

DAITSS demonstration installation guide

The purpose of this guide is to guide you through installing a simplified, demonstration instance of DAITSS. The process in this document will result in a fully functional DAITSS system. However, some of the configuration choices assumed below are not recommended for production use.

This document is intended for those who would like to evaluate DAITSS, but cannot run the DAITSS demonstration VM or would prefer to build and configure a DAITSS system themselves.

This document assumes the following:

- That the reader is familiar with Linux and the Linux commandline interface
- The operating system being used is RHEL 6.x or CentOS 6.x
- The operating system is installed with the "desktop minimal" preset
- A user named 'daitss' created as part of operating system installation process
- A separate partition is created and mounted at /var/daitss/silo as part of the installation process

Installing DAITSS system dependencies

Updating the operating system

While strictly speaking not a requirement, it is a good idea to begin by updating the operating system itself. To update the operating system, type:

```
[root@shades]# yum update
```

Ruby

To install Ruby and Ruby development libraries, type:

```
[root@shades]# yum install ruby
[root@shades]# yum install ruby-devel
```

RubyGems

To install RubyGems, type:

```
[root@shades]# yum install rubygems
```

After installing RubyGems, it is necessary to update it. Type:

```
[root@shades]# gem update --system
```

Git

To install Git, type:

```
[root@shades]# yum install git
```

Sun Java 6

RHEL/CentOS 6 comes with the OpenJDK Java implementation. For DAITSS, It is strongly recommended that the Sun Java 6 implementation be used.

First, download the Sun Java 6 binary package:

32-bit machine:

```
[root@shades]# wget http://download.oracle.com/otn-pub/java/jdk/6u29-b11/jdk-6u29-linux-i586-rpm.bin
```

64-bit machine:

```
[root@shades]# wget http://download.oracle.com/otn-pub/java/jdk/6u29-b11/jdk-6u29-linux-x64-rpm.bin
```

After it has finished downloading, make the binary package executable and run it.

```
[root@shades]# chmod +x jdk-6u27-linux-x64-rpm.bin  
[root@shades]# ./jdk-6u27-linux-x64-rpm.bin
```

The binary package will install Sun Java 6 to '/usr/java/jdk1.6.0_27'

PostgreSQL

To install PostgreSQL, type:

```
[root@shades]# yum install postgresql  
[root@shades]# yum install postgresql-contrib  
[root@shades]# yum install postgresql-server  
[root@shades]# yum install postgresql-devel
```

GCC

To install GCC, type:

```
[root@shades]# yum install gcc  
[root@shades]# yum install gcc-c++
```

LibXML

To install LibXML, type:

```
[root@shades]# yum install libxml2-devel
```

LibXSLT

To install LibXSLT, type:

```
[root@shades]# yum install libxslt-devel
```

Libcurl

To install Libcurl, type:

```
[root@shades]# yum install libcurl-devel
```

ClamAV

To install ClamAV, it is first necessary to install the EPEL repository. To do this:

```
[root@shades]# rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-5.noarch.rpm
```

After the EPEL repository has been installed, ClamAV can be installed:

```
[root@shades]# yum install clamav  
[root@shades]# yum install clamd
```

Syslog

To install syslog, type:

```
[root@shades]# yum install rsyslog
```

crontabs

To install crontabs, type:

```
[root@shades]# yum install crontabs
```

Squid

To install Squid, type:

```
[root@shades]# yum install squid
```

Apache Dev Libraries

To install the Apache Dev Libraries, type:

```
[root@shades]# yum install httpd-devel
```

Preservation Tools

To install ffmpeg, mencoder and libquicktime, tools required for preservation processing, the RPMForge repository must be set up. To set up the RPMForge repository:

For 64-bit machines:

```
[root@shades]# rpm -Uvh http://packages.sw.be/rpmforge-release/rpmforge-release-0.5.2-2.el6.rf.x86\_64.rpm
```

For 32-bit machines:

```
[root@shades]# rpm -Uvh http://packages.sw.be/rpmforge-release/rpmforge-release-0.5.2-2.el6.rf.i686.rpm
```

After the RPMForge repository is configured, the preservation tools can be installed. To install the preservation tools, type:

```
[root@shades]# yum install ffmpeg  
[root@shades]# yum install mencoder  
[root@shades]# yum install libquicktime
```

Configuring DAITSS dependencies

Now that we've installed all the software dependencies needed by DAITSS, we can work on configuring them.

Sun Java 6 configuration

First, we want to set `$JAVA_HOME` to point to Sun Java 6 for the daitss user. To do this, open the file `/home/daitss/.bashrc` and add the following lines to the end of the file:

```
export JAVA_HOME=/usr/java/jdk1.6.0_29
export PATH=$JAVA_HOME/bin:$PATH
```

Now we must set Sun Java 6 as the default Java implementation in the system wide alternatives system. To do this, type:

```
[root@shades]# alternatives --install /usr/bin/java java /usr/
java/jdk1.6.0_27/bin/java 20000
```

PostgreSQL configuration

The steps below outline the creation of a Postgres database cluster, the creation of a user, and the required postgres databases.

The first step is to initialize the Postgres database cluster. To do this, type:

```
[root@shades]# service postgresql initdb
```

Now, we will configure Postgres authentication to use MD5 for network connections, and trust for local connections. These security settings are not appropriate for a production machine, and used here only to simplify the database configuration for demonstration purposes.

Open the file `'/var/lib/pgsql/data/pg_hba.conf'`

replace the line:

```
local all all ident
```

with this line:

```
local all all trust
```

replace the line:

```
host all all 127.0.0.1/32 ident
```

with this line:

```
host all all 127.0.0.1/32 md5
```

replace the line:

```
host all all ::1/128 ident
```

with this line:

```
host all all ::1/128 md5
```

Next, start the database service:

```
[root@shades]# service postgresql start
```

Next, we must create a Postgres database user.

To do this, type:

```
[root@shades]# su postgres  
[postgres@shades]% createuser daitss
```

Now, we should set a password for the user we just created. First, we will start a Postgres client session, then set the password.

To start a Postgres client session:

```
[postgres@shades]% psql
```

In the Postgres client, type:

```
postgres=# ALTER USER daitss WITH PASSWORD 'daitss';
```

To quit the client, type:

```
postgres=# \q
```

Lastly, we need to create some Postgres databases required by DAITSS. To do so, type:

```
[postgres@shades]% createdb daitss
[postgres@shades]% createdb silo_pool_db
[postgres@shades]% createdb storage_master_db
```

Installing required RubyGems

There are 5 RubyGems that need to be installed system-wide for DAITSS to function correctly. To install them, type:

```
[root@shades]# gem install rake
[root@shades]# gem install bundler
[root@shades]# gem install rack
[root@shades]# gem install thin
[root@shades]# gem install sys-proctable --version 0.9.0 --
platform x86-64-linux
```

Getting DAITSS source, Building local dependencies

We've installed all the system-wide dependencies. It's time to get the DAITSS software and build the dependencies specific to the DAITSS components.

The first step is to create the directories needed by DAITSS:

```
[root@shades]# mkdir /opt/web-services
[root@shades]# mkdir /var/daitss
[root@shades]# mkdir /var/run/daitss
[root@shades]# mkdir /var/log/daitss
[root@shades]# chown daitss:daitss /opt/web-services
[root@shades]# chown daitss:daitss /var/daitss
[root@shades]# chown daitss:daitss /var/run/daitss
[root@shades]# chown daitss:daitss /var/log/daitss
```

As the daitss user, type the following commands to continue creating the directories needed by DAITSS:

```
[daitss@shades]% mkdir /var/daitss/silo
[daitss@shades]% mkdir /var/daitss/data
```

```
[daitss@shades]% mkdir /var/daitss/tmp
[daitss@shades]% mkdir /var/daitss/xmlresolution
[daitss@shades]% mkdir /var/daitss/xmlresolution/collections
[daitss@shades]% mkdir /var/daitss/xmlresolution/schemas
[daitss@shades]% mkdir /var/log/daitss/web-services
[daitss@shades]% mkdir /var/log/daitss/thin
[daitss@shades]% mkdir /var/log/daitss/submit
[daitss@shades]% mkdir /var/log/daitss/daemons
[daitss@shades]% mkdir /var/run/daitss/thin
[daitss@shades]% mkdir /opt/web-services/sites
[daitss@shades]% mkdir /opt/web-services/conf.d
[daitss@shades]% mkdir /opt/web-services/conf.d/thin
[daitss@shades]% chmod 777 /var/daitss/tmp
```

The rest of the commands in this section should be executed as the daitss user.

Installing Core service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites
[daitss@shades]% git clone git://github.com/daitss/core.git
```

Git will download Core service to /opt/web-services/sites/core. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/core/
```

Open the file 'Gemfile' with your favorite text editor and replace the line:

```
gem 'sys-proctable', :path => '/opt/ruby-1.8.7/lib/ruby/gems/
1.8/gems/sys-proctable-0.9.0-x86-linux'
```

with the line (32-bit machine):

```
gem 'sys-proctable', :path => '/usr/lib/ruby/gems/1.8/gems/sys-
proctable-0.9.0-x86-linux'
```

or the line (64-bit machine):

```
gem 'sys-proctable', :path => '/usr/lib64/ruby/gems/1.8/gems/
sys-proctable-0.9.0-x86-linux'
```

In the same file, 'Gemfile', remove the line:

```
gem dm-mysql-adapter
```

Now, close and save the file and type in the terminal:

```
[daitss@shades]% bundle install --path bundle
```

All of the dependencies for Core service will download and compile.

Installing Actionplan service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/  
actionplan.git
```

Git will download Actionplan service to /opt/web-services/sites/actionplan. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/actionplan/  
[daitss@shades]% bundle install --path bundle
```

All of the dependencies for Actionplan service will download and compile.

Installing Description service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/describe.git
```

Git will download Description service to /opt/web-services/sites/describe. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/describe/  
[daitss@shades]% bundle install --path bundle
```

Installing Silo-Pool service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/silo-pool.git
```

Git will download Silo-Pool service to /opt/web-services/sites/silo-pool. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/silo-pool/  
[daitss@shades]% bundle install --path bundle
```

Installing Storage Master service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/store-  
master.git storage-master
```

Git will download Storage Master service to /opt/web-services/sites/storage-master. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/storage-master/  
[daitss@shades]% bundle install --path bundle
```

Installing Transformation service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/transform.git
```

Git will download Transformation service to /opt/web-services/sites/transform. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/transform/  
[daitss@shades]% bundle install --path bundle
```

Installing Viruscheck service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/  
viruscheck.git
```

Git will download Viruscheck service to /opt/web-services/sites/viruscheck. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/viruscheck/  
[daitss@shades]% bundle install --path bundle
```

Installing XML Resolution service

First, download the source code. Type:

```
[daitss@shades]% cd /opt/web-services/sites  
[daitss@shades]% git clone git://github.com/daitss/  
xmlresolution.git
```

Git will download XML Resolution service to /opt/web-services/sites/xmlresolution. When it completes, It's time to build the dependencies. To do so, type:

```
[daitss@shades]% cd /opt/web-services/sites/xmlresolution/  
[daitss@shades]% bundle install --path bundle
```

Miscellaneous configuration

Configuring Logging

Before we can configure the DAITSS services, we must configure logging in rsyslog so that the syslog facility code used by DAITSS is associated with a file.

To do this, open the file '/etc/rsyslog.conf' and add the following line:

```
local0.*    /var/log/daitss/common.log
```

Now, we must restart rsyslog. To do so, type:

```
[root@shades]# /etc/init.d/rsyslog restart
```

Configuring Apache

We will be using Apache as a proxy for the DAITSS web-services. Before we can do so, we must configure Apache. Open the file '/etc/httpd/conf/httpd.conf' and add the following lines:

```
NameVirtualHost *:80
Include /opt/web-services/conf.d/*.conf
```

Next, we must configure SELinux (Security Enhanced Linux) to allow Apache to connect to other HTTP servers running locally (the DAITSS services). Type:

```
[root@shades]# togglesebool httpd_can_network_connect
```

Lastly, we will be using named virtual hosts for our DAITSS 2 services. In lieu of configuring DNS CNAMEs for the virtual hosts, we will be adding entries in the system hosts file for the names we will be using. Open the file '/etc/hosts' and add the following lines:

```
127.0.0.1    silo.shades.local
127.0.0.1    storagemaster.shades.local
127.0.0.1    actionplan.shades.local
127.0.0.1    core.shades.local
127.0.0.1    describe.shades.local
127.0.0.1    transform.shades.local
127.0.0.1    viruscheck.shades.local
127.0.0.1    xmlresolution.shades.local
```

Configuring Silo-Pool service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 4 steps in setting up Silo-Pool service. They are:

- setting up the DAITSS configuration file for Silo-Pool service
- setting up the database
- adding a silo
- setting up the Silo-Pool service webserver

Setting up DAITSS configuration

Create the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines:

```
database:
  silo_db: postgres://daitss:daitss@localhost/silo_pool_db

silo.shades.local:
  log_database_queries:      false
  silo_temp_directory:      /var/daitss/tmp
  log_syslog_facility:      LOG_LOCAL0
  log_filename:             /var/log/daitss/web-services/
silo.log
```

Setting up the Silo-Pool Database

Silo-Pool service includes a script to initialize its database. To run the script, type:

```
[daitss@shades]% cd /opt/web-services/sites/silo-pool
[daitss@shades]% bundle exec tools/create-db --db-string
postgres://daitss:daitss@localhost/silo_pool_db
```

Adding a Silo

Now that the Silo-Pool database has been created, the next step is to add a new silo to the database. Silo-Pool service includes a script to add a silo. We want to add a silo representing the disk we have provisioned and mounted on /var/daitss/silo. To run the script, type:

```
[daitss@shades]% cd /opt/web-services/sites/silo-pool
[daitss@shades]% bundle exec tools/add-silos --db-string
postgres://daitss:daitss@localhost/silo_pool_db --server-name
"silo.shades.local" /var/daitss/silo
```

Configuring the Silo-Pool webserver

There are two configuration files we must create to configure the Silo-Pool service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/silo.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName silo.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://silo_servers>
        BalancerMember http://127.0.0.1:7000
    </Proxy>

    ProxyPass / balancer://silo_servers/
    ProxyPassReverse / balancer://silo_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/silo.yml' with the following lines:

```
user: daitss
group: daitss
tag: silo
environment: production
pid: /var/run/daitss/thin/silo.pid
log: /var/log/daitss/thin/silo.log
port: 7000
chdir: /opt/web-services/sites/silo-pool
timeout: 300
```

Configuring Storage Master service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 4 steps in setting up Storage Master service. They are:

- setting up the DAITSS configuration file for Storage Master service
- setting up the database

- adding a silo-pool
- setting up the Storage Master service webservice

Setting up DAITSS configuration

Open the file `/opt/web-services/conf.d/daitss-config.yml` and add the following line to the database section:

```
storemaster_db: postgres://daitss:daitss@localhost/  
storage_master_db
```

Add the following lines to the end of the file:

```
defaults:  
  required_pools: 1  
  
storagemaster.shades.local:  
  log_syslog_facility: LOG_LOCAL0  
  log_database_queries: false  
  log_filename: /var/log/daitss/web-services/  
storagemaster.log
```

Setting up the Storage Master Database

Storage Master service includes a script to initialize its database. To run the script, type:

```
[daitss@shades]% cd /opt/web-services/sites/storage-master  
[daitss@shades]% bundle exec tools/create-db --db-string  
postgres://daitss:daitss@localhost/storage_master_db
```

Adding a Silo-Pool

Now that the Storage Master database has been created, the next step is to add a new Silo-Pool to the database. Storage Master service includes a script to add a silo pool. We want to add to the database a Silo-Pool representing the Silo-Pool service we just configured. To run the script, type:

```
[daitss@shades]% cd /opt/web-services/sites/storage-master  
[daitss@shades]% bundle exec tools/add-pools --db-string  
postgres://daitss:daitss@localhost/storage_master_db  
silo.shades.local
```

Configuring the Storage Master webservice

There are two configuration files we must create to configure the Storage Master service webservice, one for the Apache proxy, and another for the thin webservice that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/storagemaster.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName storagemaster.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://storagemaster_servers>
        BalancerMember http://127.0.0.1:7100
    </Proxy>

    ProxyPass / balancer://storagemaster_servers/
    ProxyPassReverse / balancer://storagemaster_servers/

</VirtualHost>
```

Next, let's configure thin. Create the file '/opt/web-services/conf.d/thin/storagemaster.yml' with the following lines:

```
user: daitss
group: daitss
tag: storagemaster
environment: production
pid: /var/run/daitss/thin/storagemaster.pid
log: /var/log/daitss/thin/storagemaster.log
port: 7100
chdir: /opt/web-services/sites/storage-master
timeout: 300
```

Configuring Actionplan service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 2 steps in setting up Actionplan service. They are:

- setting up the DAITSS configuration file for Actionplan service
- setting up the Actionplan service webserver

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines to the end of the file:

```
actionplan.shades.local:
    log_syslog_facility:      LOG_LOCAL0
    log_filename:            /var/log/daitss/web-services/
actionplan.log
```

Configuring the Actionplan webserver

There are two configuration files we must create to configure the Actionplan service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/actionplan.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName actionplan.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://actionplan_servers>
        BalancerMember http://127.0.0.1:7200
    </Proxy>

    ProxyPass / balancer://actionplan_servers/
    ProxyPassReverse / balancer://actionplan_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/actionplan.yml' with the following lines:

```
user: daitss
group: daitss
tag: actionplan
environment: production
pid: /var/run/daitss/thin/actionplan.pid
log: /var/log/daitss/thin/actionplan.log
port: 7200
chdir: /opt/web-services/sites/actionplan
timeout: 300
```

Configuring Core service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 4 steps in setting up Core service. They are:

- setting up the DAITSS configuration file for Core service
- setting up daitss user environment
- setting up the Core service database
- setting up the Core service webserver

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following line to the database section:

```
daitss_db: postgres://daitss:daitss@localhost/daitss
```

Now, add the following lines to the end of the file:

```
core.shades.local:
  log_syslog_facility: LOG_LOCAL0
  log_filename: /var/log/daitss/web-services/
core.log
  submit_log_directory: /var/log/daitss/submit
  pulse_log_filename: /var/log/daitss/daemons/
pulse.log
  mailer_log_filename: /var/log/daitss/daemons/
reporter.log
```

```
d1_globals_dir:

data_dir:                /var/daitss/data

uri_prefix:              daitss-demo://
http_timeout:            600

actionplan_url:          http://actionplan.shades.local
describe_url:            http://describe.shades.local
storage_url:             http://storagemaster.shades.local

viruscheck_url:          http://viruscheck.shades.local
transform_url:           http://transform.shades.local
xmlresolution_url:       http://xmlresolution.shades.local

ingest_throttle:         1
dissemination_throttle: 1
dlrefresh_throttle:     0
withdrawal_throttle:    1
queueing_discipline:    lifo

smtp_server:

jvm_options:
- -Xmx256m
- -Dhttp.proxyHost=localhost
- -Dhttp.proxyPort=3128
```

Configuring User Environment

The scripts bundled with the Core Service require that two environment variables be set. To set these variables, open the file '/home/daitss/.bashrc' and add these lines:

```
export DAITSS_CONFIG=/opt/web-services/conf.d/daitss-config.yml
export VIRTUAL_HOSTNAME=core.shades.local
```

Now, lets load these changes into our current shell. Type:

```
[daitss@shades]% source /home/daitss/.bashrc
```

Configuring up Core Service Database

Bundled with core is a script to initialize the Core database and create work directories. To run the script, type:

```
[daitss@shades]% cd /opt/web-services/sites/core
[daitss@shades]% bundle exec ./bin/init
```

Configuring the Core webserver

There are two configuration files we must create to configure the Core service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/core.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName core.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://core_servers>

        BalancerMember http://127.0.0.1:7300
    </Proxy>

    ProxyPass / balancer://core_servers/
    ProxyPassReverse / balancer://core_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/core.yml' with the following lines:

```
user: daitss
group: daitss
tag: core
environment: production
pid: /var/run/daitss/thin/core.pid
log: /var/log/daitss/thin/core.log
port: 7300
```

```
chdir: /opt/web-services/sites/core
timeout: 300
```

Configuring Description service

All of the commands in this section should be run as the daitss user, unless otherwise specified.

There are 2 steps in setting up Description service. They are:

- setting up the DAITSS configuration file for Description service
- setting up the Description service webserver

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines to the end of the file:

```
describe.shades.local:
  log_syslog_facility:      LOG_LOCAL0
  log_filename:            /var/log/daitss/web-services/
describe.log
  max_pdf_bitstreams: 1000

  jvm_options:
    - -Xmx256m
    - -Dhttp.proxyHost=localhost
    - -Dhttp.proxyPort=3128
```

Configuring the Description webserver

There are two configuration files we must create to configure the Description service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/describe.conf' with the following lines:

```
<VirtualHost *:80>

  ServerName describe.shades.local
  KeepAlive Off
```

```
ProxyPreserveHost On
ProxyRequests Off
ProxyTimeout 14400
```

```
<Proxy balancer://describe_servers>
  BalancerMember http://127.0.0.1:7400
</Proxy>
```

```
ProxyPass / balancer://describe_servers/
ProxyPassReverse / balancer://describe_servers/
```

```
</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/describe.yml' with the following lines:

```
user: daitss
group: daitss
tag: describe
environment: production
pid: /var/run/daitss/thin/describe.pid
log: /var/log/daitss/thin/describe.log
port: 7400
chdir: /opt/web-services/sites/describe
timeout: 300
```

Configuring Transformation service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 2 steps in setting up Transformation service. They are:

- setting up the DAITSS configuration file for Transformation service
- setting up the Transformation service webserver

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines to the end of the file:

```
transform.shades.local:
```

```
    log_syslog_facility:          LOG_LOCAL0
    log_filename:                /var/log/daitss/web-services/
transform.log
```

Configuring the Transformation webserver

There are two configuration files we must create to configure the Transformation service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/transform.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName transform.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://transform_servers>
        BalancerMember http://127.0.0.1:7500
    </Proxy>

    ProxyPass / balancer://transform_servers/
    ProxyPassReverse / balancer://transform_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/transform.yml' with the following lines:

```
user: daitss
group: daitss
tag: transform
environment: production
pid: /var/run/daitss/thin/transform.pid
log: /var/log/daitss/thin/transform.log
port: 7500
chdir: /opt/web-services/sites/transform
timeout: 300
```

Configuring Viruscheck service

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 3 steps in setting up Viruscheck service. They are:

- setting up the DAITSS configuration file for Viruscheck service
- setting up ClamAV
- setting up the Viruscheck service webserver

Setting up DAITSS configuration

Open the file `/opt/web-services/conf.d/daitss-config.yml` and add the following lines to the end of the file:

```
viruscheck.shades.local:  
    log_syslog_facility:      LOG_LOCAL0  
    log_filename:            /var/log/daitss/web-services/  
viruscheck.log
```

Configuring ClamAV

By default, SELinux doesn't allow the ClamAV daemon to scan files owned by other users. One workaround is to set SELinux to permissive mode. As root, open the file `/etc/selinux/config`.

Replace the line

```
SELINUX=enforcing
```

with the line

```
SELINUX=permissive
```

Save and close the file. The changes will take effect on next reboot. To set SELINUX to permissive now, type:

```
[root@shades]# setenforce 0
```

Next, we want to configure the clamd daemon to run as root. To do so, open the file '/etc/clamd.conf' and remove the following line:

```
User clam
```

Configuring the Viruscheck webserver

There are two configuration files we must create to configure the Viruscheck service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/viruscheck.conf' with the following lines:

```
<VirtualHost *:80>

    ServerName viruscheck.shades.local
    KeepAlive Off

    ProxyPreserveHost On
    ProxyRequests Off
    ProxyTimeout 14400

    <Proxy balancer://viruscheck_servers>
        BalancerMember http://127.0.0.1:7600
    </Proxy>

    ProxyPass / balancer://viruscheck_servers/
    ProxyPassReverse / balancer://viruscheck_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/viruscheck.yml' with the following lines:

```
user: daitss
group: daitss
tag: viruscheck
environment: production
pid: /var/run/daitss/thin/viruscheck.pid
log: /var/log/daitss/thin/viruscheck.log
port: 7600
chdir: /opt/web-services/sites/viruscheck
```

```
timeout: 300
```

Configuring XML Resolution service

All of the commands in this section should be run as the daitss user, unless otherwise specified.

There are 2 steps in setting up XML Resolution service. They are:

- setting up the DAITSS configuration file for XML Resolution service
- setting up the XML Resolution service webserver

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines to the end of the file:

```
xmlresolution.shades.local:  
    log_syslog_facility:      LOG_LOCAL0  
    log_filename:            /var/log/daitss/web-services/  
xmlresolution.log  
    data_root:                /var/daitss/xmlresolution  
    resolver_proxy:          localhost:3128
```

Configuring the XML Resolution webserver

There are two configuration files we must create to configure the XML Resolution service webserver, one for the Apache proxy, and another for the thin webserver that actually runs the service.

First, we will configure Apache. Create the file '/opt/web-services/conf.d/xmlresolution.conf' with the following lines:

```
<VirtualHost *:80>  
  
    ServerName xmlresolution.shades.local  
    KeepAlive Off  
  
    ProxyPreserveHost On  
    ProxyRequests Off  
    ProxyTimeout 14400
```

```
<Proxy balancer://xmlresolution_servers>

    BalancerMember http://127.0.0.1:7700
</Proxy>

ProxyPass / balancer://xmlresolution_servers/
ProxyPassReverse / balancer://xmlresolution_servers/

</VirtualHost>
```

Next, lets configure thin. Create the file '/opt/web-services/conf.d/thin/xmlresolution.yml' with the following lines:

```
user: daitss
group: daitss
tag: xmlresolution
environment: production
pid: /var/run/daitss/thin/xmlresolution.pid
log: /var/log/daitss/thin/xmlresolution.log
port: 7700
chdir: /opt/web-services/sites/xmlresolution
timeout: 300
```

Configuring Fixity Checking

All of the commands in this section should be run as the DAITSS user, unless otherwise specified.

There are 2 steps in setting up fixity checking. They are:

- setting up the DAITSS configuration file for fixity checking
- adding entries for the fixity checking scripts to the daitss user crontab

Setting up DAITSS configuration

Open the file '/opt/web-services/conf.d/daitss-config.yml' and add the following lines to the defaults section:

```
fixity_expired_days: 7
fixity_stale_days: 5
```

Now add the following lines to the end of the file:

```

disk-fixity:
  log_syslog_facility:      LOG_LOCAL0
  log_filename:            /var/log/daitss/daemons/disk-
fixity.log
  hostname:                silo.shades.local
  pid_directory:          /var/run/daitss
  fresh_enough:           1

collect-fixities:
  log_syslog_facility:      LOG_LOCAL0
  server_address:          storagemaster.shades.local
  pid_directory:          /var/run/daitss

```

Adding Fixity Checking entries to crontab

To add entries to start the fixity checking scripts to the system crontab, Type:

```
[daitss@shades]% crontab -e
```

In the editor window, add the following lines:

```

1 1 * * * DAITSS_CONFIG=/opt/web-services/conf.d/daitss-
config.yml BUNDLE_GEMFILE=/opt/web-services/sites/silo-pool/
Gemfile bundle exec /opt/web-services/sites/silo-pool/tools/
disk-fixity &> /dev/null
2 2 * * * DAITSS_CONFIG=/opt/web-services/conf.d/daitss-
config.yml BUNDLE_GEMFILE=/opt/web-services/sites/storage-
master/Gemfile bundle exec /opt/web-services/sites/storage-
master/tools/collect-fixities &> /dev/null

```

Exit and save.

Configuring Services to run on startup

Ideally, the DAITSS services should run automatically on system boot. To accomplish this, we must:

- make sure all system services that DAITSS depends on start up on system boot
- install the DAITSS rc script and configure it to start on boot

Configuring DAITSS dependencies to start on boot

Type:

```
[root@shades]# ntsysv
```

A GUI will display. In the GUI, ensure that postgresql, clamd, httpd, squid, and rsyslog are selected to run on startup.

Postgres is a special case. We need to set it so that it starts along with DAITSS. To do so, type:

```
[root@shades]# chkconfig --level 3 postgresql on
[root@shades]# chkconfig --level 4 postgresql on
[root@shades]# chkconfig --level 5 postgresql on
```

Installing and configuring the DAITSS rc script

The first step is to copy the DAITSS rc script into '/etc/init.d:'

```
[root@shades]# cp /opt/web-services/sites/core/bin/init.d/oss/
daitss /etc/init.d
```

Now, lets configure the system to run the RC script on boot. Type:

```
[root@shades]# cd /etc/init.d
[root@shades]# chkconfig --add daitss
[root@shades]# chkconfig --level 3 daitss on
[root@shades]# chkconfig --level 4 daitss on
[root@shades]# chkconfig --level 5 daitss on
```

Lastly, lets add sudo privileges to the daitss user so that the daitss user can start and stop DAITSS via sudo. Open the file '/etc/sudoers' and add the following lines:

```
daitss  ALL=(ALL)      ALL
```

That's it! To conclude the installation, just reboot the system.