection and re-try smoke test
b) If on Linux, run the below to fix file permissions and re-try smoke test.

```
$ sudo chmod a+r /etc/ld.so.preload
```

If after a) & b) the smoke test still fails, then run $ tsxsupport.sh and email the generated file (tsxsupport.log) to support@vornexinc.com

Smoke Test complete. TSX is fully setup. Begin time travelling!

Learn TSX Time Travel

There are three ways or app IDs to time travel with TSX. These are on an Operating System (OS) level.

1) User / Service ID
    a. **Most popular**
    b. All apps or selected apps running as the defined OS user/service ID will time travel
        i. Pick & choose apps via Include – Exclude Apps

2) Oracle Database SID
    a. **Use for Oracle Databases**
    b. Only the exact Oracle Database Instance running with the unique SID environment variable will time travel
        i. TSX will only time travel the "oracle{sid}" and related process/subprocesses
        ii. Other oracle processes (OEM, Oracle Monitoring Agents, etc) will not time travel.
        iii. All internal container/pluggable DBs associated with the SID will time travel.

3) Process ID (PID)
    a. Apps running as the defined PID(s) will time travel
        i. PID is a dynamic number. If the app restarts the PID will change. If the app spawns new processes, new PIDs will be generated. You must re-run TSX to capture the new PID numbers.
        ii. PID is the 2nd column of ps -ef output. 3rd column PPID is not supported
        iii. Not supported for HP-UX. Linux support on v6+ Solaris+AIX support on v6.7+
        iv. Once a time traveling PID terminates, TSX will automatically disable the time travel for that terminated PID # as it has been released back to the OS to be used for another process

To find the ID for your app, run `ps -ef`

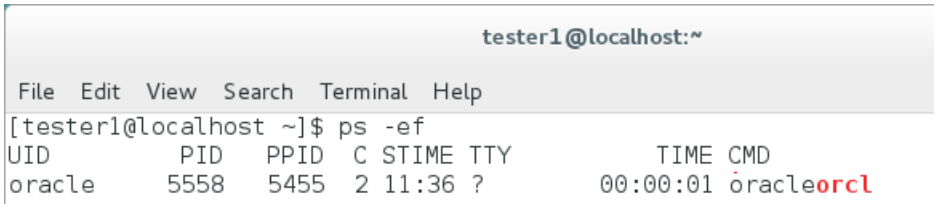


Figure 1 Example: User/Service ID = oracle PID = 5558 SID = orcl

Notes

- Apps are any binary / exe on the system, whether a database, java application, etc.
- Virtual clocks will progress forward in time (as a normal clock) unless otherwise defined.

Use TSX

Use CLI terminal commands to create/edit/remove/list a virtual clock. Multiple IDs are space delimited. If you remove "-u {USER}", TSX will time travel for the user of the current shell session.
Accounts with special symbols or spaced out words must be wrapped in double quotes. Enter users without domain info.

Create an Offset Virtual Clock

```
$ tsx set -u {User Names} [-y | -w| -d | -h | -m | -s] {integer}

$ tsx set -p {PID Numbers} [-y | -w| -d | -h | -m | -s] {integer}

$ tsx set -sid {SID Names} [-y | -w| -d | -h | -m | -s] {integer}
```

Offset virtual clocks add or subtract a value relative to the real system time. Use any combination of the below flags. Positive integers go in the future; negative integers, with a hyphen, go in the past.

-y {integer}	Years	-h {integer}	Hours
-w {integer}	Weeks	-m {integer}	Minutes
-d {integer}	Days	-s {integer}	Seconds

Examples	
Time travel Joe and Tom 5 years and 2 weeks in the *future*	`$ tsx set -u Joe Tom -y 5 -w 2`
Time travel user oracle for 4 weeks & 10 days in the *past*	`$ tsx set -u oracle -w -4 -d -10`
Time travel PID 123 and 234 1 year into the *future*	`$ tsx set -p 123 456 -y 1`
Time travel SID oracle3 2 years into the *past*	`$ tsx set -sid oracle3 -y -2`

Create a Specific Virtual Clock

```
$ tsx set -u {User Names} [mm/dd/yyyy hh:mm:ss]

$ tsx set -p {PID Numbers} [mm/dd/yyyy hh:mm:ss]

$ tsx set -sid {SID Names} [mm/dd/yyyy hh:mm:ss]
```

Specific virtual clocks define an exact date and time to jump to. Allowed date/time formats are below. Dates are in US Standard format. Times are in 24-hour format.

mm/dd/yyyy hh:mm:ss	Date and time
mm/dd/yyyy hh:mm	Date and time (without seconds)
mm/dd/yyyy	Date only. 2 digit month / 2 digit day / 4 digit year
hh:mm:ss	Time only. 2 digit hour : 2 digit minute : 2 digit seconds
hh:mm	Time only (without seconds). 2 digit hour : 2 digit minute

Note:

- If you only define the date, the current system time will be used.
- If you only define the time, the current system date will be used
- We recommend to **not** time travel seconds so that your servers stay in sync with NTP

Examples	
Time travel user ora1 to July 11th 2025 11:45 PM	`$ tsx set -u ora1 07/11/2025 23:45`
Time travel users sql1 and Joe for July 30th 2006	`$ tsx set -u sql1 joe 07/30/2006`
Time travel PID 789 to June 10th 2010	`$ tsx set -p 789 06/10/2010`
Time travel SID orcl2 for July 10th 2030 at 10AM	`$ tsx set -sid orcl2 07/10/2030 10:00`

PID Grep String Ability

If an app consists of many PIDs, instead of typing each you can enter a Grep command to auto-populate PIDs. After the ' -p ' flag enter a grep string wrapped with the special character `

Example
Time travel the oracle database instance sid "oracle3" 5 years into the future
`$ tsx set -p` **`` `ps -ef | grep -v grep | grep oracle3 | awk '{print $2}'` ``** `-y 5`

Delete all virtual clocks

```
$ tsx reset
```

All apps will revert back to present time.

List all virtual clocks

```
$ tsx list
```

Note: The below Freeze, Speed, Loop, Edit commands apply to all ID types. Just replace "-u {User Names}" with "-p {PID Numbers}" or "-sid {SID Names}"

Freeze virtual clocks

```
$ tsx set -u {User Names} [-y | -w | -d | -h | -m | -s] {integer} -f

$ tsx set -u {User Names} [mm/dd/yyyy hh:mm:ss] -f
```

Append the ' –f ' flag to a virtual clock to cause it to stop progressing forward and be frozen.

Speed virtual clocks faster or slower

```
$ tsx set -u {User Names} [-y | -w | -d | -h | -m | -s] {integer} -v {integer}

$ tsx set -u {User Names} [mm/dd/yyyy hh:mm:ss] -v {integer}
```

Append the ‘ -v ’ flag to a virtual clock followed by an integer to have your virtual clocks travel N times faster or slower than real system time.

- A speed value equal to the number 1 is real time speed
- A speed value greater than the number 1 is faster than real time
- A speed value less than the number 1 is slower than real time
- Speed value notation is 5 digits to the left and right of the decimal point. {XXXXX.nnnnn}

Speed is in relation to how fast seconds travel. See below examples

Normal Speed	-v 1
2 times faster	-v 2
4 times faster	-v 4
10 times faster	-v 10
2 times slower	-v 0.5
4 times slower	-v 0.25
10 times slower	-v 0.1

Travel 1 minute every second	-v 60
Travel 10 minutes every second	-v 600
Travel 30 minutes every second	-v 1800
Travel 1 hour every second	-v 3600
Travel 1 day every second	-v 86400
Travel 1 second every minute	-v 0.016
Travel 1 second every 10 minutes	-v 0.00166

Examples	
Create an Offset clock for Tom five years into the future at a speed 5 times faster than real time	`$ tsx set -u Tom -y 5 -v 5`
Create a Specific clock for Bob for October 10th 2010 at a speed of 4 times slower than real time	`$ tsx set -u Bob 10/10/2010 -v .25`

Loop virtual clocks

```
$ tsx set -u {User Names} [-y | -w | -d | -h | -m | -s] {integer} -l [ m | h | d | w | M | y | integer]

$ tsx set -u {User Names} [mm/dd/yyyy] [hh:mm:ss] -l [ m | h | d | w | M | y | integer]
```

Append the ‘ -l ’ (lower case L) flag to a virtual clock followed by a supported letter or integer to cause it to loop on the current value. This means the virtual clock will tick forward normally then will roll back to the current value, never crossing over to the new minute/hour/day/week/month/year value.

Flag	Definition	Meaning
-l	Loop every day same as “-l d”	At 23:59:59 time will roll back to 00:00:00 of the **same day**
-l m	Loop every minute	At hh:mm:59 time will roll back to hh:mm:00 of the **same minute**
-l h	Loop every hour	At hh:59:59 time will roll back to hh:00:00 of the **same hour**
-l d	Loop every day	At 23:59:59 time will roll back to 00:00:00 of the **same day**
-l w	Loop every week	At Saturday 23:59:59 time will roll back to Sunday 00:00:00 of the **same week**
-l M	Loop every month	At the last day of the month on 23:59:59 time will roll back to 00:00:00 of the first day of the **same month**
-l y	Loop every year	At the last day of the year at 23:59:59 time will roll back to 00:00:00 of the first day of the **same year**
-l [integer]	Loop every X seconds	At the last second of a defined range, roll back to the original second of the same range

Note 1: TSX versions below 8.6.4 only support looping on one single day
Note 2: To progress forward but keep your looping virtual clock, use the Edit feature.
Note 3: You cannot use speed or frozen flags with looping.

Examples	
Set a virtual clock for Bob to always loop on July 10 2030	`$ tsx set -u Bob 07/10/2030 -l`
Set a virtual clock for Bill to always loop on July 10 2030	`$ tsx set -u Bill 07/10/2030 -l d`
Create a minute looping virtual clock 1 year and 10 days into the future for Tom and Steve	`$ tsx set -u Tom Steve -y 1 -d 10 -l m`
Travel Larry to July 2 2002 at 10AM and loop it on that hour	`$ tsx set -u Larry 07/02/2002 10:00 -l h`
Travel Mary to 1 year in past and loop on a 60 second range	`$ tsx set -u Mary -y -1 -l 60`

Edit existing virtual clocks

```
$ tsx set -e -u {User Names} {-y | -w | -d | -h | -m | -s} {integer} [-v integer]
```

Append the '-e' flag after ' set ' and before ' -u ' and add any of the below combinations of flags to add or subtract values to an existing virtual clock or to adjust its speed. Positive integers go in the future; negative integers, with a hyphen, go in the past. The existing virtual clock type (normal, frozen, speed, loop) will be preserved unless defined in the new edit command in which case the type will be changed.

-y {integer}	Years	-h {integer}	Hours
-w {integer}	Weeks	-m {integer}	Minutes
-d {integer}	Days	-s {integer}	Seconds
-f	Freeze speed*	-v {integer}	Speed value*
-l	Loop*		

*Refer to speed, frozen, looping sections in manual

Examples	
Add 1 week (into the future) to Phil's current virtual clock.	`$ tsx set -e -u Phil -w 1`
Add 10 days and 1 hour (into the future) to Tom and Steve's existing virtual clock	`$ tsx set -e -u Tom Steve -d 10 -h 1`
Adjust Bob's virtual clock to travel 10x faster than real-time.	`$ tsx set -e -u Bob -v 10`
Subtract 1 year (into the past) to Bill's existing virtual clock and adjust it to travel 1/4 slower than real time	`$ tsx set -e -u Bill -y -1 -v .25`

Delete individual virtual clocks

```
$ tsx remove -u {User Names}

$ tsx remove -p {PID Numbers}

$ tsx remove -sid {SIDs}
```

Only apps running as the defined user accounts, PIDs, or SID instances will revert back to system time.

Delete all Users virtual clocks only

```
$ tsx reset users
```

All user virtual clocks will be deleted only. Other types of virtual clocks (PID, SID) will remain.

Delete all PID virtual clocks only

```
$ tsx reset pids
```

All PID virtual clocks will be deleted only. Other types of virtual clocks (User, SID) will remain.

Delete all SID virtual clocks only

```
$ tsx reset sids
```

All SID virtual clocks will be deleted only. Other types of virtual clocks (User, PID) will remain.

List exact types of virtual clocks

```
$ tsx list users

$ tsx list pids

$ tsx list sids
```

Uninstall TSX

1. As root, run

   ```
   # tsxuninstall.sh
   ```

2. Restart any current running apps or restart the server / zone.

Uninstall Complete. TSX is now successfully removed.

AIX libgcc Steps

For AIX, libgcc.4.8.5-5 or greater is a requirement. Follow the below steps before TSX installation.

1) Check your libgcc version via `$ lslpp -L libgcc`

2) If no libgcc.4.8.5-5+ exists…
   a. Install via `yum install libgcc*`
      or
      Manually download and install as root via `# rpm -i libgcc{aix_version}.rpm`
      See RPMs if needed
      i. AIX 6.1: https://s3.amazonaws.com/Vornex/libgcc-4.8.5-5.aix6.1.ppc.rpm
      ii. AIX 7.1: https://s3.amazonaws.com/Vornex/libgcc-4.8.5-5.aix7.1.ppc.rpm
      iii. AIX 7.2: https://s3.amazonaws.com/Vornex/libgcc-4.8.5-5.aix7.2.ppc.rpm
   b. Resume Install TSX steps

3) If libgcc.4.8.5-5+ exists, check the symlinks exist via the below. After, resume Install TSX steps

```
$ ls -l /opt/freeware/lib64/libgcc_s.a
$ ls -l /opt/freeware/lib/libgcc_s.a
```

   If the above show no symlinks, create them using your path via the below

```
$ ln -s {path_to_your_32b_lib}/libgcc_s.a /opt/freeware/lib/libgcc_s.a
$ ln -s {path_to_your_64b_lib}/libgcc_s.a /opt/freeware/lib64/libgcc_s.a
```

4) If libgcc exists but it is not 4.8.5-5+, send your rep the output of `$ lslpp -L libgcc`

Note: If after TSX install you get below errors, it means libgcc.4.8.5-5+/symlinks are not setup properly.

```
Could not load module /usr/lib/tsx32.so.
Dependent module /opt/freeware/lib64/libgcc_s.a(shr.o) could not be loaded.
The module has an invalid magic number.
```

To fix, as root run `# /etc/tsx/bin/tsxuninstall.sh` Then re-follow the above steps.

AIX libgcc setup. Now Install TSX

AIX Java J9 Setup Steps

To time travel Java apps based on J9 JVM (e.g. IBM Java, WebSphere Java, or OpenJDK with OpenJ9), you must update the Java libraries with TimeShiftX. Follow the below. Support in TSX v8.8.0+

1) As root, run the below command

```
# tsxjava.sh install [java_path]
```

For [java_path], enter the root java folder containing the “bin/java” directory tree.

Example
Time travel IBM Java 7.1 x64bit libraries on AIX
 # tsxjava.sh install /usr/java71_64/

2) Done! Now restart any current running Java apps one time to initially load the TSX library. Now License TSX

Notes

- To enable TSX for multiple Java libraries and versions just re-follow the above steps.
- To check which Java libraries have TSX enabled run

```
# tsxjava.sh list
```

- To remove TSX from an individual Java library and revert it back to its original state, run

```
# tsxjava.sh remove [java_path]
```

For [java_path], enter the root java folder used before for installing it.

e.g. # tsxjava.sh remove /usr/java71_64/

- To remove TSX from all Java libraries and revert the libraries back to their original state, run

```
# tsxjava.sh removeall
```

AIX library setup complete. Now License TSX

Include – Exclude Apps (Binaries)

By default, user virtual clocks will time travel all apps running as the defined user. To include or exclude specific apps, follow the below guide. Three modes exist. A user can only be in 1 mode at a time. Note: Inclusion/Exclusion is not supported for Alpine Linux.

Inclusion mode = Only time travel the defined apps
Exclusion mode = Time travel all apps except the defined apps
Default mode (Full Inclusion) = All apps time travel.

On Linux & Solaris, use the full binary path (not symlink path). On AIX & HP-UX 11.31 use the binary name.

Include apps

```
$ tsx include {Program Binary Path} -u {Account Name}
```

Example	
Linux-Solaris: Include only gedit to see John's virtual clock	`$ tsx include /usr/bin/gedit -u John`
AIX-HPUX 11.31: Include only gedit to see Tom's virtual clock	`$ tsx include gedit -u Tom`

Exclude apps

```
$ tsx exclude {Program Binary Path} -u {Account Name}
```

Example	
Linux-Solaris: Exclude java 7 from seeing John's virtual clocks	`$ tsx exclude /usr/local/java/jre1.7.0_51/bin/java -u John`
AIX- HPUX 11.31: Exclude java from seeing Tom's virtual clocks	`$ tsx exclude java -u Tom`

List included apps

```
$ tsx list-include
```

List excluded apps

```
$ tsx list-exclude
```

Delete all apps from inclusion-exclusion list

```
$ tsx mode-reset
```

Revert back to the default "full inclusion" mode and clear all inclusion/exclusion settings.

Remove specific apps from inclusion-exclusion list

```
$ tsx mode-remove {Program Binary Path} -u {Account Name}
```

Show a user's mode

```
$ tsx mode
```

Containers / Pods

TSX is compatible with Linux containers/pods, whether managed by Docker, Kubernetes, OpenShift, or others. Simply install TSX in each container that requires time travel or build TSX into your image. Then run a TSX command to time travel your apps running in the container/pod.

Time traveling the container will not affect / time travel the host. The host does not need TSX installed.

Email support@vornexinc.com for the Container/Pod Guide for full integration steps.

Release Notes

For details about various items (shared apps, time zones, timers, synchronization, daylight savings, cluster / load balanced environments, cron, Task Scheduler, PDB, IIS, etc.) please refer to **TimeShiftX_Notes.pdf**

Silently Install & License TSX

To silently setup TSX via the command prompt, follow the below

1. Silent Install (requires root rights)

```
$ tar -xvf tsx-<version>.tar
$ cd tsx-<version>
$ su
# ./install.sh
```

2. Silent License (requires internet access, if none contact support@vornexinc.com)

```
$ cd /etc/tsx/license/
$ ./activate key 5555-5555-5555-5555
Error checking out license
No license for product (-1)
Activation successful, license file written
```

3. Begin testing!

Note: You can broadcast these commands using an IT management tool like Puppet.

Frequently Asked Questions

1. Does TSX affect system files or file timestamps?

   No, TSX does not time travel OS File Timestamps like “Date modified, File Creation Date, etc.” as it will cause system instability. However if an app like WebSphere generates log files with time labeled filenames or timestamp messages printed inside the logs, then those will be time travelled.

2. Does TSX work with Active Directory?

   Yes. TSX maintains all environment level security tokens at all times, so you can safely time-travel your apps while inside an Active Directory or Kerberos environment.

3. What happens if the system clock is changed manually after setting a virtual clock with TSX?

   The license will become invalid.