TACTIC System-Admin Documentation

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Planning **1**

1.1 TACTIC, a Scalable Solution . . . 1

1.2 Scalable TACTIC Deployment Planning . . . 3

2 Install TACTIC Application **6**

2.1 TACTIC System Requirements . . . 6

2.2 General TACTIC Install . . . 12

2.3 How to Install TACTIC on CentOS 5.4 . . . 18

2.4 TACTIC Install - CentOS 5.5 . . . 21

2.5 TACTIC Install - CentOS 5.7 . . . 24

2.6 TACTIC Install - CentOS 6.2 . . . 34

2.7 TACTIC Install - Windows Server 2008 . . . 41

2.8 Install TACTIC - Fedora 8 . . . 48

2.9 How to Install TACTIC on Fedora 12 . . . 48

2.10 How to Install TACTIC on Ubuntu 12 . . . 50

2.11 Upgrade TACTIC . . . 51

2.12 Multiple TACTIC Services . . . 52

2.13 Install a TACTIC License . . . 53

3 Install Python **54**

3.1 Python Installation . . . 54

4 Install Web Server **55**

4.1 Apache HTTP Co-Service Configuration . . . 55

4.2 IIS 7+ HTTP Co-Service Configuration . . . 57

5 Install Database **69**

5.1 TACTIC Database Configuration . . . 69

6 Server Configuration **69**

6.1 TACTIC Configuration File . . . 69

6.2 Configure the TACTIC Service . . . 75

6.3 TACTIC Configuring SSL . . . 75

6.4 Active Directory Integration . . . 78

6.5 Configuring TACTIC and Co-Services . . . 80

6.6 Setup Email . . . 82

6.7 HTTP Co-Service Installation/Configuration . . . 83

6.8 Database Resource . . . 84

6.9 RemoteAccessTactic . . . 85

7 Fileserver **85**

7.1 Relocating the *assets* directory . . . 85

7.2 Configure the Remote Repo . . . 88

8 Scalability **88**

8.1 TACTIC Scalable Configuration Examples . . . 88

8.2 Load-Balancing TACTIC . . . 89

9 Troubleshooting **90**

9.1 TACTIC Service Troubleshooting . . . 90

9.2 Monitor Server Performance . . . 92

10 Maintenance **92**

10.1 TACTIC Backup and Restore . . . 92

10.2 Database Backup Automation . . . 94

10.3 TACTIC logs . . . 96

11 TACTIC Server VM **97**

11.1 TACTIC VM Setup and Install . . . 97

11.2 TACTIC VM Troubleshooting . . . 101

11.3 Moving VM Data . . . 102

12 Client Setup **102**

12.1 Client Computer Setup Requirements . . . 102

This document covers various topics like Planning, Installation, Maintenance, and Client set-up.

1 Planning

1.1 TACTIC, a Scalable Solution

A data management system to grow along with your expanding data needs TACTIC software meets the complex content demands of today's large-scale production environment:

- Based on existing web technologies, it is easy to maintain and scale
- Easy to load-balance and scale across cores and across multiple machines

What Makes TACTIC Scalable?

The TACTIC platform uses web technologies in its underlying architecture. As proven by many popular web sites today, web technologies are able to scale to a large number of users—to levels far beyond the size of any large installation today requiring TACTIC.

The TACTIC user interface and its API (Application Programmer Interface) are both built on web technologies. A web server controls as the user front end., and accesses a back-end database. Web servers can handle high volumes, so scalability is not an issue—especially for the relatively manageable volume seen in even the largest of digital productions.

Web server technology

TACTIC uses a web server to deliver static content to users, who are able to access TACTIC over the web. Web servers are capable of delivering enormous numbers of requests, and will remain stable during heavy loads. Web servers by definition are the most efficient vehicles for web delivery, and TACTIC technology is optimized to take advantage of this technology for fast execution and scalability.

Leading database technology

TACTIC stores all persistent data in either Oracle or PostgreSQL databases, products that both have proven track records in the most demanding of production environments.

Today's databases coupled with today's hardware are sufficiently powerful to handle any of today's production needs. The largest of digital productions cannot come close to exceeding the data limits of a typical database.

Optimized requests

With digital productions, the amount of data required is seldom the issue. Rather, bottlenecks are typically caused by complex requests slowing down database response time. Digital productions tend to use deeply nested and enormously complex relationships between various pieces of data. It is this complexity that tends to slow down applications that try to parse this data in a meaningful human way, creating complex requests that slow down database response.

TACTIC is designed from the ground up to master the complexities of digital productions. For every request, it builds a unique object-relational mapper specifically designed for the needs of high volume, large scale digital production asset management solutions.

TACTIC is flexible enough to handle the complex relationships between any part of the production pipeline, and has proven itself in large-scale projects using extremely complex data.

Easy code maintenance

The Python programming language is used for TACTIC application development. Python has many strengths that contribute to ease of maintenance (courtesy of www.python.org):

- clear, readable syntax
- strong introspection capabilities
- intuitive object orientation

- natural expression of procedural code
- full modularity supporting hierarchical packages
- exception-based error handling
- very high-level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython)
- embeddable within applications as a scripting interface

Load balancing

Load balancing the distribution of requests across TACTIC processes is possible across cores and across multiple machines, and enables TACTIC to scale to large enterprise environments.

(For details and examples, see **TACTIC Scalable Configuration Examples**.)

Segregating services

The various TACTIC services can be segregated to run independently of each other. This allows important services, such as user interface requests, to access their own resources and not be slowed down by other more resource-intensive services such as render farms.

(For details and examples, see **Scalable TACTIC Deployments**.)

Types of Scalability

There are several types of scalability that are considerations with the use of Python as the TACTIC programming language:

- Code scalability: TACTIC contains thousands of classes and many hundreds of thousands of lines of code which are facilitated by use of the Python language. There has been no reason to believe that the code base cannot continue to scale to meet software needs for the foreseeable future.
- Memory scalability: Python is known for large memory requirements when compared to other languages, but the required memory for TACTIC code is much smaller than that found on typical servers, and has seldom been an issue.
- Performance scalability: Python's GIL (Global Interpreter Lock) limits Python's scalability across multiple threads. This limitation has been long discussed in newsgroups, with the general consensus that scalability is achieved much more easily by load-balancing across processes rather than across threads. TACTIC enables easy load balancing across processes to achieve full linear scalability.
- Maintenance scalability: Clean and clear syntax makes it easy to refactor and maintain code over the long term. Experience has shown that our code base is very accessible to new developers, providing confidence that code base knowledge can be maintained over the years.

Overall, very few limitations with the Python language itself have been encountered and have easily circumvented its lack of scalability across multiple threads by optimizing proper load balancing across processes.

Additional Scalability Considerations

Availability

TACTIC has been designed to be highly stable even under extremely heavy stress. It has been tested and tuned on several projects in environments of high stress and high load and has been proven to run under duress without going down.

As with any piece of software, there is always a point a failure—there are a number of legitimate reasons for a TACTIC server becoming unresponsive or even going down:

- network cutoff
- power failure

- physical memory failure
- overloaded requests for the available resources

For projects that require high availability even in the event of catastrophic failure, there are a number of hardware solutions to provide failover should the primary resource fail. These hardware solutions will instantly redirect requests to a redundant resource, thereby providing continuous service to the users.

Memory requirements

Each available process will use a certain amount of memory, so a balance must be struck between the number of processes and the size of the complete in-memory TACTIC footprint on the server. In other words, adding more processes may or may not increase the availability of TACTIC if memory is an issue. If processes start running into swap space, performance will be severely affected.

As memory usage becomes an issue, the choice will be to load balance over multiple machines (see **Scalable TACTIC Deployments**).

Database Replication

Both Oracle and Postgres have replication capability: the ability to distribute the processing load of the database server over a number of machines. Database replication is complex and should not be taken lightly. With the advent of multiprocessor and multi-cores systems, the need for database replication has been reduced significantly. It is often much cheaper and less time consuming to increase the processors/cores of a server than it is to configure a database for replication.

1.2 Scalable TACTIC Deployment Planning

The stages of a TACTIC deployment are:

- 1. Determining usage requirements
- 2. Determining hardware requirements
- 3. Determining TACTIC Service and Co-Service Locations
- 4. Post Deployment TACTIC upscaling

Planning NOTES

Determining TACTIC Usage Requirements

One of the first questions asked in a TACTIC deployment is "what kind of deployment is required?" This is a hard question to answer unless it is known beforehand exactly how TACTIC will be used. Since every TACTIC deployment is customized by the licensee, there is really no way to calculate accurately the type of load that will eventually be placed on the TACTIC server. Fortunately, due to TACTIC's scalable architecture, a TACTIC deployment can be scaled up easily when there are unforeseen increases in usage.

Service Requirements

In a TACTIC deployment, running the TACTIC service is the primary function of the TACTIC host machine. The co-services can be deployed to the location most appropriate for the environment. However, to ensure scalability and high availability, the co-services should not be deployed on the TACTIC host for any but the smallest of deployments.

TACTIC uses four major services and co-services in its operation:

- TACTIC service: provides the API and GUI widget system.
- Database co-service: provides indexing and storage for asset meta data.
- HTTP co-service: provides delivery (and return) of raw asset data.
- Assets storage co-service: provides storage and back-up capabilities of raw asset data.

Limitations

With TACTIC, you can simply add hardware and load-balance it to handle more and more requests. You can expand scalability in this way until the database server itself becomes the bottleneck. Fortunately, with the introduction of multi-CPU and multi-core machines, database limits are extremely high.

User Interfaces: For applications with end-user interfaces (especially those using widgets, where complex interface elements need to be constructed), the load required to process TACTIC requests will remain much greater than the load required to process SQL calls, no matter how much hardware is added to handle such requests.

Client APIs: Client APIs provide a much thinner layer to the database and so are optimized to handle lower level requests very rapidly. However, because API calls are typically used for automated processes, they can generate a large number of requests quickly, creating a heavy load on the TACTIC server.

Load balancing: this is especially important for render farms, which can be massive in scale with thousands to tens of thousands of cores. It ensures that the high volume of requests is evenly distributed to all of the available TACTIC processes. The exact number of processes required to handle the high volume is highly dependent on the number of requests made and to the complexity of the requests. However, because load balancing will provide near-linear scalability, the number of processes can be increased to a level that is sufficient for the number of requests being demanded of the client API.

Report generation: TACTIC's reporting system is powerful and comprehensive, and the load on host CPUs can increase significantly when complex reports are being run. Administrators should focus on report generation when investigating load utilization.

Checkins: There are two kinds of complexity when considering scalability for checkins:

- Large Checkins: TACTIC offers a wide variety of checkin configurations that will accommodate large bandwidth, large files, and complex dependency between files.
- Many Checkins: There are few concerns with a large number of checkins. The number of checkins that occur even in the largest of productions is small compared to the limits of a database.

Determining TACTIC Hardware Requirements

Hardware Overview

TACTIC works well with most types of hardware, but industry-standard server level components are highly recommended. The exact configuration will depend on the specific needs of your facility.

For most deployments, TACTIC can be installed on the typical commodity hardware used by most small and large enterprises. TACTIC is a purpose-built relational database at its core, and your hardware choices can reflect that fact.

TACTIC service requirements: Assuming TACTIC is the only service on the host, the minimal requirements for a production-level deployment of a TACTIC host are:

- CPU: dual core
- Memory: 4G
- Disk space needed for TACTIC to run on: 10G

  (assets should be stored elsewhere, on an enterprise quality server)

These requirements represent the recommended absolute minimal hardware required to run a basic TACTIC service in a production environment. For development and testing, TACTIC will run on most hardware including laptops.

Asset storage: It is highly recommended that assets be stored on an enterprise-quality, high-availability server, and not on the TACTIC host.

RAID Options: You can use various RAID configurations to maximize reliability and performance. For example, you could locate the Operating System on one physical drive, and the database files on another. Or you could locate the database on a RAID array.

Asset meta data: The amount of asset meta data stored on disk by TACTIC is insignificant to today's hardware specifications and need not be a consideration when choosing a server: reliability and performance should be higher concerns.

Determining TACTIC Service Locations

Segregating Services

TACTIC offers a number of different services for different purposes. For example, the TACTIC server can serve out complex widgets, serve out large numbers of client API requests and perform large, complex check-ins.

Each type of service has distinct needs, so it is often beneficial to segregate the services to be independent of each other. This allows each service to operate within a pool of assigned resources without affecting (or being affected by) the other services.

For example, if a render farm used a client API script that heavily loaded the TACTIC server and its requests were shared with user interface requests, users would experience inconsistent (and occasionally unacceptably slow) response times. This experience would cause frustration amongst users, so it would be necessary to segregate their user interface service from the render farm's client API service.

Each type of service has very different demands and needs, so segregating services also allows you to tune the hardware and software to the specifications required for the service they are handling. Ultimately, you can maximize the use of your available resources.

How Load Balancing Works

Load balancing distributes requests to various TACTIC processes using an algorithm that determines which request is delegated to which process. The default algorithm is a very simple randomizer.

For most applications, the default algorithm works perfectly fine. It is left to chance whether a particular request will be assigned to a heavily loaded process or one that is sitting idle, so there may be inconsistent performance depending on load and availability of TACTIC processes not already loaded.

IIS delegation algorithms (on MSWeb Server) are also supported. For Apache, an example algorithm is provided you can customize to your needs.

Each process, by default, is assigned two simultaneous requests at most. This limit has proven in production to be the most effective because it prevents the Python process itself from being overloaded waiting for required I/O operations. If there are excess requests waiting to be processed, they are kept in a queue and assigned sequentially to processes as they become available.

Python's GIL (global interpreter lock) prevents any Python process from using more than one CPU or core, so the number of threads available for any given process is limited. True scalability is achieved by load-balancing requests over multiple processes. This method scales seamlessly and linearly over multiple cores, multiple CPUs and ultimately over multiple physical servers with only simple configuration changes.

Post Deployment TACTIC Upscaling

Scenarios

TACTIC is easily upscaled in most post deployment scenarios. Upscaling does not require any significant downtime of TACTIC services.

Some scenario examples follow, with bullet points outlining why these possible solutions would be followed;

Increasing number of TACTIC processes on a single machine

Splitting TACTIC into multiple servers from one server

- Easy way to upscale/troubleshoot
- Heavy reporting
- Many users/API calls from automated processes

Splitting the HTTP co-service into multiple servers from one server

- Many Checkins
- Large Checkins

2 Install TACTIC Application

2.1 TACTIC System Requirements

Client Computer Requirements

Supported Operating Systems

- Windows
- MAC OS
- Linux
- Smartphones
- Tablets

Note - The main criteria in the majority of cases for TACTIC access is the web browser. TACTIC is a cross platform solution.

Supported Browsers

- Chrome (Officially Supported)
- Safari (Compatible)
- Mozilla Firefox 33.0+ (Compatible)
- Internet Explorer 11.0+ (compatible for everyday usage - HTML5 support is limited on IE)

Note - Other web browsers may work but are not certified for TACTIC.

Required Browser Plugins

We have moved away from needing the Java plugin in our latest release, and instead, have moved on to using HTML5 in its place. This was done to remove external dependencies and create a better user experience. However, depending on the version of TACTIC that you use, the Java plugin may be used in many cases to communicate with the client computer's operating system. This plugin is used primarily in application integration to pass files through the client web browser to and from the TACTIC server. To download the latest version of the Java plugin, go to http://www.java.com

Server Requirements

Typical TACTIC Server Configurations

Small Shop: Less than 10 Users Medium Content Creation: 10 - 50 Users Large Scale Content Creation: 50 - 150 Users Enterprise Business: More than 150 users High Availability: High service availability (99.9% Up-time guarantee) Remote Collaboration: Live World Wide Production

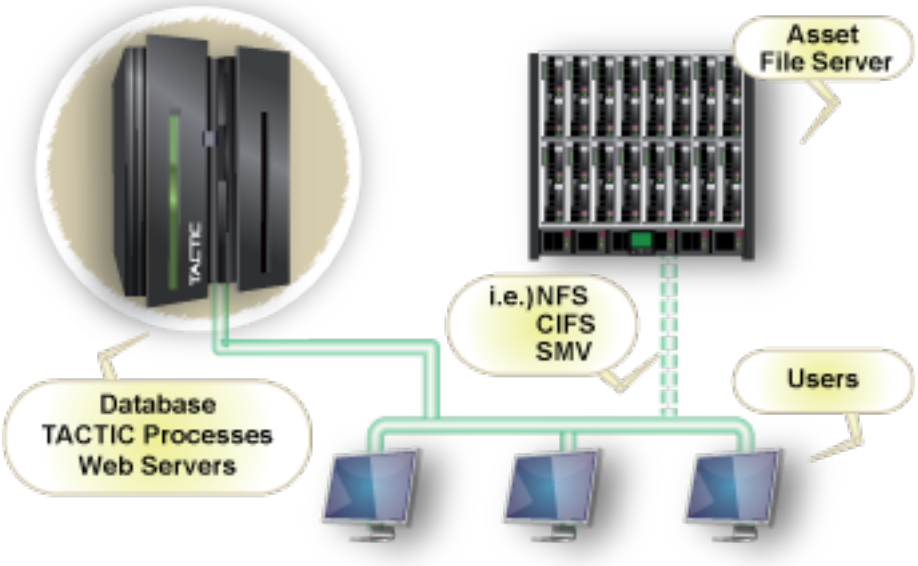
Small Shop (< 10 Users)

A small cloud business might need to track lists of information, tasks, notes and general production data. Simple files can also be tracked. TACTIC Server

- Single computer configuration with all the services installed on the same machine.
- Online cloud service or in-house installation
- CPU - Dual Core processor
- RAM - At least 600MB ram
- HDD - 10GB (+ asset storage requirements)
- 3 TACTIC Process for load balancing

Medium Content Creation (< 50 Users)

Dual server computer configuration with TACTIC. Database and web server on one computer and a separate fileserver. Examples include content creation departments, visual effects studios, media management solutions.



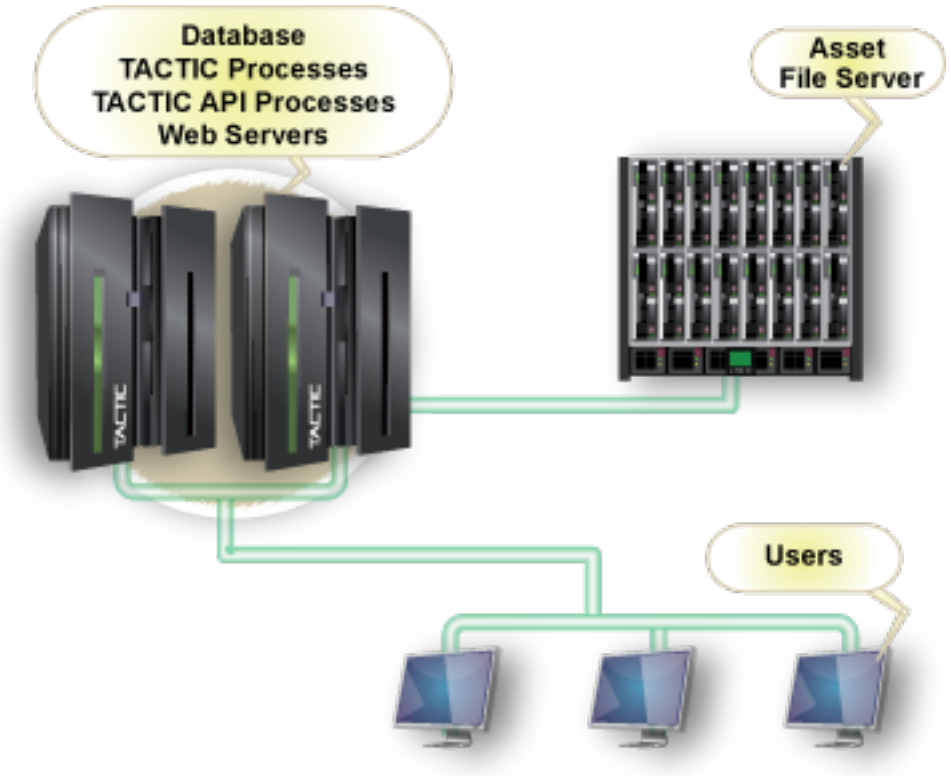
TACTIC Server

- CPU - Dual or Quad core processor
- RAM - 4-8GB ram
- HDD - 10GB
- 8 TACTIC Process for load balancing

File Server SAN or Fileserver with adequate storage based on project requirements. Note that TACTIC generates a more efficient file system so space is used more efficiently.

Large Scale Content Creation (< 150 Users)

Represented by companies who have a 100+ team producing and managing complex assets and production data. Examples include large content creation departments, feature film and episodic CG production.



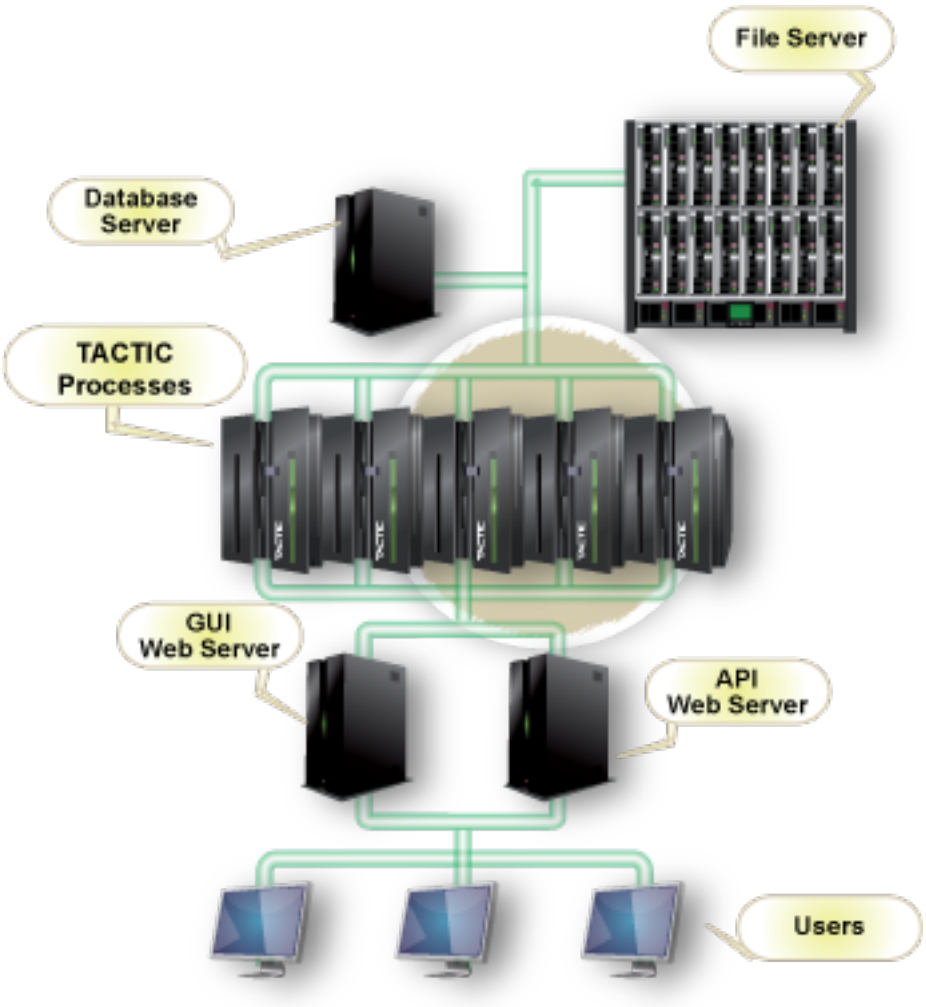
TACTIC Server

- 1-2 Servers CPU - Quad core processor
- RAM - 8GB+ ram
- HDD - 10GB
- 3 TACTIC Process for load balancing

File Server SAN or Fileserver with adequate storage based on project requirements. Note that TACTIC generates a more efficient file system so space is used more efficiently.

Enterprise Business (150+ Users)

In large scale enterprise scenarios, the scalability features in TACTIC are heavily used to accommodate the high bandwidth demands of enterprise business.



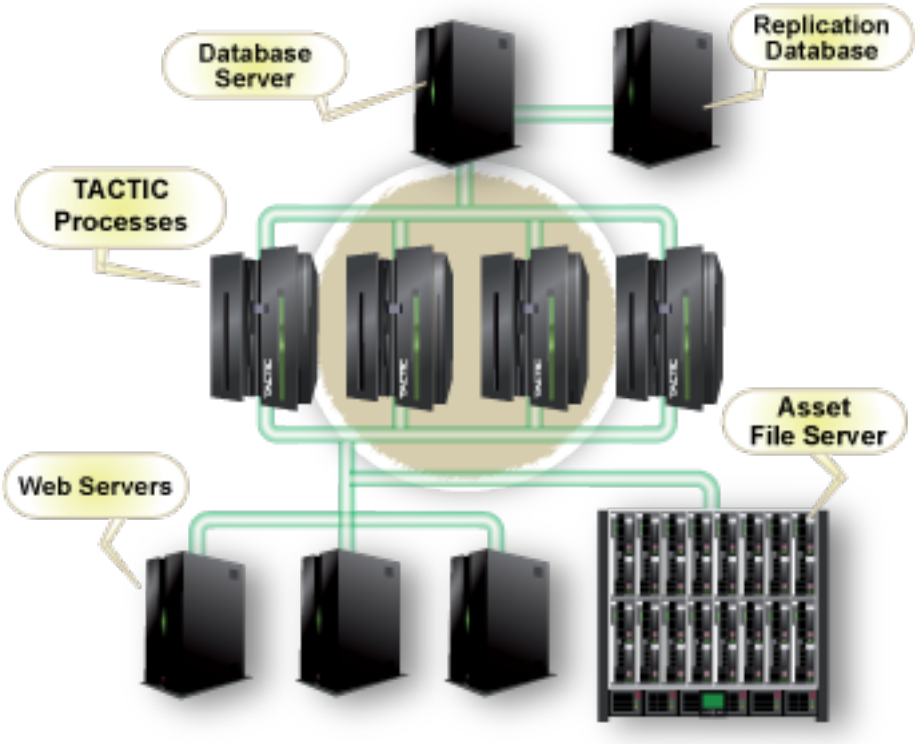
TACTIC Server

- Multiple Servers
- CPU - Quad core processor
- RAM - 8GB+ ram
- HDD - 10GB
- 10+ TACTIC Process for load balancing on each server

File Server SAN or File server with adequate storage based on project requirements. Note that TACTIC generates a more efficient file-system so space is used more efficiently. Database Server Single database server allowing for multiple physical TACTIC servers.

High Availability - Up-time Guarantee

In almost any size business, there may be a requirement for TACTIC to be a high availability service. The main aspect of



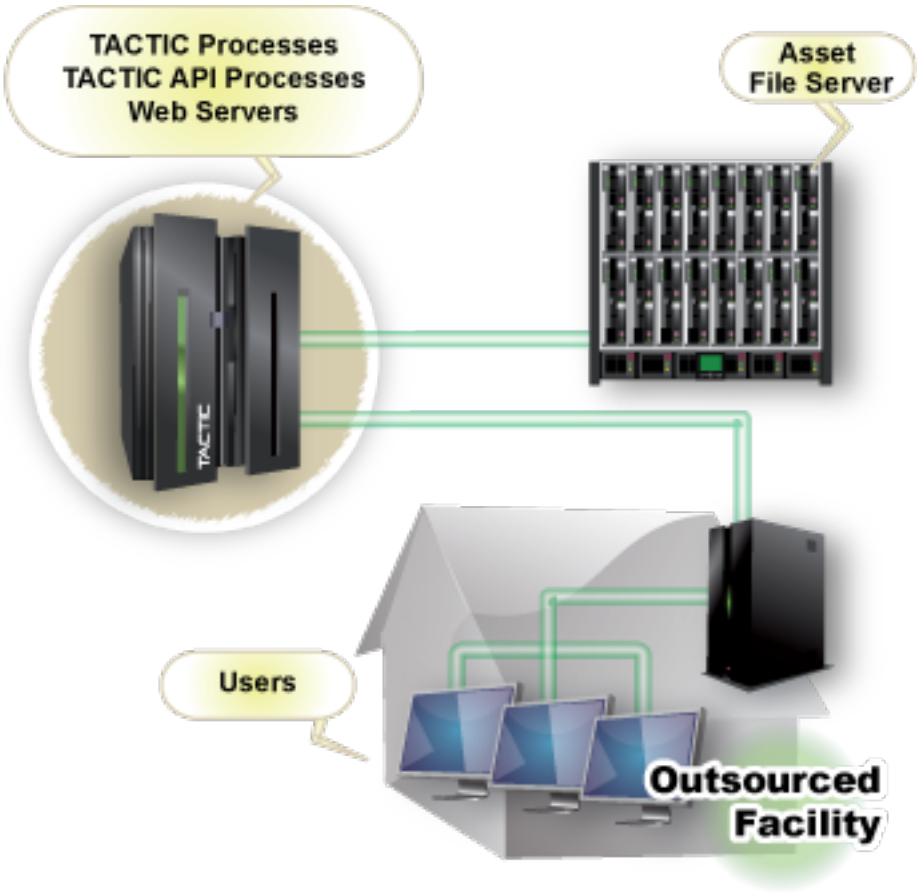
configuring TACTIC with this approach is providing redundant services at all levels. TACTIC Server 2+ Physical TACTIC servers with redundant power, HDD etc.

- CPU - Quad core processor
- RAM - 8GB+ ram
- HDD - 10GB
- 8+ TACTIC Process for load balancing per server

File Server SAN or File server with adequate storage based on project requirements. Redundant data retention for file system up-time. Database Server 2+ database servers setup with replication for database up-time. Return to section top

Remote Collaboration (World wide production)

TACTIC is a web based enterprise application which can be scaled out to provide seamless collaboration between physical locations using common database and file system technology and practices.



TACTIC Server

- Master Physical TACTIC server
- CPU - Quad core processor

- RAM - 4GB+ ram
- HDD - 10GB
- 8+ TACTIC Process for load balancing per server
- Remote collaboration server(s)
- CPU - Quad core processor

- RAM - 4GB+ ram
- HDD - 10GB
- 3+ TACTIC Process for load balancing

File Server

- Master Fileserver
- Base central fileserver
- Remote File Server(s)

- Synchronized through common protocols or through specific processes/triggers in TACTIC

TACTIC Server Installation Requirements

Supported Operating Systems

- Fedora 12
- CentOS 5.5
- Windows Server 2008
- TACTIC has been set-up on a wide variety of operating systems. Our documentation provides detailed instructions on the above, however, it is left to the user to install on others.

API

- Python 2.5-2.7
- Javascript (Client side Interaction)

Web Servers

- HTTP
- Apache 2.2+
- IIS 7+

Database

- TACTIC Database Configuration

Installation Process Requirements

What sort of access do you need to complete a full TACTIC installation and implementation?

- A physical of virtual server dedicated to TACTIC
- Root Access to the server
- Knowledge of the current network infrastructure
- Ability to install Modules etc
- Knowledge of Python is a plus

*More information about TACTIC installation can be found in TACTIC installation manuals. Please contact sales for further information or sign up today for our community site - Sign up today!

2.2 General TACTIC Install

This document guides you through the general installation of TACTIC in Fedora Linux with some brief background information. For installation on Windows or CentOS, please refer to the corresponding sections in this book. To completely install TACTIC, there are three main components that have to be set up in the following order:

1. A Database
2. The TACTIC Installer

3. A Web Server

TACTIC stores all metadata in a database called PostgreSQL. This is an industrial strength, hugely scalable database that has proven itself in thousands of industries around the world. We currently recommend 8.4 and upwards for ease of installation. Go to the PostgreSQL website[www.postgresql.org] for more information.

1. rpm -Uvh http://yum.pgrpms.org/reporpms/8.4/pgdg-fedora-8.4-2.noarch.rpm
2. yum install postgresql postgresql-server postgresql-contrib postgresql-devel
3. It is located in /var/lib/pgsql/data This file determines all of the user permissions for PostgreSQL. To begin with, turn on all of the permissions contained in this file. This is a temporary measure that will greatly simplify the installation process. You may lock down these permissions at a later date. Please consult the PostgreSQL documentation on how to do this. TACTIC ships with a sample "pg_hba.conf" file (located in <TACTIC_unzipped_ package>/src/install/postgresql/pg_hba.conf). This file has an open security setting for ease of installation. It's best to back up your current pg_hba.conf file before copying over with the file provided by TACTIC.
4. Restart the PostgreSQL Service

Verification

Verify that psql works in the command prompt:

```
> psql -U postgres template1
```

It should give you a prompt:

```
template1=#
```

If you see this prompt without the need to enter a password, you have successfully installed the PostgreSQL database. Type \q to exit.

Install Python and supporting modules

The TACTIC source code is written in Python. As such, the complete codebase is open.

To install Python, you can find an msi at http://www.python.org. TACTIC requires Python 2.6 or higher.

TACTIC also requires a number of Python modules to function correctly. These modules are generally not installed by default with the standard Python distributions. For Windows, they are already packaged in a file python_modules.zip available for download in the release section of the downloads page at http://support.southpawtech.com/downloads

- `PIL 1.1.7` (Python Image Library)
- `ImageMagick 6.7.8` (A command line program which complements PIL) - http://www.imagemagick.org
- `FFmpeg version 0.6` (Solution to record, convert and stream audio and video and metadata parsing) - http://www.ffmpeg.org
- `psycopg2 2.3.1` (Database connectivity)
- `PyCrypto version 2.0.1/2.1` (For de-encrypting the license file) - http://www.amk.ca/python/code/crypto
- `simplejson 2.1.1` (not needed since Python 2.6) simplejson.egg can be installed with the EasyInstaller. It is needed for JSON string encoding and parsing
- `lxml 2.1 or 2.2` (XML and XPath processing)

Unzip the file in a temporary location.

```
# cd /tmp
# unzip tactic_#.#.#.#.zip
```

Go to /tmp/tactic_#../src/install/

```
# cd /tmp/tactic_#.#.#.#/src/install
```

Execute: install.py

> **Note**
>
> You must invoke the installation with root user privileges because it attempts to write to the <python_install>/site-packages directory.

```
# su
# python install.py
```

The installer will ask a number of questions. First it ask for the <TACTIC_BASE_DIR>:

```
Please enter the base path of the TACTIC installation:

(/home/apache) ->
```

Enter the user the Apache Web Server is run under.

```
Please enter the user Apache Web Server is run under:

(apache) ->
```

It would copy the source code to the base path and create a symbolic link to it. An Apache Web Server file will be generated at the end which you would need to copy to the Apache config area upon installation of the web server. If there are existing files in the destination directory the installer will ask for your confirmation to remove it. At the end, you will see this:

```
*** Installation of TACTIC completed at [/home/apache] ***


Next, please install the Apache Web Server and then copy the Apache config extension
[/home/apache/tactic_data/config/tactic.conf] to the Apache web server config area. e.g.
/etc/httpd/conf.d/
```

Verification

When the installation is completed, an asset directory will be created at *<TACTIC_DATA_DIR>/assets*.

The file tactic_paths.py will be created in the following directories:

Linux

```
/usr/lib/python#.#/site-packages/tacticenv
```

Default contents:

```
TACTIC_INSTALL_DIR='/home/apache/tactic'
TACTIC_SITE_DIR=''
TACTIC_DATA_DIR='/home/apache/tactic_data'
```

Network machine access

If you are on the server, you can access it by using the URL http://localhost/tactic. For other people to access it on the network, you need to find out its IP address. In a linux server, you can use the command "ifconfig" to locate it. It's the one listed as the inet addr.

Next, you need to run TACTIC behind Apache.

TACTIC should be run behind an Apache web server. You can download Apache software at http://www.apache.org/

The TACTIC application server is able to serve up static content such as images, PDF files, Quicktime files, and so on, but it is not the most efficient at this because it is written in Python. This is what Apache is designed for. By running TACTIC behind Apache, it relieves TACTIC from serving the static content so that it can focus on the dynamic content.

For production use, it is highly recommended that TACTIC is run behind the Apache server. This has many scalability advantages. When running behind Apache, Apache uses a reverse proxy and proxy balancer module to forward requests to communicate with TACTIC.

Linux

After the installation, some changes may need to be made in the "httpd.conf" file for Apache.

```
Fedora Core: /etc/httpd/conf/httpd.conf
```

Make sure the following lines are uncommented:

```
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule deflate_module modules/mod_deflate.so
```

Additional lines For Apache 2.4:

```
LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
LoadModule filter_module modules/mod_filter.so
LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
```

These lines may already be uncommented, depending on your distribution and version of Apache. You need Apache version 2.0.31 or later.

The TACTIC installer generates an Apache extension conf file on completion. Copy that file to a directory that is recognized by Apache.

For Fedora Core: copy the TACTIC generate Apache conf extension file to the following directory:

```
/etc/httpd/conf.d/
```

If there is no such configuration extension folder, you must add the following line to the "httpd.conf" file so that it will read the extension configuration file provided by the installer:

```
Include conf/tactic_win32.conf
```

Web Server configuration

In our provided file:

```
tactic.conf
```

Make sure the following lines exist:

```
# Using the ProxyPass directives
ProxyPreserveHost on

<Proxy balancer://tactic>
BalancerMember http://localhost:8081/tactic
BalancerMember http://localhost:8082/tactic
BalancerMember http://localhost:8083/tactic
</Proxy>
ProxyPass /tactic balancer://tactic
ProxyPass /tactic_data balancer://tactic
```

For Apache 2.4:

```
Comment out:
#Order Allow,Deny
#Allow from All

Uncomment:
Require all granted
```

Note

Warning: For load balancing, only use either:

1. the Proxy Balancer method (recommended)

   or

2. the RewriteRule method (not recommended).

Do **not** use both methods at the same time.

Note

For configuring load-balancing set-up in a real production, please refer to the Load Balancing section in the Sys-admin docs.

Note

When trying to set up Apache on a Windows Server, to specify a share folder for Apache to use, you may need to include the name of the share folder in the path.

For example, use the following if you have named the share folder "my_share":

```
Alias /assets "//10.0.0.17/my_share/assets"
```

Finally, after verifying the configuration is correct, restart the Apache service:

```
service httpd restart
```

Go to the "<TACTIC_INSTALL_DIR>/src/bin" folder

Then su as the user tactic and run startup_dev.py:

```
cd /home/apache/tactic/src/bin

su tactic

python startup_dev.py
```

This "startup_dev.py" script is the development script which will dump output to the screen. The other startup script "startup.py" is the production start-up script and will dump output to a log file. The development start-up script is also much slower as it monitors the file system to see if any files have changed.

The output would look like the following:

```
Registering site ... admin
Registering site ... default
Registering site ... test
Registering site ... my_project

Starting TACTIC ...

05/Jul/2007:11:16:29 CONFIG INFO Server parameters:
05/Jul/2007:11:16:29 CONFIG INFO server.environment: development
```

```
05/Jul/2007:11:16:29 CONFIG INFO server.log_to_screen: True
05/Jul/2007:11:16:29 CONFIG INFO server.log_file: D:/tactic_temp/log/tactic_log
05/Jul/2007:11:16:29 CONFIG INFO server.log_tracebacks: True
05/Jul/2007:11:16:29 CONFIG INFO server.log_request_headers: True
05/Jul/2007:11:16:29 CONFIG INFO server.protocol_version: HTTP/1.0
05/Jul/2007:11:16:29 CONFIG INFO server.socket_host:
05/Jul/2007:11:16:29 CONFIG INFO server.socket_port: 8081
05/Jul/2007:11:16:29 CONFIG INFO server.socket_file:
05/Jul/2007:11:16:29 CONFIG INFO server.reverse_dns: False
05/Jul/2007:11:16:29 CONFIG INFO server.socket_queue_size: 10
05/Jul/2007:11:16:29 CONFIG INFO server.thread_pool: 5
05/Jul/2007:11:16:30 HTTP INFO Serving HTTP on http://localhost:8081/
```

`http://<TACTIC_server_address>/tactic/admin/`

You should see the TACTIC login appear.

There is a default user created on installation. This is the "admin" user and this user has the ability to see and change all aspects of the system. Log in as the admin user:

> **Note**
>
> You may be asked to change your password automatically at startup
>
> without entering these default credentials.

user:	`admin`
password:	`tactic`

> **Note**
>
> If you have not set up the TACTIC service, refer to the page "Configure the TACTIC Service".

First, stop TACTIC running in dev mode if applicable by pressing Ctrl ^ C in that shell

In Linux:

```
service tactic start
```

At this point you will need to install a TACTIC license file and then begin to set up a project.

For more information on installing the license file, please refer to the Install License File chapter.

For more information on getting started with projects, please refer to Setup documentation.

2.3 How to Install TACTIC on CentOS 5.4

Below are the step-by-step instructions of how to install TACTIC on CentOS 5.4.

CentOS 5.4

Unfortunately, CentOS 5.4 comes pre-packaged with an outdated version of Python: Python 2.4. TACTIC requires Python 2.7.

To get around this, when we install Python 2.7, we do not overwrite the original Python 2.4. See steps #8 and #9.

Overwritting the original Python would break some software packages, such as Yum.

We must remember to call Python2.7 explicitly whenever we run any TACTIC python scripts, including startup.py

1. Log in as the root user.
2. Open the passwd file.

```
vi /etc/passwd
```

Modify the apache home directory and login shell to look like the following:

```
apache:x:48:48:Apache:/home/apache:/bin/bash
```

1. Set password for apache to: south123paw

```
passwd apache
```

1. Open to the sudoers file.

```
visudo
```

Add the apache user by including the following line in the appropriate location in the file:

```
apache ALL=(ALL) ALL
```

1. Create the home directory for apache.

```
mkdir /home/apache
chown apache:apache /home/apache
chmod a rx /home/apache
```

1. Disable SElinux.

```
vi /etc/selinux/config
```

1. Install modules.

```
yum install -y gcc zlib-devel samba libxslt-devel libxml2-devel postgresql-server  ↩
    postgresql-devel
```

1. Install Python 2.7.

```
cd; wget http://www.python.org/ftp/python/2.7.1/Python-2.7.1.tgz
tar zxvf Python-2.7.1.tgz
cd Python-2.7.1
./configure -with-zlib=/usr/include
make install
```

1. Installing Python 2.7 will install python2.7 command "ahead" of the original python so that "python" from the command line is now 2.7. This will break yum. To fix this, remove the python executable for 2.7.

```
rm /usr/local/bin/python
```

1. Restart the shell so that the environment is clean.
2. Install lxml.

```
cd; wget http://codespeak.net/lxml/lxml-2.2.8.tgz
tar zxvf lxml-2.2.8.tgz
cd lxml-2.2.8
python2.7 setup.py install
```

1. Install PIL.

```
cd; wget http://effbot.org/downloads/Imaging-1.1.7.tar.gz
tar zxvf Imaging-1.1.7.tar.gz
cd Imaging-1.1.7
python2.7 setup.py install
```

1. Install psycopg2.

```
cd; wget http://pypi.python.org/packages/source/p/psycopg2/psycopg2-2.3.2.tar.gz
tar zxvf psycopg2-2.3.2.tar.gz
cd psycopg2-2.3.2
python2.7 setup.py install
```

1. Install PyCrypto.

```
cd; wget http://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.3.tar.gz
tar zxvf pycrypto-2.3.tar.gz
cd pycrypto-2.3
python2.7 setup.py install
```

1. Disable firewall.

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

1. Create and open the index.html for redirection.

```
vi /var/www/html/index.html
```

Insert the following contents:

```
<META http-equiv="refresh" content="0;URL=/tactic">
```

1. Re-login as the apache user.
2. Download the TACTIC source code and setup the service.

Open the following link in a web browser

```
http://support.southpawtech.com/download
```

Setup the TACTIC service.

```
cd /tmp
unzip tactic_#.#.#.#.zip
sudo cp /tmp/tactic_#.#.#.#/src/install/service/tactic /etc/init.d
sudo chmod 775 /etc/init.d/tactic
sudo /sbin/chkconfig tactic on
```

1. Setup Postgres.

```
sudo /etc/init.d/postgresql start
sudo cp /tmp/tactic_#.#.#.#/src/install/postgresql/pg_hba.conf /var/lib/pgsql/data
sudo chown postgres:postgres /var/lib/pgsql/data/pg_hba.conf
sudo /sbin/chkconfig postgresql on
sudo /etc/init.d/postgresql restart
```

1. Setup Apache.

```
sudo cp /home/apache/tactic_data/config/tactic.conf /etc/httpd/conf.d/
sudo /sbin/chkconfig httpd on
sudo /etc/init.d/httpd start
```

1. Install TACTIC.

```
cd /tmp/tactic_#.#.#.#/src/install
sudo python2.7 install.py
sudo /sbin/chkconfig tactic on
sudo chown -R apache:apache /home/apache/tactic /home/apache/assets /home/apache/ ↩
    tactic_data /home/apache/tacticTemp
```

1. Upgrade the database.

```
python2.7 /home/apache/tactic/src/bin/upgrade_db.py
```

1. Startup TACTIC in dev mode.

```
python2.7 /home/apache/tactic/src/bin/startup_dev.py
```

1. Try accessing TACTIC through a web browser on a client machine.
2. Once startup_dev works, Ctrl^C out of the process.

```
Ctrl^C
```

1. Open the TACTIC service file for edit.

```
sudo vi /etc/init.d/tactic
```

Modify the variable for PYTHON as follows:

```
PYTHON=/usr/local/bin/python2.7
```

1. Open the TACTIC configuration file for edit.

```
vi /home/apache/tactic_data/config/tactic_linux-conf.xml
```

Modify the option variable for python as follows:

```
<python>/usr/local/bin/python2.7</python>
```

1. Start TACTIC as a service.

```
sudo /etc/init.d/tactic start
```

End of installation instructions.

2.4 TACTIC Install - CentOS 5.5

Below are the step-by-step instructions of how to install TACTIC on CentOS 5.5. For CentOS 6.2 and above, please refer to the TACTIC Install - CentOS 6.2 page.

CentOS 5.5

Unfortunately, CentOS 5.5 comes pre-packaged with an outdated version of Python: Python 2.4. TACTIC requires Python 2.7.

To get around this, when we install Python 2.7, we do not overwrite the original Python 2.4. See steps #8 and #9.

Overwritting the original Python would break some software packages, such as Yum.

We must remember to call Python2.7 explicitly whenever we run any TACTIC python scripts, including startup.py

1. Log in as the root user.
2. Open the passwd file.

   ```
   vi /etc/passwd
   ```

   Modify the apache home directory and login shell to look like the following:

   ```
   \apache:x:48:48:Apache:/home/apache:/bin/bash
   ```

3. Set password for apache to: south123paw

   ```
   passwd apache
   ```

4. Open to the sudoers file.

```
visudo
```

   Add the apache user by including the following line in the appropriate location in the file:

```
apache ALL=(ALL) ALL
```

5. Create the home directory for apache.

```
mkdir /home/apache
chown apache:apache /home/apache
chmod a+rx /home/apache
```

6. Disable SElinux by setting SELINUX=disabled

```
vi /etc/selinux/config
```

7. Remove the existing Postgres and Install Postgres 8.4 modules.

```
yum remove postgresql postgresql-libs postgresql-server

vi /etc/yum.repos.d/CentOS-Base.repo
Add the exclude line to the 2 section base and updates:

[base]
exclude=postgresql*
[updates]
exclude=postgresql*

rpm -Uvh http://yum.pgrpms.org/reporpms/8.4/pgdg-centos-8.4-2.noarch.rpm
yum install -y postgresql postgresql-server postgresql-contrib
yum install -y postgresql-devel
yum install -y gcc zlib-devel samba libxslt-devel libxml2-devel
```

8. Install Python 2.7.

```
cd; wget http://www.python.org/ftp/python/2.7.1/Python-2.7.1.tgz
tar zxvf Python-2.7.1.tgz
cd Python-2.7.1
./configure -with-zlib=/usr/include
make install
```

9. Installing Python 2.7 will install python2.7 command "ahead" of the original python so that "python" from the command line is now 2.7. This will potentially break yum or other OS admin tools. To fix this, remove the python executable for 2.7.

```
rm /usr/local/bin/python
```

10. Restart the shell so that the environment is clean.

11. Install lxml.

```
cd; wget http://codespeak.net/lxml/lxml-2.2.8.tgz
tar zxvf lxml-2.2.8.tgz
cd lxml-2.2.8
python2.7 setup.py install
```

12. Install PIL.

```
cd; wget http://effbot.org/downloads/Imaging-1.1.7.tar.gz
tar zxvf Imaging-1.1.7.tar.gz
cd Imaging-1.1.7
python2.7 setup.py install
```

13. Install psycopg2.

```
cd; wget http://pypi.python.org/packages/source/p/psycopg2/psycopg2-2.3.2.tar.gz
http://initd.org/psycopg/tarballs/PSYCOPG-2-3/psycopg2-2.3.1.tar.gz
tar zxvf psycopg2-2.3.2.tar.gz
cd psycopg2-2.3.2
python2.7 setup.py install
```

14. Install PyCrypto.

```
cd; wget http://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.3.tar.gz
tar zxvf pycrypto-2.3.tar.gz
cd pycrypto-2.3
python2.7 setup.py install
```

15. Disable firewall.

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

16. Create and open the index.html for redirection.

```
vi /var/www/html/index.html
```

Insert the following contents:

```
<META http-equiv="refresh" content="0;URL=/tactic">
```

17. Re-login as the apache user.
18. Download the TACTIC source code

Open the following link in a web browser and download the latest TACTIC Enterprise release.

```
http://community.southpawtech.com/downloads
```

Set up the TACTIC service.

```
cd /tmp
unzip tactic_#.#.#.#.zip

sudo cp /tmp/tactic_#.#.#.#/src/install/service/tactic/etc/init.d
sudo chmod 775 /etc/init.d/tactic
sudo /sbin/chkconfig tactic on
```

19. Set up Postgres.

```
sudo service postgresql initdb
sudo /etc/init.d/postgresql start

sudo mv /var/lib/pgsql/data/pg_hba.conf /var/lib/pgsql/data/pg_hba.conf.bak
sudo cp /tmp/tactic_#.#.#.#/src/install/postgresql/pg_hba.conf /var/lib/pgsql/data
sudo chown postgres:postgres /var/lib/pgsql/data/pg_hba.conf
sudo /sbin/chkconfig postgresql on
sudo /etc/init.d/postgresql restart
```

20. Install TACTIC. You will be asked to enter a base directory for installation. We call this <TACTIC_BASE_DIR>

```
cd /tmp/tactic_#.#.#.#/src/install
sudo python2.7 install.py
sudo /sbin/chkconfig tactic on
```

21. Set up Apache. Copy the tactic.conf generated by the TACTIC Installer.

```
sudo cp  <TACTIC_BASE_DIR>/tactic_data/config/tactic.conf /etc/httpd/conf.d/
sudo /sbin/chkconfig httpd on
sudo /etc/init.d/httpd start
```

22. Startup TACTIC in dev mode.

```
python2.7 /home/apache/tactic/src/bin/startup_dev.py
```

23. Try accessing TACTIC through a web browser on a client machine. http://<server IP>/tactic
24. Once startup_dev works, Ctrl^C out of the process.

```
Ctrl^C
```

25. Open the TACTIC service file for edit.

```
sudo vi /etc/init.d/tactic
```

    Modify the variable for PYTHON as follows:

```
PYTHON=/usr/local/bin/python2.7
```

26. Open the TACTIC configuration file for edit.

```
vi  <TACTIC_BASE_DIR>/tactic_data/config/tactic_linux-conf.xml
```

    Modify the option variable for python as follows:

```
<python>/usr/local/bin/python2.7</python>
```

27. Start TACTIC as a service and install the license in a Java-enabled browser.

```
sudo /etc/init.d/tactic start
```

28. To view different information about the system and set-up, you can go to the Site Admin -→ System Info page. For example, you can verify if load-balancing is set up and certain key directories are writable by TACTIC.

End of installation instructions.

2.5 TACTIC Install - CentOS 5.7

CentOS 5.5 has now been deprecated in favour of CentOS 5.7 or CentOS 6.0. Since CentOS 5.5 will no longer receive updates and we urge you to consider using a more up to date version of the operating system such as CentOS 5.7.

For CentOS 6.2 and above, please refer to the TACTIC Install - CentOS 6.2 page. Step-by-step instructions for Tactic 3.6.0.v01 are provided below.

Requirements

Tactic requires the following software to be installed:

- Apache HTTP server 2.2

- Postgres Database Server 8.4 or higher
- Python 2.7 and the following Python modules:
  - Python Imaging Library 1.1.7
  - Python lxml 2.2.8
  - Pycrypto 2.3
  - Psycopg2 2.3.2

It should be noted that while there may be newer versions of these Python modules, Tactic 3.6.0.v01 is built with these particular versions and may not work as intended should you use later versions.

Disabling Security for Testing

For the sake of getting Tactic up quickly without a large amount of fuss, it is prudent to disable firewalling and SELinux. This is by no means an endorsement to run your server without these services. Once you have Tactic up and running we encourage you to read the section <TACTIC SECURITY>.

CentOS 5.7 uses the iptables service as a firewall. To disable this temporarily you can issue the following commands as root.

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

This will disable the firewall for the currently running CentOS session. Should you wish these settings to persist across reboots. You can issue the following command as root.

```
/sbin/chkconfig iptables off
```

To disable SELinux, edit the file /etc/selinux/config as root with your favourite editor and set the SELINUX variable to *disabled*.

```
SELINUX=disabled
```

Getting this setting to take effect requires a reboot.

Installing Apache HTTP server

To install Apache 2.2 on CentOS 5.7 issue the following command as root:

```
yum install httpd
```

This package has several dependencies and they should be installed as well.

It would be good to familarize yourself with some of the more pertinent paths in the Apache installation on CentOS5.7. We've listed them below for your convenience.

Apache 2.2 paths

/etc/httpd/conf/httpd.conf	Path to the main Apache configuration file.
/etc/httpd/conf.d/	Path to the Apache dynamic configuration directory.
/var/log/httpd/error.log	Path the the Apache error log.
/var/www/html	Path to the Document Root folder of the base Apache install.

It's best to test your Apache install at this point. The included main Apache configuration file is enough to get the server started. You will need create a small html file to do this. Using your favourite editor create a file called index.html, including the following contents:

```
<html>
  <head>
    <title>Apache Base Install</title>
  </head>
<body>
```

```
   <p>Apache Install Successful</p>
</body>
</html>
```

Save the file in /var/www/html and issue the following command as root to ensure proper permissions are set.

```
chmod 755 /var/www/html/index.html
```

Now we can start the Apache server itself. As root issue the following command.

```
/etc/init.d/httpd start
```

The service should start successfully. If it does not consult the Apache error log listed above to discover any problems that may have occured.

At last we should now be able to point a web browser to the IP address associated with the server. If you see the text "Apache Install Successful", then your server is up and running and server pages properly.

Finally we will want the Apache service to persist through reboots. As root issue the following command:

```
/sbin/chkconfig httpd on
```

Then reboot the server and make sure that httpd comes up on reboot and that you can view the test page properly. Later in the install we will configure the Apache server to interact with Tactic properly.

Installing Postgres Database Server

CentOS 5.7 repositories use Postgres version 8.1.23-1. However Tactic requires version 8.4 or higher. First we must make sure that we didn't include the default version Postgres in our base install. As root please issue the following command:

```
rpm -qva | grep -i postgres
```

This command will search your installed packages and filter the output for any packages containing the work postgres. If you receive any output from this command please uninstall the packages by issuing the following command:

```
rpm -e <package_name>
```

Where <package_name> is the name of the package you wish to uninstall. Consult the RPM or Yum manpages for further instructions on installing and uninstalling packages.

Lastly before installing any new Postgres repository, we want to make sure to exclude the default version of Postgres from showing up in the base and update Yum repositories included with the operating system itself. Using your favourite editor, edit the file /etc/yum.repos.d/CentOS-Base.repo to include the line:

```
exclude=postgresql*
```

Under both the [base] section and the [updates] section of the file. Save the file. For more information on the Yum installer and repository system please see the manpages for yum or the yum website http://yum.baseurl.com

Once we have uninstalled any previous versions of Postgres, we can then proceed installing a new Yum repository that contains the version of Postgres we want to install. As root issue the following commands:

```
cd /tmp
wget http://yum.pgrpms.org/9.1/redhat/rhel-5-i386/pgdg-centos-91-9.1-4.noarch.rpm
rpm -Uvh pgdg-centos-91-9.1-4.noarch.rpm
```

At the time of writing the current version of Postgres was 9.1-4, for more current installation versions please visit http://yum.pgrpms.org/-repopackages.php and view the available repository RPMs for your operating system and architecture.

Lastly we will install the proper Postgres version 9.1 RPMs. As root issue the following command:

```
yum install postgresql91 postgresql91-server postgresql91-devel
```

Now that Postgres is installed we will do some basic configuration and testing to make sure our installation is ready for use with Tactic.

Again we should first familiarize ourselves with the pertinent Postgres file paths that will be used throughout the install:

Common Paths in Postgres 9.1 on CentOS 5.7

/var/lib/pgsql/9.1	Path to the main Postgres directory.
/var/lib/pgsql/data/pg_hba.conf	Path to the host based authentication file for Postgres.
/var/lib/pgsql/9.1/pgstartup.log	Path to the Postgres startup log.
/var/lib/pgsql/9.1/data/pg_log	Path to Postgres database logs.

If you have used the base postgresql-server install before you will notice that the above paths are slightly different than the usual paths used on CentOS. This install also varies from the base install in that the user must manually initialize the databases used by Postgres. To do this issue the following command as root:

```
/etc/init.d/postgresql-91 initdb
```

Once that command successfully completes we can then start the service. As root issue the following command:

```
/etc/init.d/postgresql-91 start
```

If the service starts successfully we can move forward. If not please view any startup errors in the Postgres startup log. Further troubleshooting help can be found at http://postgresql.org,

Lastly we will want to make sure the Postgres service persists through reboots. As root issue the following command:

```
/sbin/chkconfig postgresql-91 on
```

Reboot your server to ensure the service comes up appropriately.

Installing Python 2.7

As noted above Tactic requires version 2.7 of Python. CentOS comes with, and depends on an earlier version of Python. This means we simply can't remove Python from the system as we did with Postgres. We have to create a custom Python 2.7 install and keep it seperate from the base install.

First we need to install some development libraries and a compiler. As root issue the following command:

```
yum install gcc zlib-devel libxslt-devel libxml2-devel
```

Once those are installed we can get Python 2.7, as root issue the following command:

```
cd /tmp
wget http://www.python.org/ftp/python/2.7.1/Python-2.7.1.tgz
tar -zxvf Python-2.7.1.tgz
cd Python-2.7.1
./configure --prefix=/opt/python2.7
make && make install
```

This will install Python 2.7 in the custom path /opt/python2.7. You can choose a different custom install directory should you choose.

Installing Python 2.7 Modules

Next we will install the necessary Python 2.7 modules. These provide extended functionality to the Python environment. Before we begin however we will have to set an environment variable such that the Python 2.7 modules we are installing can find their link dependencies. In the BASH shell environment as root issue the following command:

```
export LD_LIBRARY_PATH=/opt/python2.7/lib
```

This will prefix your link dependency library path with our custom python libaries ensuring they get precedence in the installation of the modules. This environment variable only exists for the current shell environment. Any new shells you open will need to have this variable set as well. We will institute a permanent solution to this problem later.

Installing lxml 2.2.8

Tactic requires lxml version 2.2.8. At the time of writing Tactic encountered errors running with lxml 2.3 installed. Please make sure you install lxml 2.2x. Installation and setup for each module is almost the same. As root issue the following commands

```
cd /tmp
wget http://codespeak.net/lxml/lxml-2.2.8.tgz
tar -zxvf lxml-2.2.8.tgz
cd lxml-2.2.8
/opt/python/bin/python2.7 setup.py install
```

We have used the location of our custom installed python binary in the last line, if you have chosen a different install location use it instead.

Verify the installation by running your custom python binary and trying to import the module. As root issue the following command:

```
/opt/python2.7/bin/python2.7
```

You should receive a python environment and prompt similar to this:

```
Python 2.7.1 (r271:86832, Apr 12 2011, 16:16:18)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-51)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

At the prompt please type the following:

```
import lxml
```

If you receive another prompt and no error occurs, the import was successful and lxml is installed. We will use the same procedure to check the veracity of our other modules installations as well.

Installing Python Imaging Library 1.1.7

We follow the same procedure to install the Python Imaging Library. As root issue the following commands:

```
cd /tmp
wget http://effbot.org/downloads/Imaging-1.1.7.tar.gz
tar -zxvf Imaging-1.1.7.tar.gz
cd Imaging-1.1.7
/opt/python2.7/bin/python2.7 setup.py install
```

Check the installation of the module by loading python and issuing the following command at the python prompt.

```
import PIL
```

Installing Pycrypto 2.3

The installation of Pycrypto also follows this template. As root issue the following commands:

```
cd /tmp
wget http://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.3.tar.gz
tar -zxvf pycrypto-2.3.tar.gz
cd pycrypto-2.3
/opt/python2.7/bin/python2.7 setup.py install
```

Check the installation of the module by loading python and issuing the following command at the python prompt.

```
import Crypto
```

Installing Psycopg2 2.3.2

Psycopg2 is a Postgres database adapter for Python. The install is akin to the previous 3 except that since we've customized our Postgres install, we have to let Psycopg2 know that as well. First lets get the package and uncompress it. As root issue the following:

```
cd /tmp
wget http://pypi.python.org/packages/source/p/psycopg2/psycopg2-2.3.2.tar.gz
tar -zxvf psycopg2-2.3.2.tar.gz
cd psycopg2-2.3.2
```

Now we have to modify the pg_config variable in the file setup.cfg. Using your favourite editor open /tmp/psycopg2-2.3.2/setup.cfg and edit the pg_config entry to match the following:

```
pg_config=/usr/pgsql-9.1/bin/pg_config
```

Now we should be able to install Psycopg2 properly. As root issue the following command:

```
/opt/python2.7/bin/python2.7 setup.py install
```

Check the installation of the module by loading python and issuing the following command at the python prompt.

```
import psycopg2
```

Now that all the Tactic components are installed, we can begin installing Tactic itself.

Installing Tactic 3.6.0.v01

First we have to download the Tactic source from the Southpaw Community website. Using a web browser navigate to http://community.southpawtech.com and login using the support account you were assigned. After you have logged in, please click on "Downloads" in the upper right hand corner of the website. Once you are on the downloads page, you should see the download link for TACTIC Enterprise. Save the TACTIC Enterprise zip file in /tmp on the server.

Once we have the Tactic source we need to unpack it. As root issue the following commands:

```
cd /tmp
unzip tactic-3.6.0.v01.zip
```

Once the Tactic source is unpacked, you should familarize yourself with the Tactic directory structure. The prefix of the paths will change during install, however the directory structure will remain the same.

Common Paths in Tactic 3.6.0.v01

/tmp/tactic-3.6.0.v01/doc	Path to the Tactic documentation.
/tmp/tactic-3.6.0.v01/src	Path to the Tactic source files.
/tmp/tactic-3.6.0.v01/src/install	Path to the Tactic install script and configuration files for Tactic dependencies.

Before we run the Tactic installer, we must further configure Postgres to ensure proper installation of Tactic.

Configuring Postgres

Tactic supplies a custom host based authentication file for Postgres. After stopping the Postgres service we will install the file in question, restart the service and verify that the configuration has been properly put in place. As root issue the following commands:

```
/etc/init.d/postgresql-9.1 stop
mv /var/lib/pgsql/9.1/data/pg_hba.conf /var/lib/pgsql/9.1/data/pg_hba.conf.INSTALL
cp /tmp/tactic-3.6.0.v01/src/install/postgresql/pg_hba.conf /var/lib/pgsql/9.1/data/pg_hba. ↩
    conf
chown postgres:postgres /var/lib/pgsql/9.1/data/pg_hba.conf
/etc/init.d/postgresql-9.1 start
```

If the service starts successfully we can then move onto testing. If not please consult the Postgres startup log for error information. Postgres must be running as a service during the Tactic install. Next we will test the install, as root issue the following commands:

```
psql -U postgres template1
```

This command should return you to a prompt like this one:

```
template1=#
```

If you are not prompted for a password and, see this prompt, then the configuration changes have been successful. Type \q to exit.

Installing Tactic

We are now ready to run the Tactic installer. The installer doesn't know about our custom Python installation location though, so we have edit the install.py file. Using your favorite editor, edit the file /tmp/tactic-3.6.0.v01/src/install/install.py.

Find the following line in the file:

```
python_site_packages_dir = "/usr/local/lib/python%s.%s/site-packages"
```

Replace it with the following line:

```
python_site_packages_dir = "/opt/python2.7/lib/python%s.%s/site-packages"
```

Save the file. We will test this by running the installer. Make sure you are using a shell with the LD_LIBRARY_PATH set as mentioned in the "Installing Python 2.7 Modules" section. As root issue the following command:

```
/opt/python2.7/bin/python2.7 /tmp/tactic-3.6.0.v01/src/install/install.py
```

The installer will ask you a number of questions. You should be prompted by the following:

```
Please enter the base path of the TACTIC installation:

(/home/apache) -> /opt/tactic
```

Choose your custom Tactic path, this will be the base directory for your Tactic installation. We have chosen /opt/tactic. Next you will be asked to identify the user used to run the Apache service.

```
Please enter the user Apache Web Server is run under:

(apache) ->
```

In many cases the default user "apache" is correct. However you can verify this easily in another shell by issuing the following command as root:

```
ps -ef | grep httpd
```

The first column of output is the username Apache is running under. Should you receive no output, make sure your httpd service is running. Under CentOS 5.7 the apache user is "apache" so we will accept the default the installer gives us.

The installer will then install the Tactic databases, configure Tactic and copy the source code into the Tactic base directory.

Configuring Apache

Lastly Tactic requires us to install the custom Apache configuration file it has created during the installation process. As root issue the following commands:

```
cp /opt/tactic/tactic_data/config/tactic.conf /etc/httpd/conf.d
```

To test this we should first verify that /etc/httpd/conf.d is an included module path in the main Apache configuration file. As root issue the following command:

```
cat /etc/httpd/conf/httpd.conf | grep conf.d
```

If you receive the following as output we have installed the Tactic Apache configuration file correctly.

```
Include conf.d/*.conf
```

If you do not receive this output, using your favourite editor as root edit the file /etc/httpd/conf/httpd.conf and at the bottom of the file insert the statement listed above. Save the file.

Lastly we will create a redirect for Tactic using the index.html file at the document root of the Apache server. Using your favourite editor as root, edit the file /var/www/html/index.html to contain only the following line:

```
<META http-equiv="refresh" content="0;URL=/tactic">
```

Restart the Apache service to make sure the configuration hasn't caused any problems. As root issue the following command:

```
/etc/init.d/httpd restart
```

Lastly we need to ensure that the Apache server has loaded the modules needed for proxying and load balancing the Tactic service. As root issue the following command:

```
/usr/sbin/httpd -t -D DUMP_MODULES
```

In the resulting output look for the following modules:

```
rewrite_module (shared)
proxy_module (shared)
proxy_http_module (shared)
proxy_balancer_module (shared)
deflate_module (shared)
```

The CentOS 5.7 install of Apache includes a large array of modules enabled by default, including the ones required by Tactic.

Starting Tactic in Development Mode

We are now prepared to start the Tactic service for the first time. Tactic must be run as the Apache user. Usually we would su to the apache user, however since its a system account without a login this will not work without optionally telling su to give the Apache user a shell environment. In a shell with the LD_LIBRARY_PATH set as laid out in the section "Installing Python 2.7 Modules", as root issue the following command:

```
su apache -s /bin/bash -c "/opt/python2.7/bin/python2.7 /opt/tactic/tactic/src/bin/ ↩
    startup_dev.py"
```

You should see the following output:

```
Registering site ... admin
Registering site ... sthpw
Registering site ... default

Starting TACTIC ...

Starting Scheduler ....
[08/Nov/2011:23;13:20] ENGINE Bus STARTING
[08/Nov/2011:23;13:20] ENGINE Started monitor thread '_TimeoutMonitor'.
[08/Nov/2011:23;13:20] ENGINE Started monitor thread 'Autoreloader'
[08/Nov/2011:23;13:21] ENGINE Serving on 127.0.0.1:8081
[08/Nov/2011:23;13:21] ENGINE Bus STARTED
```

You should now be able to navigate a web browser to http://localhost/tactic/admin. Alternatively use the non-local IP address or fully qualified domain name you have assigned to your Tactic server in place of localhost. You should see the Tactic login screen.

Tactic defines an administrative user by default. The login information is:Use

Username: admin; Password: (set it on your first login)

In 3.9, the login screen requires you to change the password immediately. You will have to type it in twice to confirm.

Installing Tactic as a Service

You can use the key combination CTRL+c in the shell running the Tactic Development mode service to stop it when you are ready

The Tactic service file is simple to install, however due to the custom Python pathing we have used we have edit the Tactic service file to reflect those changes. As root, using your favourite editor open the file /opt/tactic/tactic/src/install/service/tactic and make the following changes.

The Python environment variable PYTHON should be set to the path of our custom Python binary.

```
PYTHON=/opt/python2.7/bin/python2.7
```

Additionally we want to setup the LD_LIBRARY_PATH variable to include our custom Python libraries. Under the following line:

```
export PYTHON
```

Add the following lines to the file:

```
LD_LIBRARY_PATH=/opt/python2.7/lib
export LD_LIBRARY_PATH
```

Lastly, Tactic startup uses the same mechanism as Developer mode, however this is not yet reflected in the Tactic service file. Change the following line:

```
su - $TACTIC_USER -m -c "$LAUNCH" >> $LOG 2>&1 &
```

To this:

```
su $TACTIC_USER -s /bin/bash -m -c "$LAUNCH" >> $LOG 2>&1 &
```

Save the file.

One last Python path must be corrected before we install the Tactic service. During installation Tactic created a file that you will become more familiar with throughout your use of Tactic. That file is /opt/tactic/tactic_data/config/tactic_linux-conf.xml. Using your favourite editor as root edit this file. Find the following line:

```
<python>python</python>
```

And replace it with:

```
<python>/opt/python2.7/bin/python2.7</python>
```

Save the file.

Now we will install the Tactic service. As root type the following:

```
cp /opt/tactic/tactic/src/install/service/tactic /etc/init.d/tactic
chmod 755 /etc/init.d/tactic
/sbin/chkconfig tactic on
/etc/init.d/tactic start
```

Tactic should now start properly as a service. Test this by launching a web browser and navigating to the Tactic login page as you did previously. If you are having trouble the following error logs will help you diagnose your problems:

Common Error Logs

/opt/tactic/tactic_temp/log/stdout.log	Path to the TACTIC standard out log assuming you have chosen /opt/tactic as the base installation path
/var/log/tactic	Path to the Tactic error log.
/var/log/httpd/error_log	Path to the Apache error log.
/var/log/httpd/access_log	Path to the Apache access log.

Finally, reboot the server and test whether the TACTIC service continues to perform across reboots. You are advised to set up log rotation by reviewing the Rotate Log section in the Sys Admiin documentation.

2.6 TACTIC Install - CentOS 6.2

While there may be some slight variations in the steps in different versions of CentOS 6, this guide can be used for CentOS 6.x as a reference.

Requirements

TACTIC requires the following software to be installed:

- Apache HTTP server 2.2
- Postgres Database Server 8.4 or higher
- Python 2.6 or 2.7 with the following Python modules:
  - Python Imaging Library 1.1.7
  - Python lxml 2.3.5
  - Pycrypto 2.3
  - Psycopg2 2.4.6

It should be noted that while there may be newer versions of these Python modules, TACTIC 3.9.0.rc04 has been tested with these particular versions.

Disabling Security for Testing

To get TACTIC up quickly without fiddling with OS security, it is prudent to disable firewalling and SELinux. This is by no means an endorsement to run your server without these services. Once you have TACTIC up and running, you are advised to read the section <TACTIC SECURITY>.

CentOS 6.2 uses the iptables service as a firewall. To disable this temporarily, you can issue the following commands as root.

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

This will disable the firewall for the currently running CentOS session. Should you wish these settings to persist across reboots, you can issue the following command as root.

```
/sbin/chkconfig iptables off
```

To disable SELinux, edit the file /etc/selinux/config as root with your editor of choice and set the SELINUX variable to *disabled*.

```
SELINUX=disabled
```

Since getting this setting to take effect requires a reboot, you can also issue the following command for it to take effect immediately.

```
echo 0 > /selinux/enforce
```

Installing Apache HTTP server

In case Apache 2.2 is not already pre-installed on CentOS 6.2, issue the following command as root:

```
yum install httpd
```

This package has several dependencies and they should be installed as well.

It would be good to familarize yourself with some of the more pertinent paths in the Apache installation on CentOS 6.2. They are listed below for your convenience.

Table 1: Common Paths in Apache 2.2

/etc/httpd/conf/httpd.conf	Path to the main Apache configuration file.
/etc/httpd/conf.d/	Path to the Apache dynamic configuration directory.
/var/log/httpd/error.log	Path the the Apache error log.
/var/www/html	Path to the Document Root folder of the base Apache install.

It's best to test your Apache install at this point. The included main Apache configuration file is enough to get the server started. You will need create a small html file to do this. Using your editor of choice, create a file called index.html, with the content as follows:

```
<html>
  <head>
    <title>Apache Base Install</title>
  </head>
<body>
   <p>Apache Install Successful</p>
</body>
</html>
```

Save the file in /var/www/html and issue the following command as root to ensure proper permissions are set:

```
chmod 755 /var/www/html/index.html
```

The Apache server can be started. As root, issue the following command:

```
/etc/init.d/httpd start
```

The service should start successfully. Otherwise, consult the Apache error log listed above to discover any problems that may have occured.

At last, you can point a web browser to the IP address associated with the server. If you see the text "Apache Install Successful", then your server is up and running and can serve pages properly.

Finally, to make the Apache service persist through reboots, issue the following command as root:

```
/sbin/chkconfig httpd on
```

Then reboot the server and make sure that httpd comes up on reboot and that you can view the test page properly. Later in the install, the Apache server will be configured to interact with TACTIC properly.

Installing Postgres Database Server

TACTIC requires PostgreSQL version 8.4 or higher. First we must make sure that we didn't include the default version Postgres in our base install. As root please issue the following command:

```
rpm -qva | grep -i postgres
```

This command will search your installed packages and filter the output for any packages containing the work postgres. If you receive any output from this command please uninstall the packages by issuing the following command:

```
rpm -e <package_name>
```

Where <package_name> is the name of the package you wish to uninstall. Consult the RPM or Yum manpages for further instructions on installing and uninstalling packages.

Lastly before installing any new Postgres repository, we want to make sure to exclude the default version of Postgres from showing up in the base and update Yum repositories included with the operating system itself. Using a text editor, edit the file /etc/yum.repos.d/CentOS-Base.repo to include the line under both the [base] section and the [updates] section of the file. :

```
exclude=postgresql*
```

Save the file. For more information on the Yum installer and repository system please see the manpages for yum or the yum website http://yum.baseurl.com

Once we have uninstalled any previous versions of Postgres, we can then proceed installing a new Yum repository that contains the version of Postgres we want to install. As root issue the following commands:

```
rpm -Uvh http://yum.postgresql.org/9.2/redhat/rhel-6-x86_64/pgdg-centos92-9.2-6.noarch.rpm
```

At the time of writing the current version of Postgres was 9.2-6, for more current installation versions please visit http://yum.pgrpms.org/-repopackages.php and view the available repository RPMs for your operating system and architecture.

Lastly, we will install the proper Postgres version 9.2 RPMs. As root, issue the following command:

```
yum install postgresql postgresql-server postgresql-contrib postgresql-devel
```

Now that Postgres is installed, you are advised to make some basic configuration to make sure the installation is ready for use with TACTIC.

Here are the pertinent Postgres file paths that will be used throughout the install:

Table 2: Common Paths in Postgres 9.2 on CentOS 6.2

/var/lib/pgsql/9.2	Path to the main Postgres directory.
/var/lib/pgsql/data/pg_hba.conf	Path to the host based authentication file for Postgres.
/var/lib/pgsql/9.2/pgstartup.log	Path to the Postgres startup log.
/var/lib/pgsql/9.2/data/pg_log	Path to Postgres database logs.

Initialize the database:

```
su - postgres -c /usr/pgsql-9.2/bin/initdb
```

Once that command successfully completes we can then start the service. As root issue the following command:

```
service postgresql-9.2 start
```

If the service starts successfully, we can move forward. Otherwise, you can review any startup errors in the Postgres startup log. Further troubleshooting help can be found at http://postgresql.org.

Lastly, to make sure the Postgres service persist through reboots. you will issue the following command:

```
chkconfig --levels 235 postgresql-9.2 on
```

Reboot your server to ensure the service comes up appropriately.

Installing Python Devel for the preinstalled Python

As noted above TACTIC requires version 2.6 or 2.7 of Python. In CentOS 5.x, we used to recommend installing Python 2.7 along side the built-in Python 2.4. In CentOS 6.2, the built-in Python 2.6.6 already suffices.

First, we need to install some development libraries and a compiler. As root issue the following command:

```
yum install gcc zlib-devel libxslt-devel libxml2-devel

yum install python-devel
```

This will install the Python Devel and all the other relevant libraries to install the various Python modules.

Installing Python Modules

Installing lxml 2.3.5

TACTIC requires lxml version 2.2 or above. 2.3.5 is used at the time of writing. As root, issue the following commands:

```
cd /tmp
wget http://lxml.de/files/lxml-2.3.5.tgz
tar -zxvf lxml-2.3.5.tgz
cd lxml-2.3.5
python setup.py install
```

Verify the installation by running your custom python binary and trying to import the module. As root issue the following command:

```
python
```

You should receive a python environment and prompt similar to this:

```
Python 2.6.6 (r271:86832, Apr 12 2011, 16:16:18)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-51)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

At the prompt please type the following:

```
import lxml
```

If you receive another prompt and no error occurs, the import was successful and LXML is installed. We will use the same procedure to check the veracity of our other modules installations as well.

Installing Python Imaging Library 1.1.7

We follow the same procedure to install the Python Imaging Library. As root issue the following commands:

```
cd /tmp
wget http://effbot.org/downloads/Imaging-1.1.7.tar.gz
tar -zxvf Imaging-1.1.7.tar.gz
cd Imaging-1.1.7
/opt/python2.7/bin/python2.7 setup.py install
```

Check the installation of the module by loading python and issuing the following command at the python prompt:

```
import PIL
```

Installing Pycrypto 2.3

The installation of Pycrypto also follows this template. As root issue the following commands:

```
cd /tmp
wget http://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.3.tar.gz
tar -zxvf pycrypto-2.3.tar.gz
cd pycrypto-2.3
python setup.py install
```

Check the installation of the module by loading python and issuing the following command at the Python prompt:

```
import Crypto
```

Installing Psycopg2 2.4.6

Psycopg2 is a Postgres database adapter for Python. The install is akin to the previous 3 except that since we've customized our Postgres install, we have to let Psycopg2 know that as well. First lets get the package and uncompress it. As root issue the following:

```
yum install postgresql-libs
```

```
cd /tmp
wget http://initd.org/psycopg/tarballs/PSYCOPG-2-4/psycopg2-2.4.6.tar.gz
tar -zxvf psycopg2-2.4.6.tar.gz
cd psycopg2-2.4.6
```

Now we have to modify the pg_config variable in the file setup.cfg. Using your editor to open /tmp/psycopg2-2.4.6/setup.cfg and edit the pg_config entry to match the following:

```
pg_config=/usr/pgsql-9.2/bin/pg_config
```

Now we should be able to install Psycopg2 properly. As root issue the following command:

```
python setup.py install
```

Check the installation of the module by loading python and issuing the following command at the python prompt:

```
import psycopg2
```

Now that all the TACTIC components are installed, we can begin running the TACTIC installer.

Installing Tactic 3.9.0.rc04

First we have to download the TACTIC source from the Southpaw Community website. Using a web browser navigate to http://community.southpawtech.com. After you have logged in, please click on "Downloads" in the upper right hand corner of the website. Once you are on the downloads page, you should see the download link for the latest TACTIC Enterprise. Save the TACTIC Enterprise zip file in /tmp on the server.

Once we have the Tactic source we need to unpack it. As root issue the following commands:

```
cd /tmp
unzip tactic-3.9.0.rc04.zip
```

Once the TACTIC source is unpacked, you should familarize yourself with its directory structure. The prefix of the paths will change during install; however, the directory structure will remain the same.

Table 3: Common Paths in Tactic 3.9.0.rc04

/tmp/tactic-3.9.0.rc04/doc	Path to the Tactic documentation.
/tmp/tactic-3.9.0.rc04/src	Path to the Tactic source files.
/tmp/tactic-3.9.0.rc04/src/install	Path to the Tactic install script and configuration files for Tactic dependencies.

Before we run the Tactic installer, we must further configure Postgres to ensure proper installation of Tactic.

Configuring Postgres

Tactic supplies a custom host based authentication file for Postgres. After stopping the Postgres service we will install the file in question, restart the service and verify that the configuration has been properly put in place. As root issue the following commands:

```
service postgresql-9.2 stop
mv /var/lib/pgsql/9.2/data/pg_hba.conf /var/lib/pgsql/9.2/data/pg_hba.conf.INSTALL
cp /tmp/tactic-3.9.0.rc04/src/install/postgresql/pg_hba.conf /var/lib/pgsql/9.2/data/pg_hba ↩
    .conf
chown postgres:postgres /var/lib/pgsql/9.2/data/pg_hba.conf
service postgresql-9.2 start
```

If the service starts successfully we can then proceed to verification. If not please consult the Postgres startup log for error information. Postgres must be running as a service during the Tactic install. Next we will test the install. As root, issue the following command:

```
psql -U postgres template1
```

This command should fire up a prompt like this:

```
template1=#
```

If you can see this prompt without being asked for a password, the configuration changes have been successful. Type \q to exit.

Installing Tactic

```
python /tmp/tactic-3.9.0.rc03/src/install/install.py
```

The installer will ask you a number of questions. You should be prompted by the following:

```
Please enter the base path of the TACTIC installation:

(/home/apache) -> /opt/tactic
```

Choose your custom Tactic path, this will be the base directory for your Tactic installation. We have chosen /opt/tactic. Next you will be asked to identify the user used to run the Apache service.

```
Please enter the user Apache Web Server is run under:

(apache) ->
```

In many cases the default user "apache" is correct. However you can verify this easily in another shell by issueing the following command as root:

```
ps -ef | grep httpd
```

The first column of output is the username Apache is running under. Should you receive no output, make sure your httpd service is running. Under CentOS 6.2 the apache user is "apache" so we will accept the default the installer gives us.

The installer will then install the Tactic databases, configure Tactic and copy the source code into the Tactic base directory.

Configuring Apache

Lastly Tactic requires us to install the custom Apache configuration file it has created during the installation process. As root issue the following commands:

```
cp /opt/tactic/tactic_data/config/tactic.conf /etc/httpd/conf.d
```

To test this we should first verify that /etc/httpd/conf.d is an included module path in the main Apache configuration file. As root issue the following command:

```
cat /etc/httpd/conf/httpd.conf | grep conf.d
```

If you receive the following as output we have installed the Tactic Apache configuration file correctly.

```
Include conf.d/*.conf
```

If you do not receive this output, using your editor as root edit the file /etc/httpd/conf/httpd.conf and at the bottom of the file insert the statement listed above. Save the file.

Lastly we will create a redirect for Tactic using the index.html file at the document root of the Apache server. Using a text editor as root, edit the file /var/www/html/index.html to contain only the following line:

```
<META http-equiv="refresh" content="0;URL=/tactic">
```

Restart the Apache service to make sure the configuration hasn't caused any problems. As root issue the following command:

```
/etc/init.d/httpd restart
```

Lastly we need to ensure that the Apache server has loaded the modules needed for proxying and load balancing the Tactic service. As root issue the following command:

```
/usr/sbin/httpd -t -D DUMP_MODULES
```

In the resulting output look for the following modules:

```
rewrite_module (shared)
proxy_module (shared)
proxy_http_module (shared)
proxy_balancer_module (shared)
deflate_module (shared)
```

The CentOS 6.2 install of Apache includes a large array of modules enabled by default, including the ones required by Tactic.

Starting Tactic in Development Mode

We are now prepared to start the Tactic service for the first time. Tactic must be run as the Apache user. Usually we would su to the apache user, however since its a system account without a login this will not work without optionally telling su to give the Apache user a shell environment. In a shell with the LD_LIBRARY_PATH set as laid out in the section "Installing Python 2.7 Modules", as root issue the following command:

```
su apache -s /bin/bash -c "/opt/python2.7/bin/python2.7 /opt/tactic/tactic/src/bin/ ↩
    startup_dev.py"
```

You should see the following output:

```
Registering site ... admin
Registering site ... sthpw
Registering site ... default

Starting TACTIC ...

Starting Scheduler ....
[08/Nov/2011:23;13:20] ENGINE Bus STARTING
[08/Nov/2011:23;13:20] ENGINE Started monitor thread '_TimeoutMonitor'.
[08/Nov/2011:23;13:20] ENGINE Started monitor thread 'Autoreloader'
[08/Nov/2011:23;13:21] ENGINE Serving on 127.0.0.1:8081
[08/Nov/2011:23;13:21] ENGINE Bus STARTED
```

You should now be able to navigate a web browser to http://localhost/tactic/admin. Alternatively, use the non-local IP address or fully qualified domain name you have assigned to your TACTIC server in place of localhost. You should see the TACTIC login screen.

TACTIC defines an administrative user by default. The login information is:

Username: admin; Password: (set it on your first login)
In 3.9, the login screen requires you to change the password immediately. You will have to type it in twice to confirm.

Installing TACTIC as a Service

The TACTIC service file is simple to install

Use the key combination CTRL+c in the shell running TACIC in Development mode to stop it first when you are ready to run TACTIC as a service.

Change the following line in the file /opt/tactic/tactic/src/install/service/tactic:

```
su - $TACTIC_USER -m -c "$LAUNCH" >> $LOG 2>&1 &
```

to this:

```
su $TACTIC_USER -s /bin/bash -m -c "$LAUNCH" >> $LOG 2>&1 &
```

Save the file.

Now as root, type the following:

```
cp /opt/tactic/tactic/src/install/service/tactic /etc/init.d/tactic
chmod 755 /etc/init.d/tactic
/sbin/chkconfig tactic on
/etc/init.d/tactic start
```

TACTIC should now start properly as a service. Test this by launching a web browser and navigating to the TACTIC login page as you did previously. If you are having trouble, the following error logs will help you diagnose your problems:

Table 4: Common Error Logs

/opt/tactic/tactic_temp/log/stdout.log	Path to the TACTIC standard out log assuming you have chosen /opt/tactic as the base installation path
/var/log/tactic	Path to the TACTIC error log in service mode
/var/log/httpd/error_log	Path to the Apache error log
/var/log/httpd/access_log	Path to the Apache access log

Finally, reboot the server and test whether the TACTIC service continues to perform across reboots. You are advised to set up log rotation by reviewing the Rotate Log section in the Sys Admin documentation.

2.7 TACTIC Install - Windows Server 2008

To completely install TACTIC Enterprise, there are three main components that have to be set up in the following order:

1. A Database
2. The TACTIC Installer
3. A Web Server

TACTIC stores all metadata in a database called PostgreSQL. This is an industrial strength, hugely scalable database that has proven itself in thousands of industries around the world. Go to the PostgreSQL website for more information.

From the PostgreSQL site (http://www.postgresql.org), download the .msi file for the latest binary distribution. We have encountered no problems using any of the versions above 8.4. You can find the binary distribution at this site. Follow the installation wizard for PostgreSQL. It is safe to choose all of the default settings if you do not have much experience with the PostgreSQL. It is best not to set a password during installation, although the Windows installer may force you to set one up.

Unzip Tactic Enterprise in a temporary location like C:/temp

It is mandatory you edit the "pg_hba.conf" file after installation

It is located in C:/Program Files/PostgreSQL/<version>/data This file determines all of the user permissions for PostgreSQL. You can open this file with any text editor like notepad. To begin with, turn on all of the permissions contained in this file. This is a temporary measure that will greatly simplify the installation process. You may lock down these permissions at a later date. Please consult the PostgreSQL documentation on how to do this. TACTIC ships with a sample "pg_hba.conf" file (located in <TACTIC_unzipped_package>/src/install/postgresql/pg_hba.conf). This file has an open security setting for ease of installation. It's best to back up your current pg_hba.conf file before copying over with the file provided by TACTIC. Note: in Windows, you would need to comment out the line

local all all trust

You add a comment by putting a # at the beginning of the line like this:

#local all all trust

Add the PostgreSQL bin directory to the environment path in Advanced System Settings: C:\Program Files\PostgreSQL\<version>\bin\

To do this, go to Start -→ Control Panel. If it's organized in category mode, go to system and security -→ system. If its not categorized, you should see system directly. In system, find advanced system and settings -→ go to the advanced tab -→ Environment variables. Now in the system variables box, Navigate to the variable "Path". Add the bin directory to the end of the value.

Restart the PostgreSQL Service:

You can go to the control panel again to do this. If it's organized in category mode, go to system and security -→ administrative tools. If it's not categorized, you will see administrative tools directly. Inside, you can find services. Find the postgresql service and restart it. Note: If you have command prompt open, you need to reopen it for the environment variable to take effect.

Verification

Verify that psql works in the command prompt:

```
> psql -U postgres template1
```

It should give you a prompt:

```
template1=#
```

If you see this prompt without the need to enter a password, you have successfully installed the PostgreSQL database. Type \q to exit.

Install Python and supporting modules

The TACTIC source code is written in Python. As such, the complete codebase is open.

To install Python, you can find an msi at http://www.python.org. TACTIC requires Python 2.6 or higher.

TACTIC also requires a number of Python modules to function correctly. These modules are generally not installed by default with the standard Python distributions. For Windows, they are already packaged in a file python_modules.zip available for download in the release section of the downloads page at http://support.southpawtech.com/downloads

- `PIL 1.1.7` (Python Image Library)
- `ImageMagick 6.7.8` (A command line program which complements PIL) - http://www.imagemagick.org
- `FFmpeg version 0.6` (Solution to record, convert and stream audio and video and metadata parsing) - http://www.ffmpeg.org
- `psycopg2 2.3.1` (Database connectivity)
- `PyCrypto version 2.0.1/2.1` (For de-encrypting the license file) - http://www.amk.ca/python/code/crypto
- `simplejson 2.1.1` (not needed since Python 2.6) simplejson.egg can be installed with the EasyInstaller. It is needed for JSON string encoding and parsing
- `lxml 2.1 or 2.2` (XML and XPath processing)
- `pywin32-(build 2##)` (Python Windows OS module)

Windows

Unfortunately, Python is not preinstalled on Windows machines; however, with the installer this is not difficult.

The Python installer does not set the Python path, so this must be added

Go to Start→Control Panel→System Click on the "Advanced" tab. Select "Path" under "System Variables" and click the "Edit" button. Add Python to the environment path: "C:\Python26\" to the end of the "Variable Value" To test that Python is working from the command prompt, run the command: `python`

You should get the following prompt:

```
Python 2.6.2 (#67, Sep 28 2009, 12:41:11) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you haven't done so already above, unzip Tactic Enterprise in a temporary location like C:/temp

Go to C:/temp/tactic_#../src/install/

```
# cd C:/temp/tactic_#.#.#.#/src/install
```

Execute: install.py

> **Note**
>
> You must invoke the installation with root user privileges because it attempts to write to the <python_install>/site-packages directory.

```
# python install.py
```

The installer will ask a number of questions. First it ask for the <TACTIC_BASE_DIR>:

```
Please enter the base path of the TACTIC installation:

(C:/Program Files/Southpaw) ->
```

It would copy the source code to the base path. An Apache Web Server file will be generated at the end which you would need to copy to the Apache config area upon installation of the web server. If there are existing files in the destination directory the installer will ask for your confirmation to remove it. At the end, you will see this:

```
Installing service TacticService
Changing service configuration
Service updated

*** Installation of TACTIC completed at [C:/Program Files/Southpaw] ***


Next, please install the Apache Web Server and then copy the Apache config extension
[C:/Program Files/Southpaw/tactic_data/config/tactic_win32.conf] to the Apache web server ↩
    config area. e.g.
C:/Program Files/Apache Software Foundation/Apache2.2/conf
```

Verification

When the installation is completed, an asset directory will be created at *<TACTIC_BASE_DIR>/assets*.

The file tactic_paths.py will be created in the following directories:

Windows

```
C:/Python##/Lib/site-packages/tacticenv
```

Default contents:

```
TACTIC_INSTALL_DIR='C:/Program Files/Southpaw/tactic'
TACTIC_SITE_DIR=''
TACTIC_DATA_DIR='C:/Program Files/Southpaw/tactic_data'
```

Network machine access

If you are on the server, you can access it by using the URL http://localhost/tactic. For other people to access it on the network, you need to find out its IP address. In a linux server, you can use the command "ifconfig" to locate it. It's the one listed as the inet addr.

Next, you need to run TACTIC behind Apache.

TACTIC should be run behind an Apache web server. You can download Apache software at http://www.apache.org/

The TACTIC application server is able to serve up static content such as images, PDF files, Quicktime files, and so on, but it is not the most efficient at this because it is written in Python. This is what Apache is designed for. By running TACTIC behind Apache, it relieves TACTIC from serving the static content so that it can focus on the dynamic content.

For production use, it is highly recommended that TACTIC is run behind the Apache server. This has many scalability advantages. When running behind Apache, Apache uses a reverse proxy and proxy balancer module to forward requests to communicate with TACTIC.

After the installation, some changes may need to be made in the "httpd.conf" file for Apache.

```
C:/Program Files/Apache Software Foundation/Apache2.2/conf/httpd.conf
```

Make sure the following lines are uncommented:

```
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule deflate_module modules/mod_deflate.so
```

Additional lines For Apache 2.4:

```
LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
LoadModule filter_module modules/mod_filter.so
LoadModule lbmethod_byrequests_module modules/mod_lbmethod_byrequests.so
```

These lines may already be uncommented, depending on your distribution and version of Apache. You need version 2.0.31 or later.

The TACTIC installer generates an Apache extension conf file on completion. You need to copy that file to a directory that is recognized by Apache.

For Windows, this directory is:

` C:/Program Files/Apache Software Foundation/Apache2.2/conf`

You must add the following line to the "httpd.conf" file so that it will read the extension conf file provided by the installer:

```
Include conf/tactic_win32.conf
```

Web Server configuration

In our provided file:

```
tactic_win32.conf
```

Make sure the following lines exist:

```
# Using the ProxyPass directives
ProxyPreserveHost on

<Proxy balancer://tactic>
BalancerMember http://localhost:8081/tactic
BalancerMember http://localhost:8082/tactic
BalancerMember http://localhost:8083/tactic
</Proxy>
ProxyPass /tactic balancer://tactic
ProxyPass /tactic_data balancer://tactic
```

For Apache 2.4:

```
Comment out:
#Order Allow,Deny
#Allow from All

Uncomment:
Require all granted
```

Note

Warning: For load balancing, only use either:

1. the Proxy Balancer method (recommended)

   or

2. the RewriteRule method (not recommended).

Do **not** use both methods at the same time.

Note

For configuring load-balancing set-up in a real production, please refer to the Load Balancing section in the Sys-admin docs.

Note

When trying to set up Apache on a Windows Server, to specify a share folder for Apache to use, you may need to include the name of the share folder in the path.

For example, use the following if you have named the share folder "my_share":

```
Alias /assets "//10.0.0.17/my_share/assets"
```

Note: For configuring load-balancing set-up in a real production, please refer to the Load Balancing section of this Sys-admin doc.

Finally, after verifying the configuration is correct, restart the Apache service:

```
You can do it in the Windows Services UI or through the task tray and right click on the  ↩
    Apache icon.
```

Go to the <TACTIC_INSTALL_DIR>/src/bin folder

`cd C:/Program Files/Southpaw/tactic/src/bin `

```
python startup_dev.py
```

This "startup_dev.py" script is the development script which will dump output to the screen. The other startup script "startup.py" is the production start-up script and will dump output to a log file. The development start-up script is also much slower as it monitors the file system to see if any files have changed.

The output would look like the following:

```
Registering site ... admin
Registering site ... default
Registering site ... test
Registering site ... my_project

Starting TACTIC ...

05/Jul/2007:11:16:29 CONFIG INFO Server parameters:
05/Jul/2007:11:16:29 CONFIG INFO server.environment: development
05/Jul/2007:11:16:29 CONFIG INFO server.log_to_screen: True
```

```
05/Jul/2007:11:16:29 CONFIG INFO server.log_file: D:/tactic_temp/log/tactic_log
05/Jul/2007:11:16:29 CONFIG INFO server.log_tracebacks: True
05/Jul/2007:11:16:29 CONFIG INFO server.log_request_headers: True
05/Jul/2007:11:16:29 CONFIG INFO server.protocol_version: HTTP/1.0
05/Jul/2007:11:16:29 CONFIG INFO server.socket_host:
05/Jul/2007:11:16:29 CONFIG INFO server.socket_port: 8081
05/Jul/2007:11:16:29 CONFIG INFO server.socket_file:
05/Jul/2007:11:16:29 CONFIG INFO server.reverse_dns: False
05/Jul/2007:11:16:29 CONFIG INFO server.socket_queue_size: 10
05/Jul/2007:11:16:29 CONFIG INFO server.thread_pool: 5
05/Jul/2007:11:16:30 HTTP INFO Serving HTTP on http://localhost:8081/
```

`http://<TACTIC_server_address>/tactic/admin/`

You should see the TACTIC login appear.

There is a default user created on installation. This is the "admin" user and this user has the ability to see and change all aspects of the system. Log in as the admin user:

Note: You may be asked to change your password automatically at startup without entering these default credentials.

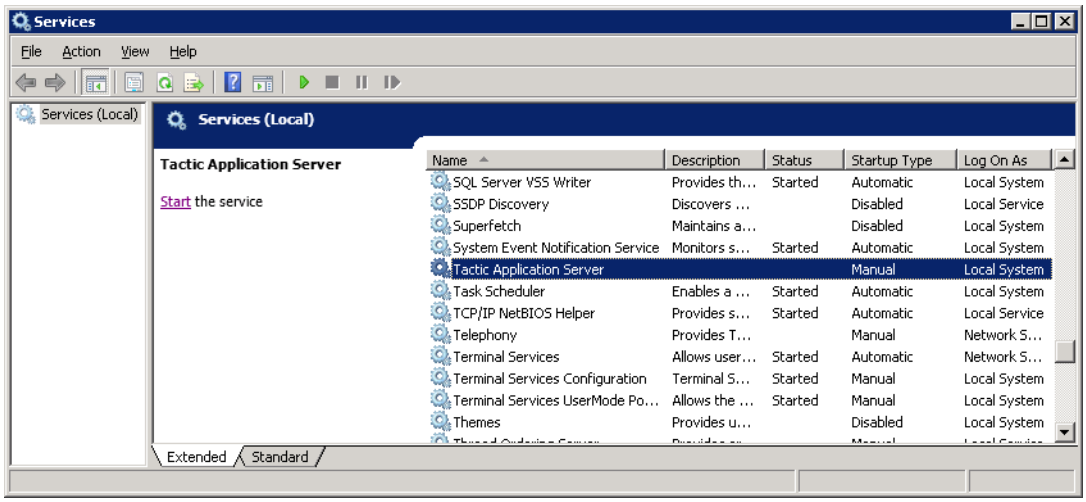
user:	`admin`
password:	`tactic`

Note: The TACTIC service should have been set up during the installation. You just need to start it.

First, stop TACTIC running in dev mode if applicable by pressing Ctrl
C in that shell

In Windows, :

`Start the "Tactic Application Server" in Windows Services UI.`



At this point you will need to install a TACTIC license file and then begin to set up a project.

- For more information on installing the license file, please refer to the Install License File chapter.*
- For more information on getting started with projects, please refer to Setup documentation.*

Note on supplementary directories used in TACTIC:

In the config file <TACTIC_DATA_DIR>/config/tactic-conf.xml, you will find different references to directory under the path /home/apache like /home/apache/tactic_temp , assets, and handoff. They are there because /home/apache is the base directory chosen for installation. You will find the Windows equivalent in the Windows installation. It could be different in your case. In the future, you can change their locations by editing the config file. If the assets directory is changed, ensure it's also updated in the Apache Web server config extension tactic.conf.

To view different information about the system and set-up, you can go to the Global -→ System Info page. For example, you can verify if load-balancing is set up and certain key directories are writable by TACTIC.

- If you see missing images on logging in or errors in the output log saying certain js files are not found, it means the Apache extension file tactic.conf is not being referenced or you have not updated the paths in it to reflect current <TACTIC_INSTALL_DIR>. For example, if your <TACTIC_INSTALL_DIR> is /home/apache/tactic, you should see the line in tactic.conf:

```
Alias /context C:/Program Files/Southpaw/tactic/src/context
```

- If you try to check in a Preview Image and can't see the icon generated, your assets alias in the Apache extension file may not be in sync with the asset_base_dir in the TACTIC config file tactic_linux-conf.xml

```
Alias /assets           C:/ProgramData/Southpaw/assets
```

  should point to the asset_base_dir as found in tactic_win32-conf.xml

```
<asset_base_dir>C:/ProgramData/Southpaw/assets</asset_base_dir>
```

- If you want to re-run install.py, the installation will ask if you to confirm backing up the sthpw database before dropping it. If you choose to do it yourself, here is the command:

```
pg_dump -c -U postgres sthpw > sthpw_backup.sql
dropdb -U postgres sthpw
```

- If you try to connect to the http://<server IP>/tactic and it just times out, your server firewall could be blocking access.
- If the Windows TACTIC Service has trouble starting up, you can try to edit the Properties , go to Log On tab, and adjust the Log on as user with the Windows login which has write/read access to the tactic_temp location as well as the asset_base_dir location.

2.8 Install TACTIC - Fedora 8

2.9 How to Install TACTIC on Fedora 12

Below are the step-by-step instructions of how to install TACTIC on Fedora 12.

Fedora 12

There are no known issues with running a TACTIC server on Fedora 12.

Southpaw Technology has chosen to distribute their virtual machine for evaluation on Fedora 12.

1. Log in as the root user.
2. Open the passwd file.

   ```
   vi /etc/passwd
   ```

   Modify the apache home directory and login shell to look like the following:

   ```
   apache:x:48:48:Apache:/home/apache:/bin/bash
   ```

3. Set password for apache to: south123paw

   ```
   passwd apache
   ```

4. Open to the sudoers file.

   ```
   visudo
   ```

   Add the apache user by including the following line in the appropriate location in the file:

   ```
   apache ALL=(ALL) ALL
   ```

5. Create the home directory for apache.

   ```
   mkdir /home/apache
   chown apache:apache /home/apache
   chmod a+rx /home/apache
   ```

6. Disable SElinux.

   ```
   vi /etc/selinux/config
   ```

7. Install modules.

   ```
   yum install -y gcc zlib-devel samba libxslt-devel libxml2-devel postgresql-server ↩
       postgresql-devel
   ```

8. Disable firewall.

   ```
   /etc/init.d/iptables save
   /etc/init.d/iptables stop
   ```

9. Create and open the index.html for redirection.

   ```
   vi /var/www/html/index.html
   ```

   Insert the following contents:

   ```
   <META http-equiv="refresh" content="0;URL=/tactic">
   ```

10. Re-login as the apache user.
11. Download the TACTIC source code and setup the service.

    Open the following link in a web browser

    ```
    http://support.southpawtech.com/downloads
    ```

    Setup the TACTIC service.

    ```
    cd /tmp
    unzip tactic_#.#.#.#.zip
    sudo cp /tmp/tactic_#.#.#.#/src/install/service/tactic /etc/init.d
    sudo chmod 775 /etc/init.d/tactic
    sudo /sbin/chkconfig tactic on
    ```

12. Setup Postgres.

    ```
    sudo /etc/init.d/postgresql start
    sudo cp /tmp/tactic_#.#.#.#/src/install/postgresql/pg_hba.conf /var/lib/pgsql/data
    sudo chown postgres:postgres /var/lib/pgsql/data/pg_hba.conf
    sudo /sbin/chkconfig postgresql on
    sudo /etc/init.d/postgresql restart
    ```

13. Setup Apache.

    ```
    sudo cp /home/apache/tactic_data/config/tactic.conf /etc/httpd/conf.d/
    sudo /sbin/chkconfig httpd on
    sudo /etc/init.d/httpd start
    ```

14. Install TACTIC.

    ```
    cd /tmp/tactic_#.#.#.#/src/install
    sudo python install.py
    sudo /sbin/chkconfig tactic on
    sudo chown -R apache:apache /home/apache/tactic /home/apache/assets /home/apache/ ↩
        tactic_data /home/apache/tacticTemp
    ```

15. Upgrade the database.

    ```
    python /home/apache/tactic/src/bin/upgrade_db.py
    ```

16. Startup TACTIC in dev mode.

    ```
    python /home/apache/tactic/src/bin/startup_dev.py
    ```

17. Try accessing TACTIC through a web browser on a client machine.
18. Once startup_dev works, Ctrl^C out of the process.

    ```
    Ctrl^C
    ```

19. Start TACTIC as a service.

    ```
    sudo /etc/init.d/tactic start
    ```

End of installation instructions.

2.10 How to Install TACTIC on Ubuntu 12

How To Install TACTIC 3.8 (Open Source Release) On Ubuntu Server 12.04

Revision 2 - August 12th, 2012

Install Ubuntu Server 12.04 as a VM ... go ahead with the default creation of initial non-root user (for example when using Easy-install for a VMWare VM OS installation), just to expedite the install process for Ubuntu Login as the non-root user that was created, and then activate root user account using (provide non-root user's password for sudo, then provide a password, and confirm it, for the root user): `$ sudo passwd root` Logout of non-root user account and then login with the newly activated root user account, using the password you just provided for the root user. Install necessary tools ... A. install gcc, make, unzip ... `$ apt-get install gcc make unzip` B. install Image Magick ... `$ apt-get install imagemagick` C. if needed, install OpenSSH server ... `$ apt-get install openssh-server` D. install other necessary modules ... `$ apt-get install samba libxslt1-dev libxml2-dev` E. install postgres ... `$ apt-get install postgresql-9.1 postgresql-server-dev-9.1` F. install Apache ... `$ apt-get install apache2p` Install Python 2.7 dependencies (NOTE: Ubuntu Server 12.04 comes with Python 2.7.3 pre-installed as its default Python) ... A. install PIL module ... (a) install library dependencies ... `$ apt-get install libjpeg-dev libpng-dev zlib1g-dev liblcms1-dev python-dev` (b) download and build the jpeg decoder ... (i) download the http://www.ijg.org/files/jpegsrc.v8d.tar.gz file into /root/temp (this file is provided in the "for_PIL" folder of this install guide package) (ii) unarchive the package file ... `$ cd /root/temp ` `$ tar xvzf jpegsrc.v8d.tar.gz ` (iii) build the jpeg decoder and install into the system ...`$ cd /root/temp/jpeg-8c ` `$ ./configure --enable-shared ` `$ make $ sudo make install ` (c) download PIL 1.1.7 source kit from http://effbot.org/downloads/Imaging-1.1.7.tar.gz (see more info at http://www.pythonware.com/products/pil/) and place in /root/temp (NOTE:this file is provided in the "for_PIL" folder of this install guide package) (d) set-up PIL ...`$ cd /root/temp ` `$ tar xvzf Imaging-1.1.7.tar.gz ` `$ cd Imaging-1.1.7 $ vi setup.py` ... look for declarations of JPEG_ROOT and ZLIB_ROOT and replace them as follows ... JPEG_ROOT = "/usr/local/include" ZLIB_ROOT = "/usr/local/include" ... the write the file and exit vi (e) install PIL ... `$ python setup.py build_ext -i ` `$ python selftest.py ` `$ sudo python setup.py install` B. install other needed Python modules ... `$ apt-get python-pythonmagick python-psycopg2 python-crypto python-pycryptopp python-simplejson python-lxml ` Create "apache" user and group ...A.Add the "apache" group (as root user) ...`$ groupadd --gid 48 apache ` B.Add entry for "apache" user in the /etc/passwd file ...$ vi /etc/passwd ...then add the following line to the file and write the file and exit ...apache:x:48:48:Apache:/home/apache:/bin/bash C.Set the "apache" user password ...`$ passwd apache ` D.Give apache user sudo privileges ...$ visudo ...and add this line in the appropriate place:`apache ALL=(ALL) ALL` ... then ctrl+O to write file (overwrite /etc/sudoers directly) and then ctrl+X to exit E. Create apache home directory ... `$ mkdir /home/apache $ chown apache:apache /home/apache $ chmod a+rx /home/apache` Set-up web-site index page for redirection ... `$ vi /var/www/index.html` ... replace its contents with: `<META http-equiv="refresh" content="0;URL=/tactic">` ... write the file to save and exit vi, then put it into an "html" sub-folder in /var/www ... `$ mkdir /var/www/html ` `$ cp /var/www/index.html /var/www/html/` Logout the root user then login as the "apache" user Create a "install_packages" folder in /home/apache Download the latest open source "Enterprise" release of TACTIC ... go to the downloads page of the Southpaw community site at: `"http://community.southpawtech.com/downloads"` ... and click "Download" for the "Enterprise" release. Copy this release package (e.g. tactic-3.8.0.rc02) to the /home/apache/install_packages folder Unarchive the release package ... `$ cd /home/apache/install_packages ` `$ unzip tactic-3.8.0.rc02.zip` ... you should now have the folder /home/apache/install_packages/tactic-3.8.0.rc02/ Set-up TACTIC to run as a service ... A. Modify service script to run on Ubuntu Server ... `$ cd /home/apache/install_packages/tactic-3.8.0.rc02/src/install ` `$ vi service/tactic ` ...modify the first line of that file to use the "bash" shell explicitly:`#!/bin/bash` ... then write out the file and exit vi B. Copy the service script to the proper location ... `$ sudo cp service/tactic /etc/init.d ` `$ sudo chmod 775 /etc/init.d/tactic ` Configure Postgres for use with TACTIC ...`$ sudo cp /etc/postgresql/9.1/main/pg_hba.conf /etc/postgresql/9.1/main/pg_hba.conf--ORIG ` `$ sudo cp /home/apache/install_packages/tactic-3.8/src/install/postgresql/pg_hba.conf /etc/postgresql/9.1/main/` `$ sudo chown postgres:postgres /etc/postgresql/9.1/main/pg_hba.conf ` `$ sudo chown postgres:postgres /etc/postgresql/9.1/main/pg_hba.conf--ORIG ` `$ sudo /etc/init.d/postgresql restart ` Configure Apache for use with TACTIC ...`$ sudo cp /home/apache/install_packages/tactic-3.8.0.rc02/src/install/apache/tactic.conf /etc/apache2/conf.d/` `$ sudo a2enmod proxy_http ` `$ sudo a2enmod proxy_balancer ` `$ sudo a2enmod rewrite ` `$ sudo service apache2 restart ` Se

tup bootstrap TACTIC module in Python install area (THIS STEP MUST BE DONE FOR TACTIC I NSTALL TO WORK) ...A.Manually create "tacticenv" folder in "/usr/lib/python2.7/dist-pac kages/" ...`$ sudo mkdir /usr/lib/python2.7/dist-packages/tacticenv ` `$ sudo chmod 755 /usr/lib/python2.7/dist-packages/tacticenv ` B.Revise the "tactic_paths.py" file to co ntain the proper paths ...`$ vi /home/apache/install_packages/tactic-3.8.0.rc02/src/ins tall/data/tactic_paths.py ... add the following lines (no indentation!) to the bottom of the file (if not already present): TACTIC_INSTALL_DIR='/home/apache/tactic' ` `TACTIC_SITE_DIR='' ` `TACTIC_DATA_DIR= '/home/apache/tactic_data' ... then write those changes to the file and exit vi C. Manually copy the "*init*.py" and "tactic_paths.py" files into this new "tacticenv" module folder ... $ sudo cp /home/apache/install_packages/tac tic-3.8.0.rc02/src/install/data/*.py /usr/lib/python2.7/dist-packages/tacticenv/` `$ s udo chmod a+r /usr/lib/python2.7/dist-packages/tacticenv/*.py Run TACTIC install script ... $ cd /home/apache/install_packages/tactic-3.8.0.rc02/src/install $ sudo python install.py ... NOTE: specify "/home/apache" for the TACTIC installation base path and specify "apache" for the user for the Apache Web Server. $ sudo chown -R apache:apache /home/apache/tactic /home/apache/tactic_* ` `$ sud o chown -R apache:apache /home/apache/assets /home/apache/tactic_data ` Copy the provi ded TACTIC license file to the correct location ...`$ cp /home/apache/tactic/src/install/ start/config/tactic-license.xml /home/apache/tactic_data/config/` Ensure databases are upgraded to the given version of TACTIC ...`$ python /home/apache/tactic/src/bin/upgra de_db.py Test to make sure TACTIC server is able to run ... $ python /home/apache/tactic/src/bin/startu p_dev.py ... and then, from a browser, go to the TACTIC web UI and confirm everything is functioning properly Finally set up TACTIC to automatically start up as a service when the server is rebooted ... $ sudo update-rc.d tactic defaults

2.11 Upgrade TACTIC

Upgrading TACTIC is usually a relatively simple procedure:

1. Download the new release of TACTIC from the Download section of support.southpawtech.com . TACTIC is delivered as a zipped (.zip) file.
2. Copy the new version to the installed directory of TACTIC and unpack it. e.g.

   cd /home/apache/

   unzip tactic-2.6.0.v01.zip 3. Stop the TACTIC service

   /etc/init.d/tactic stop 4. Switch the symbolic link. See explanation on symbolic link below. 5. In a cmd shell, cd to the bin directory of the new installation:

   cd /home/apache/tactic/src/bin 6. Back up the current database:

   pg_dumpall -U postgres -c > database_<date>.sql 7. Run the upgrade script. The version number argument (for example, 2.6.0.v01) is optional.

   python upgrade_db.py

   First, you will be prompted to confirm whether the displayed version is the right one to upgrade to.

   On proceeding, it will check the upgrade script's version tree against the currently installed TACTIC version (the new one).

   Normally, they should be the same. In case they are not, you will be prompted again to confirm if this is actually intended. Warning: if the versions do not match, it is very likely you have not stopped the old version from running or have not symlinked "tactic" to the new version yet. 8. Restart the TACTIC service:

   /etc/init.d/tactic start 9. Go to the "Projects" view in the Admin site. 10. Optional: Compare each project's schema with the one used by TACTIC.

Explanation of switching symbolic links

By default, in your python installation, like /usr/lib/python2.5/site-packages/tacticenv/tactic_paths.py it defines the TACTIC_INSTALL_ = */home/apache/tactic* and so a convenient way to upgrade is to untar the new release in /home/apache/ as e.g , /home/apache/tactic-2.5.0.rc04 cd /home/apache then delete any existing symbolic link named "tactic" by running rm tactic. Afterwards, you can create a new symlink using ln -s tactic-2.5.0.rc04 tactic With the tactic service stopped, and the TACTIC_INSTALL_DIR practically pointing to the new release because of the symlink, now you can safely run src/bin/upgrade_db.py in the new release.

1. Download the new release of TACTIC from the Download section of support.southpawtech.com . TACTIC is delivered as a zipped (.zip) file.
2. Copy the new version to the installed directory of TACTIC and unzip it.
3. Stop the TACTIC service

   `Run services.msc`

   `Select the service Tactic Server, click on the Stop button` 4. Rename the top directory of the new version to TACTIC. 5. In a cmd shell, cd to the bin directory of the new installation:

   `cd C:\Program Files\Southpaw\Tactic\src\bin` 6. Back up the current database:

   `pg_dumpall -U postgres -c > database_<date>.sql` 7. Run the upgrade script. The version number argument (for example, 2.6.0.v01) is optional.

   `python upgrade_db.py`

   First, you will be prompted to confirm whether the displayed version is the right one to upgrade to.

   On proceeding, it will check the upgrade script's version tree against the currently installed TACTIC version (the new one).

   Normally, they should be the same. In case they are not, you will be prompted again to confirm if this is actually intended. Warning: if the versions do not match, it is very likely you have not stopped the old version from running or have not symlinked "tactic" to the new version yet. 8. Restart the TACTIC service:

   `Select the service Tactic Server, click on the Start button` 9. Go to the "Projects" view in the Admin site. 10. Optional: Compare each project's schema with the one used by TACTIC.

2.12 Multiple TACTIC Services

Multiple versions of TACTIC can be easily run side by side on the same machine. This is extremely useful for testing purposes. Running a newer version of TACTIC on a production server machine using the production server environment and production database. You can run this without interrupting service ot the production server.

If this is not explicitly set, TACTIC will use whatever settings exist when TACTIC was first installed. These are defined in the tacticenv module which is first imported on all command line TACTIC scripts by:

```
import tacticenv
```

These values are typically found in the Python distribution site-packages. The exact location of these files can vary between differing operarting systems, but common examples are:

```
windows: C:\Python27\site-packages\Lib\site-packages\tacticenv
```

```
linux: /usr/lib/python2.7/site-packages
```

When TACTIC starts up, it will first look at the environment variable TACTIC_INTALL_DIR to determine which distribution of TACTIC will be used. Before a newer version of TACTIC is executed, it is necessary to upgrade to the newer database schema. Unless clearly stated in the distribution release, it is always safe to upgrade the database for a newer version. Changes to the database are always backwards compatible, so that older versions of TACTIC can always run on newer schema of the database.

Steps

1. Unzip a TACTIC distribution anywhere in the file system. For purposes of examples, /home/apache will be used as the base directory for TACTIC

   ```
   cd /home/apache
   unzip tactic-3.0.0.zip
   ```

2. Set the environment variable TACTIC_INSTALL_DIR

   ```
   export TACTIC_INSTALL_DIR=/home/apache/tactic-3.0.0
   ```

3. Upgrade the database.

```
su apache
cd tactic-3.0.0.src/bin
python upgrade.py
```

4. Execute tactic

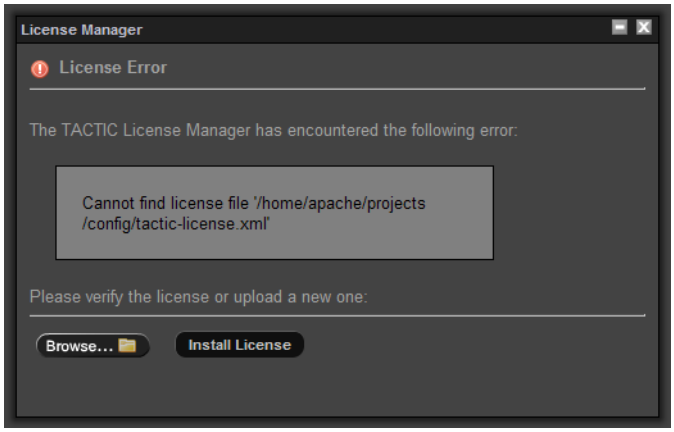
```
python startup_dev.py
```

This will execute TACTIC from the command line in dev mode. All output will go to this console.

2.13 Install a TACTIC License

Install License

When TACTIC is accessed through the web browser, if a license file cannot be found, TACTIC will prompted for a license file.



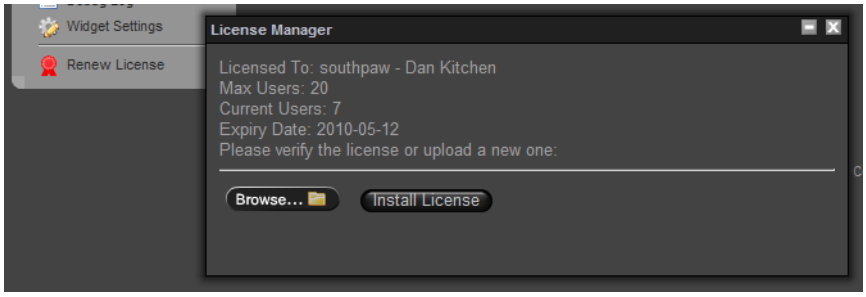
To install a license file, browse for the license file (.xml) provided by Southpaw Technology. Once the file has been selected for upload, click **Install License** to upload and install the license file.

If the license file is valid, TACTIC will display the message "License renewal completed"

Once this has been completed, TACTIC will no longer prompt for a valid license file.

Renew License

To view information on the total users for the license and how many are active and to update the existing license file, in the sidebar, load **Global** → **License** From here you view the License statistics and update a new license file.



Manual License Install

When the user interface tool is used to upload the license, TACTIC renames the file to tactic-license.xml and places it on the server under

```
<TACTIC_DATA_DIR>/config/tactic-license.xml
```

The XML file provided from the TACTIC representative can be renamed. After the rename, place the license on the server according to the structure outlined above and re-browse to tell TACTIC the new license name.

3 Install Python

3.1 Python Installation

Python is a core element of the TACTIC application. The TACTIC application has been developed around the the python programming language.

The installations of both python and supporting modules is a reqirement in a successful installation of TACTIC. Although the TACTIC core files can exist on the host machine, it is the existance of both Python and the supporting modules.

Python 2.5 is the defacto standard for deployments of the TACTIC application. 2.6 and 2.7 are supported as well. Currently, Python 3.0 is unsupported.

The installation of core Python is not enough for the successful installation of TACTIC. Several modules must be installed. These modules provide the following functions:

- Extensive **XML support**
- Connectivity to **DB co-service**
- *Win32*support on windows
- **Cryptography** support
- **JSON** support
- Imaging support

The TACTIC application can support different different Python Supporting Modules, depending on the version of TACTIC.

There are a number of variables that will affect which modules will be required to install on a host machine. These considerations are;

- DB Co-service deployed
- Desired support module deployed when multiple support module types exist.

Support Module Type	Module name	TACTIC supported version
XML support	lxml	All
DB Co-Service Support	Psycopg2 or CX-Oracle (psycopg2 is preferred for postgres database set-up)	All
Win32 (used in client computers interacting with the server)	PyWin32	All
Cryptography	PyCrypto	All
JSON (not needed since Python 2.6)	SimpleJSON	All
Imaging	PIL and ImageMagick (optional)	All

On Windows systems, TACTIC Python and supporting module availability is through the websites of the supporting module projects. Additionally, Southpaw Technology supports redistribution of installers for these modules on the support web site.

On UNIX/Linux systems, the system package manager typically has options for installing all the required Python packages onto the UNIX host machine. To find the official distribution of a supporting package, the search functionality of a package manager is useful.

```
#/ yum search lxml
```

4 Install Web Server

4.1 Apache HTTP Co-Service Configuration

Installation of Apache can be done through any number of means, depending on the software package on hand, and the target OS used. Please consult the Apache documentation for installation information.

All steps require use of a command shell. The example posted below is working as root user, but some OS's are not enabled for root user access. The root user may have do be enabled on the target machine, otherwise commands must be prepended with "sudo", or whatever method required for editing service files, starting and stopping services, etc.

The httpd.conf file (sometimes alternatively, the apache2.conf file) is the configuration file for Apache. Locate this file on the target machine. For purposes of this illustration, the httpd.conf file name will be used.

```
leowiz:~ root# locate httpd.conf
/private/etc/apache2/httpd.conf
/private/etc/apache2/original/httpd.conf
leowiz:~ root#
```

Alternatively "find" will do the same thing;

```
leowiz:~ root# find / -name httpd.conf
/private/etc/apache2/httpd.conf
/private/etc/apache2/original/httpd.conf
leowiz:~ root#
```

In this case, the "/private/etc/apache2/httpd.conf" file is the one we are interested in.

For isolation of configuration options for editing purposes, the TACTIC configuration file is a single file which should be referred to by the Apache configuration file. In this case, the default "tactic.conf" file will be used. The file is included at the end of this document. Note this default "tactic.conf" has already be altered during the running of tactic installation install.py so that the path matches the installation directories.

Find out if the httpd.conf file has an "Include" statement that refers to a directory the TACTIC Apache configuration file can be put in.

```
leowiz:~ root# grep Include /private/etc/apache2/httpd.conf
Include /private/etc/apache2/extra/httpd-mpm.conf
#Include /private/etc/apache2/extra/httpd-default.conf
#Include /private/etc/apache2/extra/httpd-ssl.conf
Include /private/etc/apache2/other/*.conf
leowiz:~ root#
```

In this case, the line at the bottom is the one that is required;

```
Include /private/etc/apache2/other/*.conf
```

If there is no such include, then a line can be added to the httpd.conf file.

In the above example, there is a reference to a directory wild-card configuration inclusion. Essentially then, any files with the suffix ".conf" will be activated. This is where the TACTIC configuration file will be stored. Either the default configuration file can be used with some editing, due to the variations in location of the TACTIC service application.

For purposes of efficiency, Apache will proxy files that are static. In the tactic.conf file, there are 2 major directives that are of concern regarding directory access. There is the directives that enable the TACTIC GUI widgets to work,

```
<Directory "/home/apache/tactic" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
```

The second set is:

```
<Directory "/home/apache/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
```

In the default example, the TACTIC application is stored at "/home/apache/tactic" while the assets are stored at "/home/apache/assets" The directives here allow Apache access to proxy these file locations for TACTIC.

Since Apache is serving only static files, it can be configured to serve the dynamic content served by TACTIC using a proxy.

To enable the Apache proxy service, the tactic.conf file must contain directives that enable the proxy and rewrite modules to serve the TACTIC service.

```
ProxyPreserveHost on
RewriteEngine on

# for cherrypy
RewriteRule   ^/tactic/(.+)$ http://localhost:8081/tactic/$1 [P,L]
RewriteRule   ^/tactic http://localhost:8081/tactic/ [P,L]
RewriteRule   ^/projects/(.+)$ http://localhost:8081/tactic/$1 [P,L]
RewriteRule   ^/projects http://localhost:8081/tactic/ [P,L]
```

In this example, the TACTIC service is located on the same machine, on port 8081. All URL requests that have a "/tactic" or "/projects" in the URL will be redirected to the TACTIC service, which by default is on port 8081.

To effectively use load-balancing on this machine, replace it with the following configuration instead:

```
# This is for using a random load_balancing scheme
RewriteMap    lb    rnd:/home/apache/sites/load_balance.txt
RewriteRule   ^/tactic/(.+)$ http://${lb:dynamic}/tactic/$1    [P,L]
RewriteRule   ^/projects/(.+)$ http://${lb:dynamic}/tactic/$1  [P,L]
RewriteRule   ^/tactic http://${lb:dynamic}/tactic/            [P,L]
RewriteRule   ^/projects http://${lb:dynamic}/tactic/          [P,L]
```

```
load balance
```

New Method

The use of balancer module is recommended for its reliablity and ease of use.

<Proxy balancer://tactic> BalancerMember http://localhost:8081/tactic BalancerMember http://localhost:8082/tactic BalancerMember http://localhost:8083/tactic </Proxy> ProxyPass /tactic balancer://tactic ProxyPass /tactic_data balancer://tactic

Old Method

The file, /home/apache/sites/load_balance.txt contains

```
lb  localhost:8081|localhost:8082|localhost:8083
```

The Apache configuration files have now been modified to proxy and rewrite TACTIC requests.

Apache needs to be restarted for the configuration changes to take effect. Most Apache installations have a "apachectl" command. To restart apache, use;

` apachectl restart`

Once Apache has been restarted, it should be serving TACTIC requests.

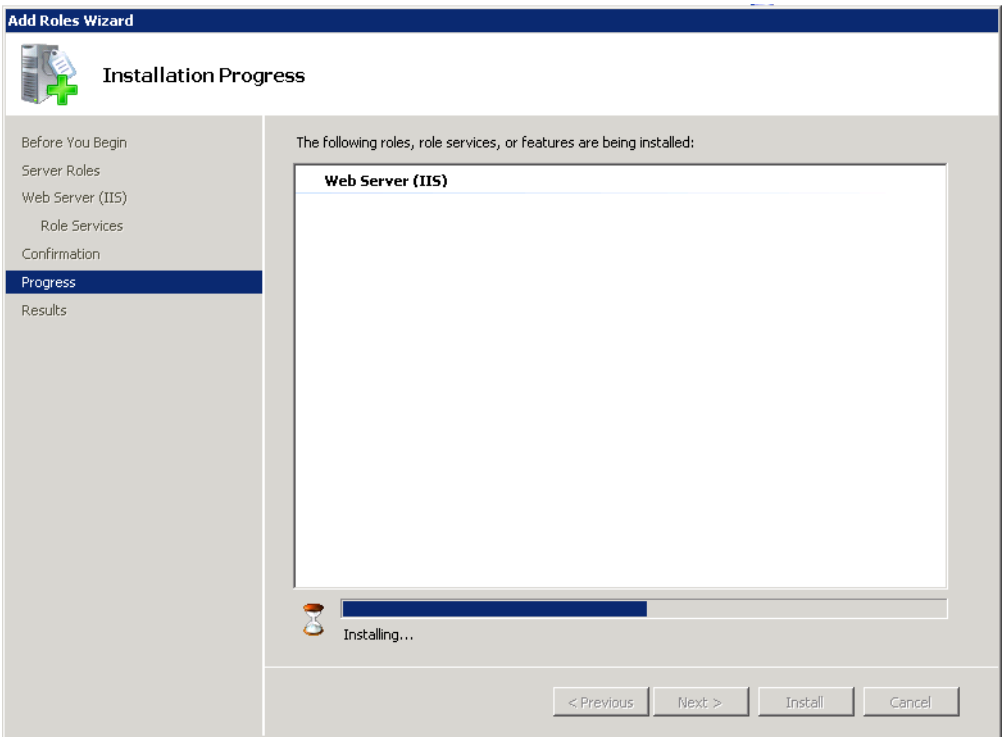
At this point Apache should be configured to proxy TACTIC service requests to the TACTIC service, while leaving all other requests to Apache itself.

4.2 IIS 7+ HTTP Co-Service Configuration

Windows HTTP co-service installations are IIS based.

On IIS installations prior to version 7, appropriate 3rd party proxying software (the ISAPI_Rewrite module) is required to be installed with IIS to enable the proper functioning of TACTIC.

TACTIC can be configured to work with the native IIS 7.5 services available on applicable Windows OS's, such as Windows 7 and Windows Server 2008. Prior versions of IIS have required 3rd party software (the ISAPI_Rewrite module) to be installed alongside the IIS service to facilitate proxy and URL rewriting. With IIS 7.5, and the introduction of the ARR and URL rewrite module, the requirement of a 3rd party software is no longer needed.



On Windows 7, and Windows Server 2008, IIS is installed through means and directions published on Microsoft support websites. Please consult installation guides published by Microsoft for the installation procedure of IIS.

Typically, IIS 7.5 does not come with the ARR module, and depending on the IIS installation, URL rewrite. These modules are requirements for a installation alongside TACTIC. Please refer to Microsoft published documentation regarding installation of these modules.

Permissions must be set for IIS to be able to serve assets that TACTIC manages.

TACTIC requires that the IIS user have access to TACTIC assets and all of the virtual directories that contain TACTIC widgets.

Whether the assets directory is stored on the host machine or on a network mount, there should be a assigned user that runs TACTIC.

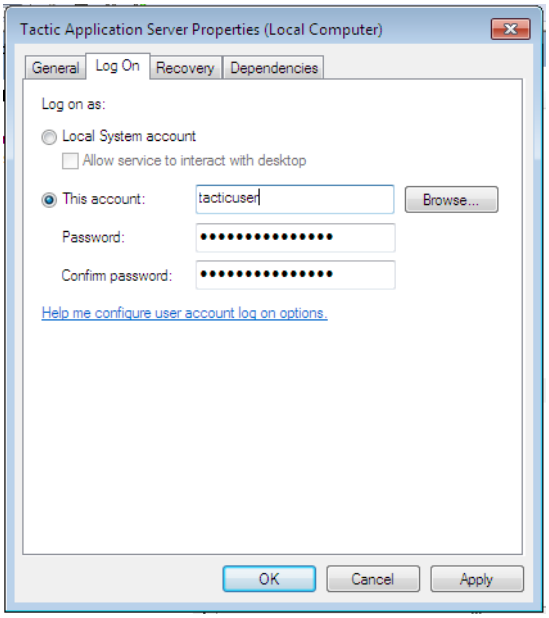
For non-domain Windows machines, a local user created on the TACTIC host, such as the IUSR_<computer_name> automatically created by an IIS installation will suffice, provided that the network mounted directories are writable by the system user.

For ADS domains, a user can be created on the domain, and assigned via the Services control panel. Assigning a domain user to the TACTIC service will allow domain level security rules to apply.

To assign a user to a service:

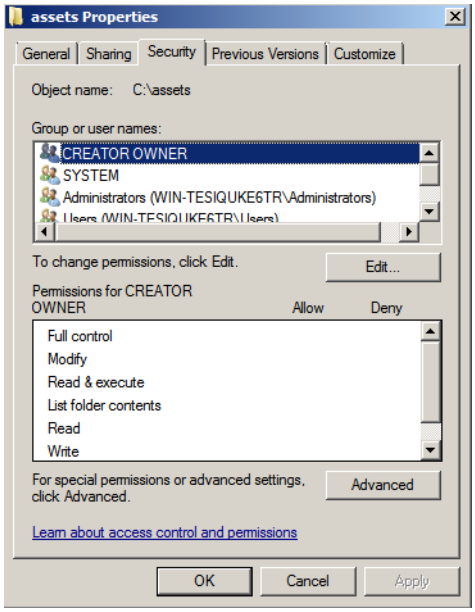
1. Open the Services control panel
2. Right click on the "TACTIC application server"
3. Click on the "Log On" tab

4. Change the "log on as" option to reflect the user created, whether local or domain based.



For a locally hosted asset directory , the anonymous user (typically IUSR_<computer name>) account needs to be granted access to the directories, and all subdirectories under them.

Find the directories above in Windows Explorer, and right click on the directory to bring up “Sharing and Security&#xfffd;? for that directory.



IIS must be configured to serve static content, such as TACTIC assets, and TACTIC UI skins.

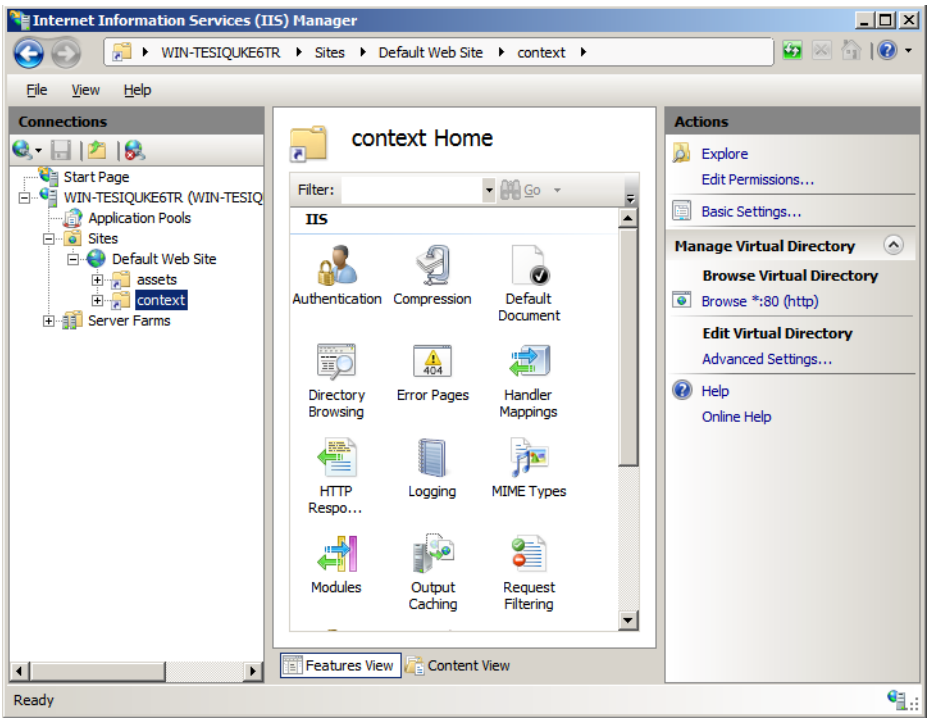


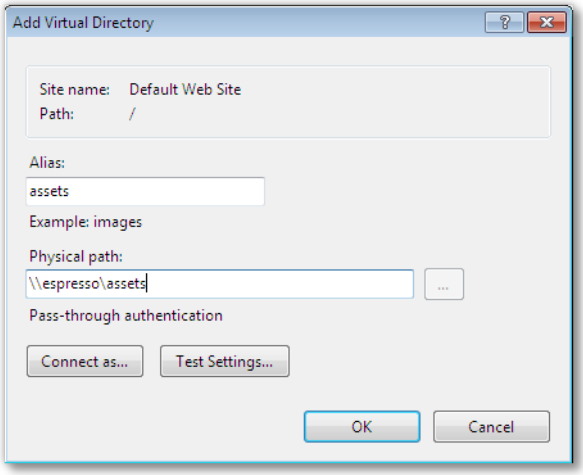
Table 5: Virtual directories required by TACTIC

Directory Description	Directory Alias	Location
Assets storage	assets	User provided during installation
TACTIC UI widgets	context	Inside TACTIC installation directory
Documentation	doc	Inside TACTIC installation directory
Project storage	projects	User provided during installation

There are four “virtual directories&#xfffd;? that must be created to access the static content provided by TACTIC.

To create the directories;

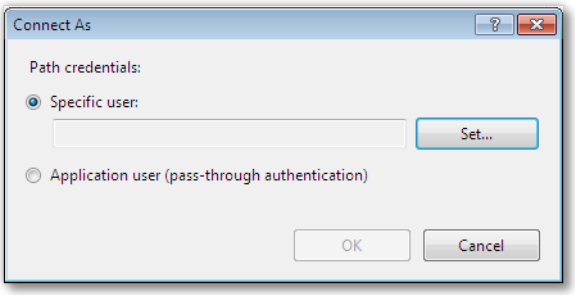
1. Right click on “Default web site&#xfffd;? and “Add Virtual Directory&#xfffd;?
2. Create the virtual directory, using the paths that were created by the installation of TACTIC. The default paths may not apply.



3.

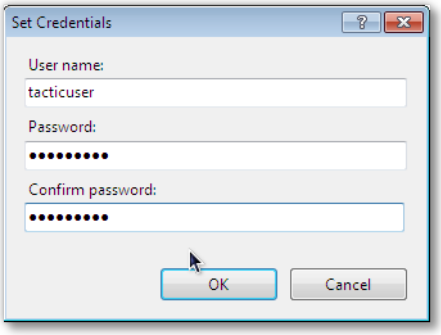
Important

If the directory is located on a network mount, it may have to be connected to as the user running the TACTIC service. To connect as a different user than the IIS user, click on the "Connect As"



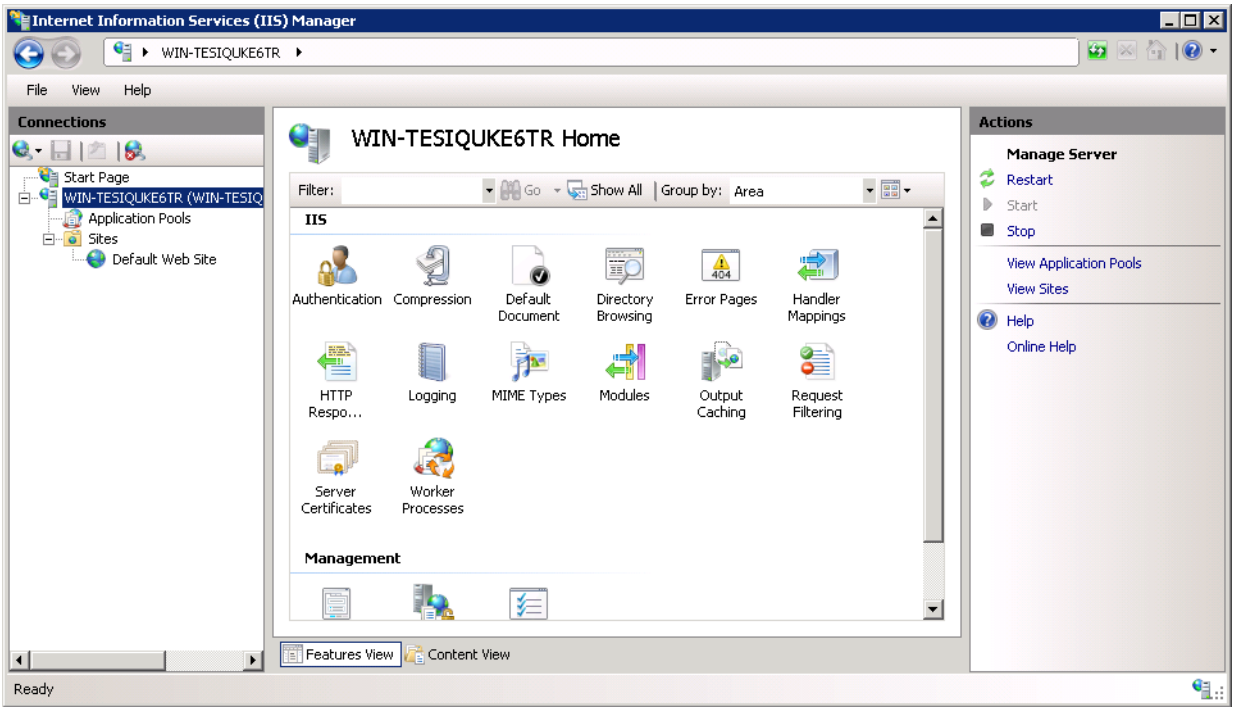
+

+ Then click on the "Set. . . " button, and fill in the details of the user that will run the TACTIC service.



+

There should now be 4 virtual entries on the IIS service.



Application Request Routing (ARR) is the module snap-in that will proxy and load balance requests. ARR is required by IIS to split incoming TACTIC service requests between the dynamic content that drives the API and the TACTIC web UI, and static content. In order to achieve this, IIS must be configured to send certain requests to the TACTIC application server, while static data requests (usually assets) are sent to IIS.

This guide currently supports two methods of attaching the TACTIC service to an IIS co-service.

Proxy/Load balancing

The Load balancing configuration of IIS splits TACTIC service requests into multiple streams, with each stream utilizing its own TACTIC service process.

Proxy Only

The proxy-only configuration routes all TACTIC service requests to a single TACTIC service. This method should only be used in light usage TACTIC installations such as development servers.

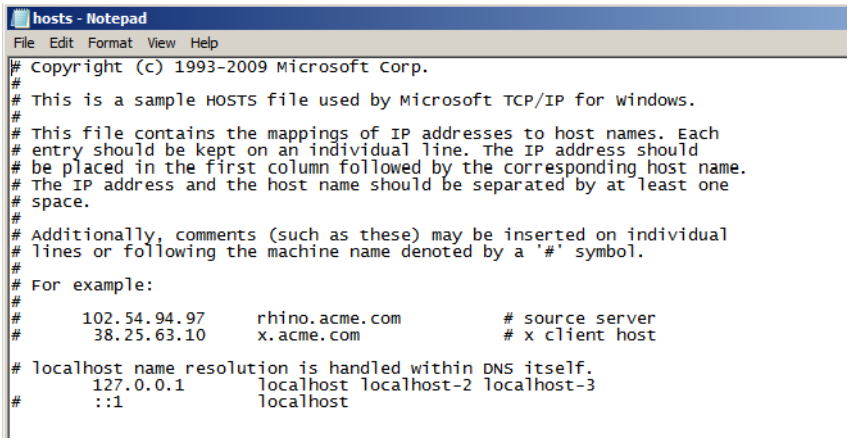
A TACTIC installation by default runs 3 seperate TACTIC service listeners, arranged on default ports 8081, 8082 and 8083. To split TACTIC service requests into multiple streams, IIS must be configured to consider TACTIC part of a “server farm&#xfffd;?. Since all of the TACTIC services are running on one single host, IIS currently must be tricked into routing requests to the 3 different TACTIC streams on the localhost.

By default, windows looks in the hosts file, then to DNS for named hosts. To divide localhost into three “different&#xfffd;? machines, the `"%SYSTEMROOT%/System32/drivers/etc/host&#xfffd;?` file must be edited to be able to address localhost as more than one machine.

> **Important**
>
> If TACTIC is set to run a number of processes other than the default of three, then these instructions must reflect that number. Add or subtract from the list of localhost entries and web farm entries accordingly.

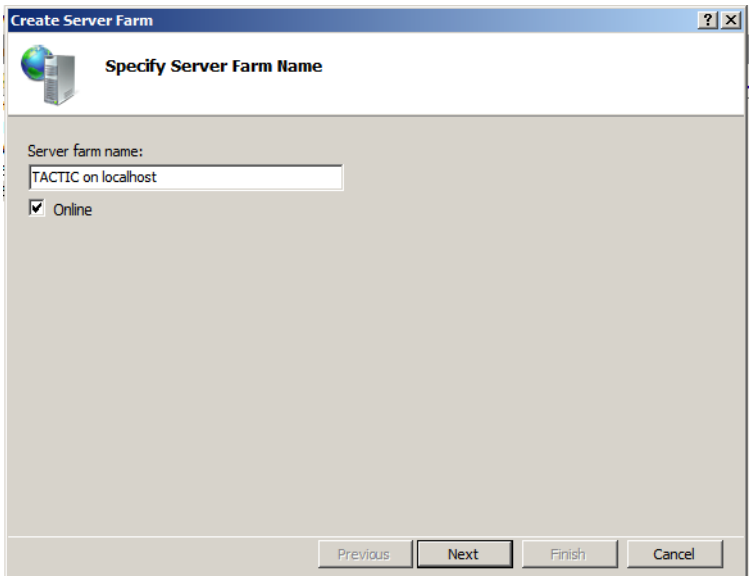
To create a web farm;

1. As an “Administrator&#xfffd;? user, edit the “127.0.0.1&#xfffd;? line in this file



   The addition of localhost-2 and localhost-3 that reference 127.0.0.1 allow this host machine to access localhost as more than one machine. 2. Now the Server Farm reference must be created.

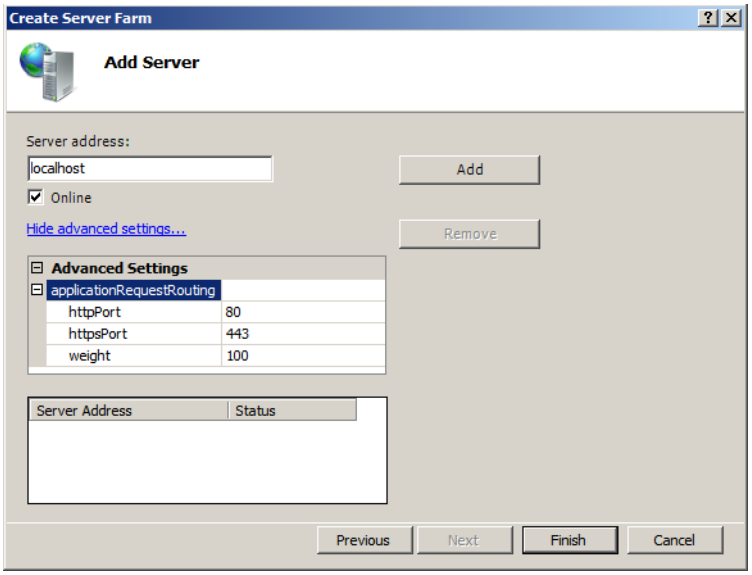
   Start the IIS snap in, and select the Web server that will be used as a TACTIC co-service in the left side navigation bar. Right click on the “Server Farms&#xfffd;? folder and create a new server farm.



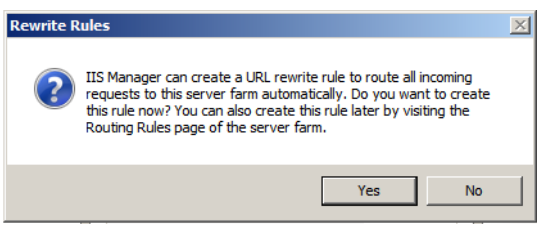
3 seperate servers must be added to the server farm to correspond to the 3 default TACTIC service processes. Add each server according to the below table.

Server Address	httpPort
localhost-1	8081
localhost-2	8082
localhost-3	8083

Click on “Add&#xfffd;? when done each entry in the above list. For additional servers, the “server address&#xfffd;? must correspond to the additional localhost entries in the “hosts&#xfffd;? file. The ports must correspond to 8082, 8083, etc. Add the required number of servers according to TACTIC deployment requirements.

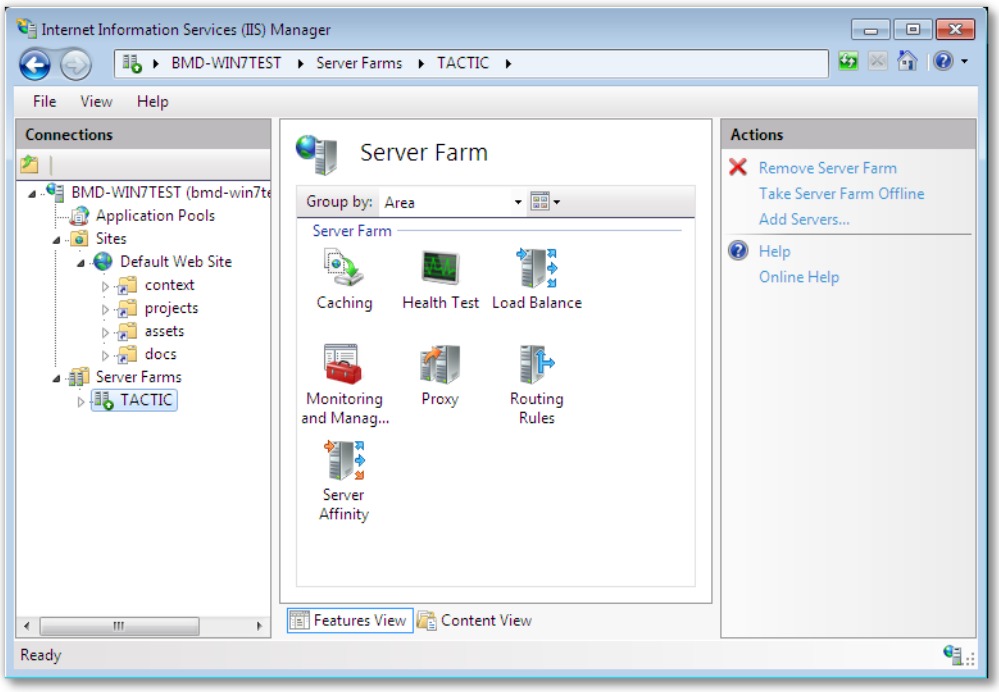


3. Click “Finish&#xfffd;? when done. 4. Create the rewrite rule when prompted.

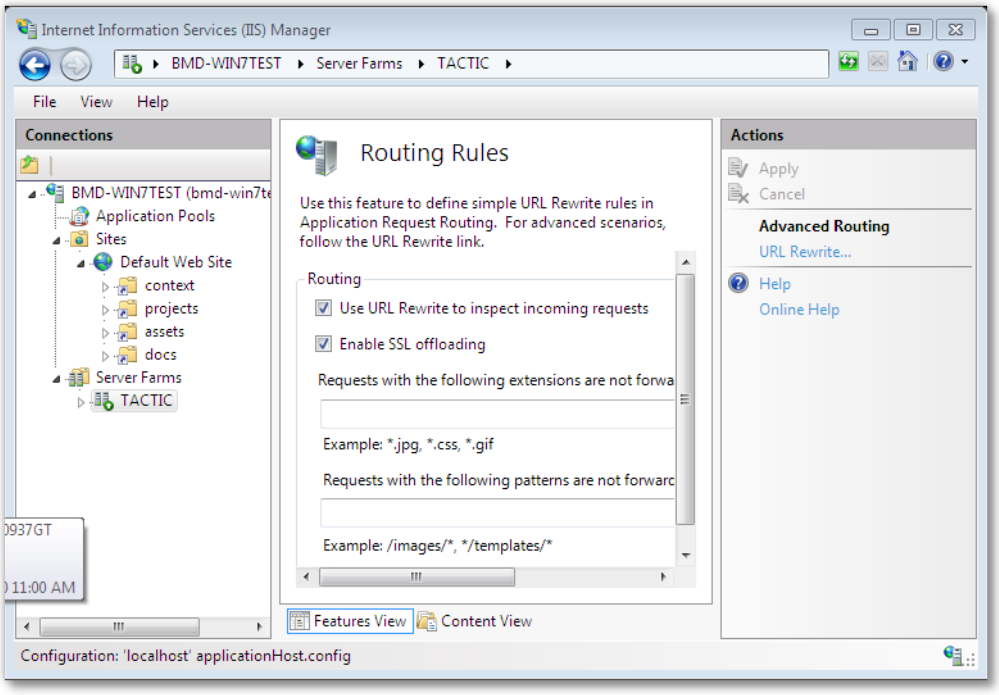


URL rewrite must now be configured to only send TACTIC API requests to the server farm.

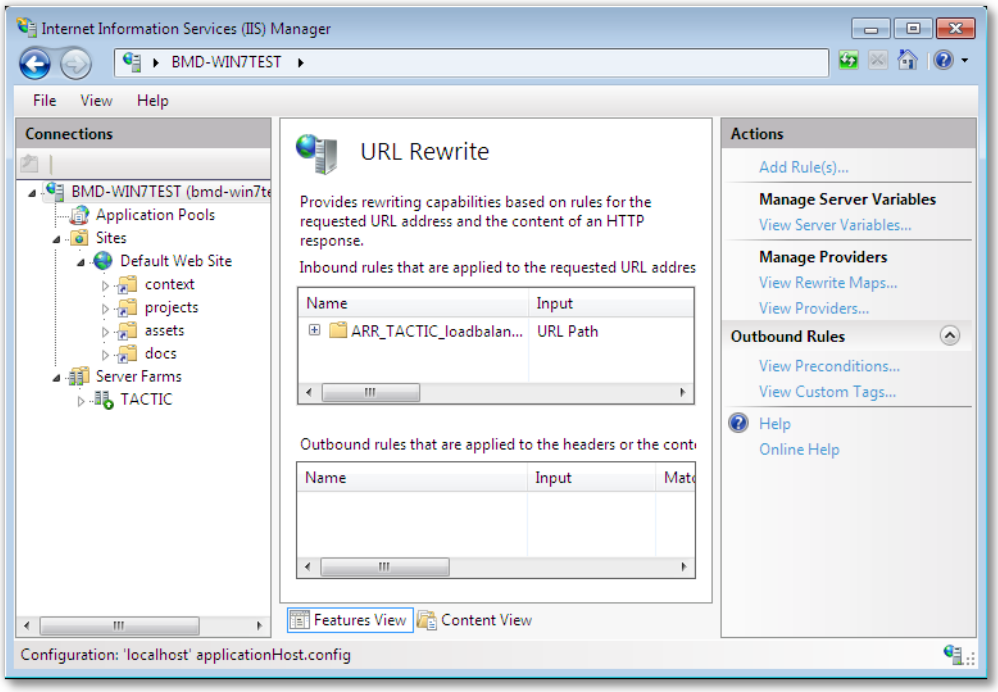
1. Click on the server farm icon on the left-hand side of IIS manager and click the “Routing Rules&#xfffd;? icon.



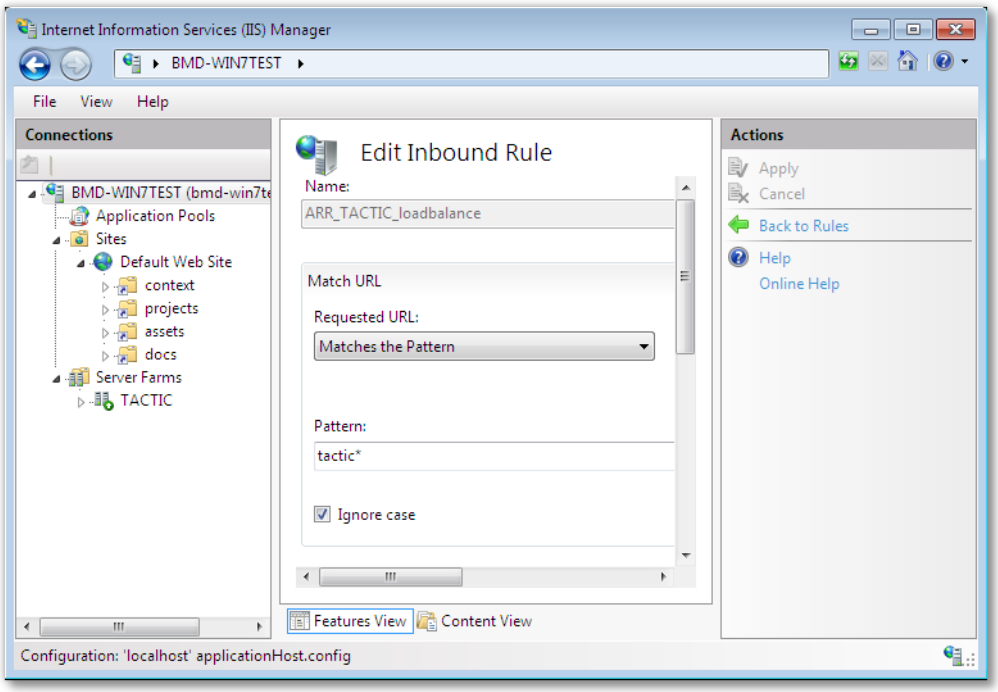
2. In the right pane of the Routing rules section, click the "URL rewrite" link



3. Select the automatically created URL rewrite rule created when the web farm was created, and click on the "Edit" link in the right pane.

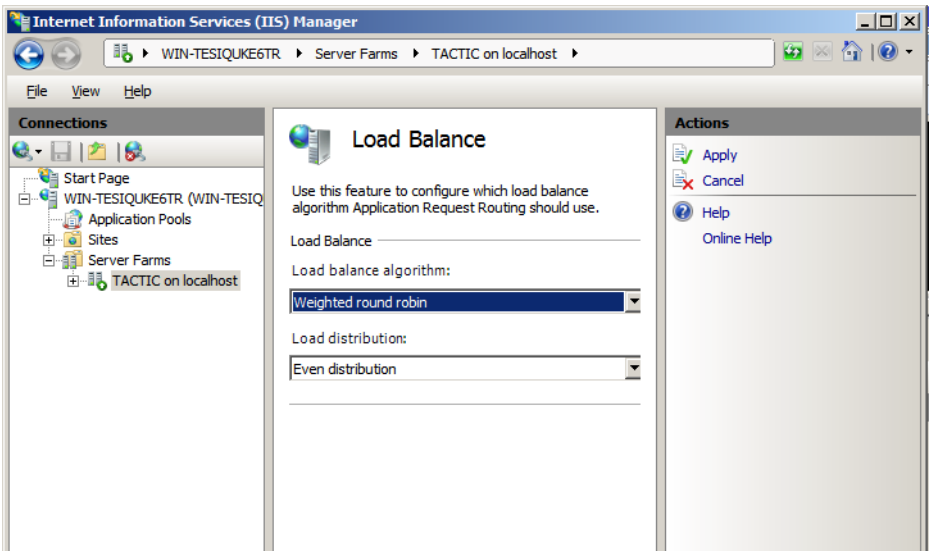


In the “Edit Inbound rule&#xfffd;? pane, edit the pattern to read “tactic*,projects*&#xfffd;? This will instruct IIS to route all TACTIC UI requests to the TACTIC service.



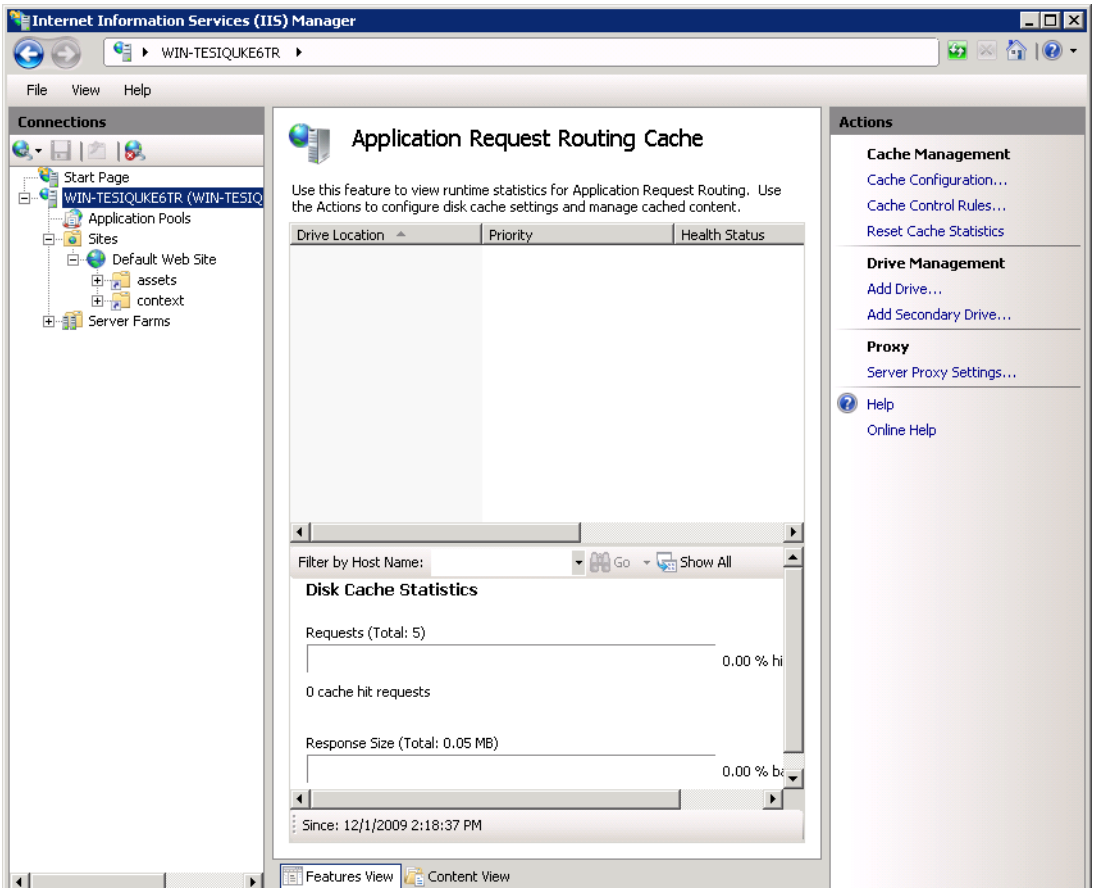
4. Apply the change.

ARR comes with several load balancing algorithms. Click on the newly created server farm, and click on the “Load Balancing&#xfffd;? icon. Select “Weighted round robin&#xfffd;? as the algorithm, and “Even distribution&#xfffd;? as the load distribution.

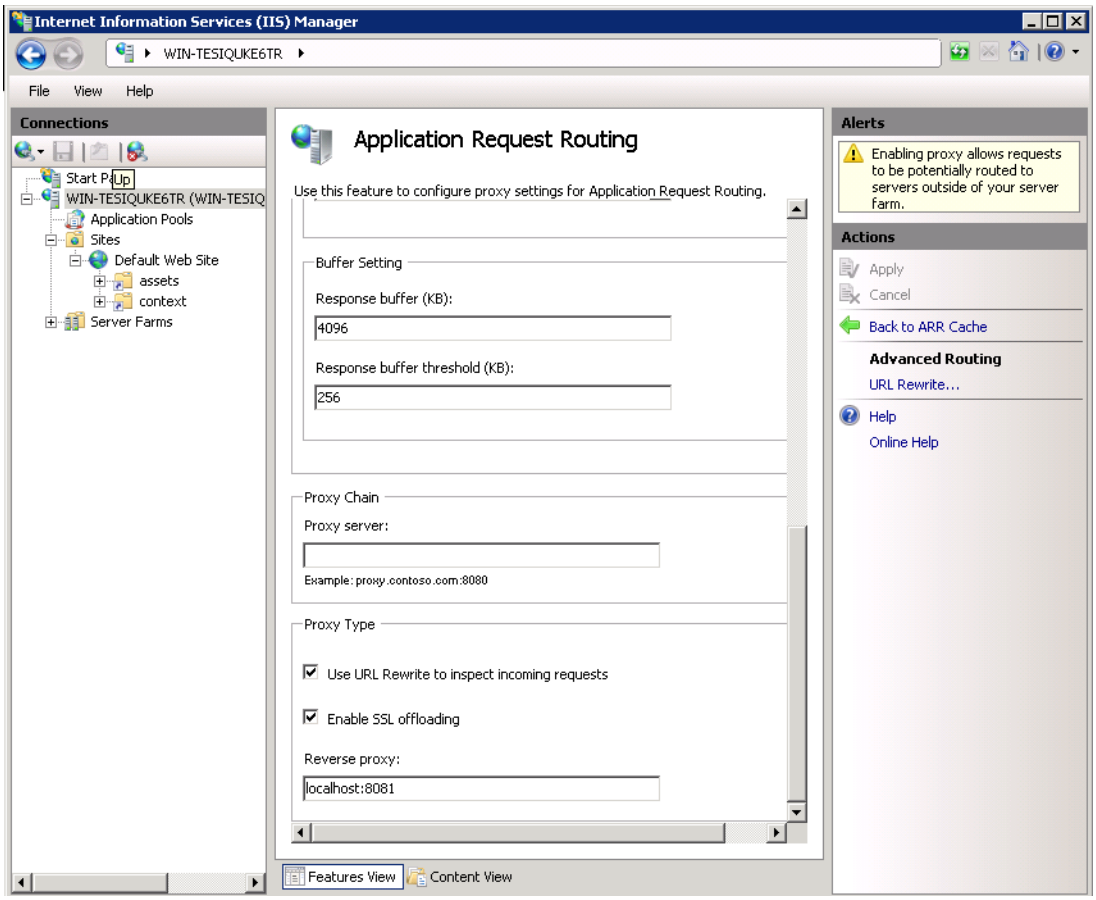


IIS can be configured to run a single rewrite rule to forward requests to a single TACTIC service.

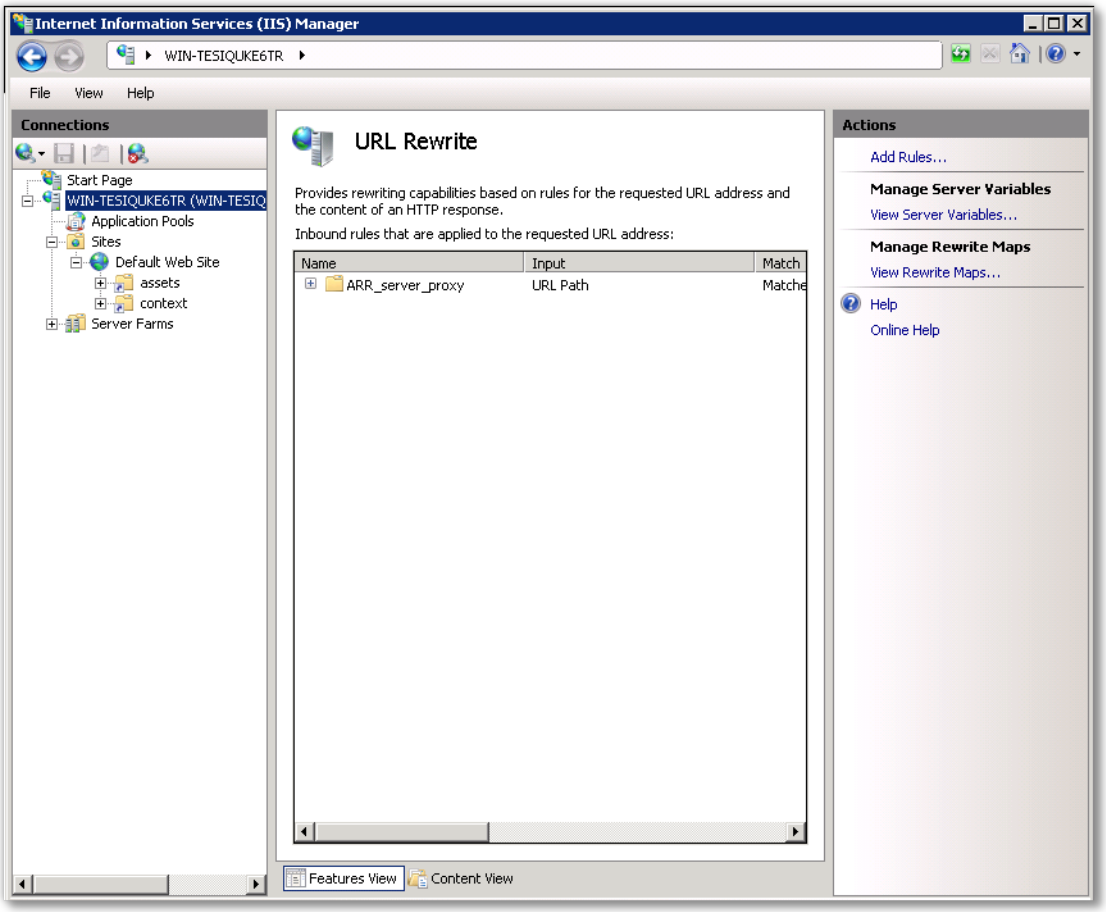
Start the IIS snap in, and select the Web server that will be used as a TACTIC co-service in the left side navigation bar. Choose “Application request routing&#xfffd;? And click on server proxy settings on the right-hand side.



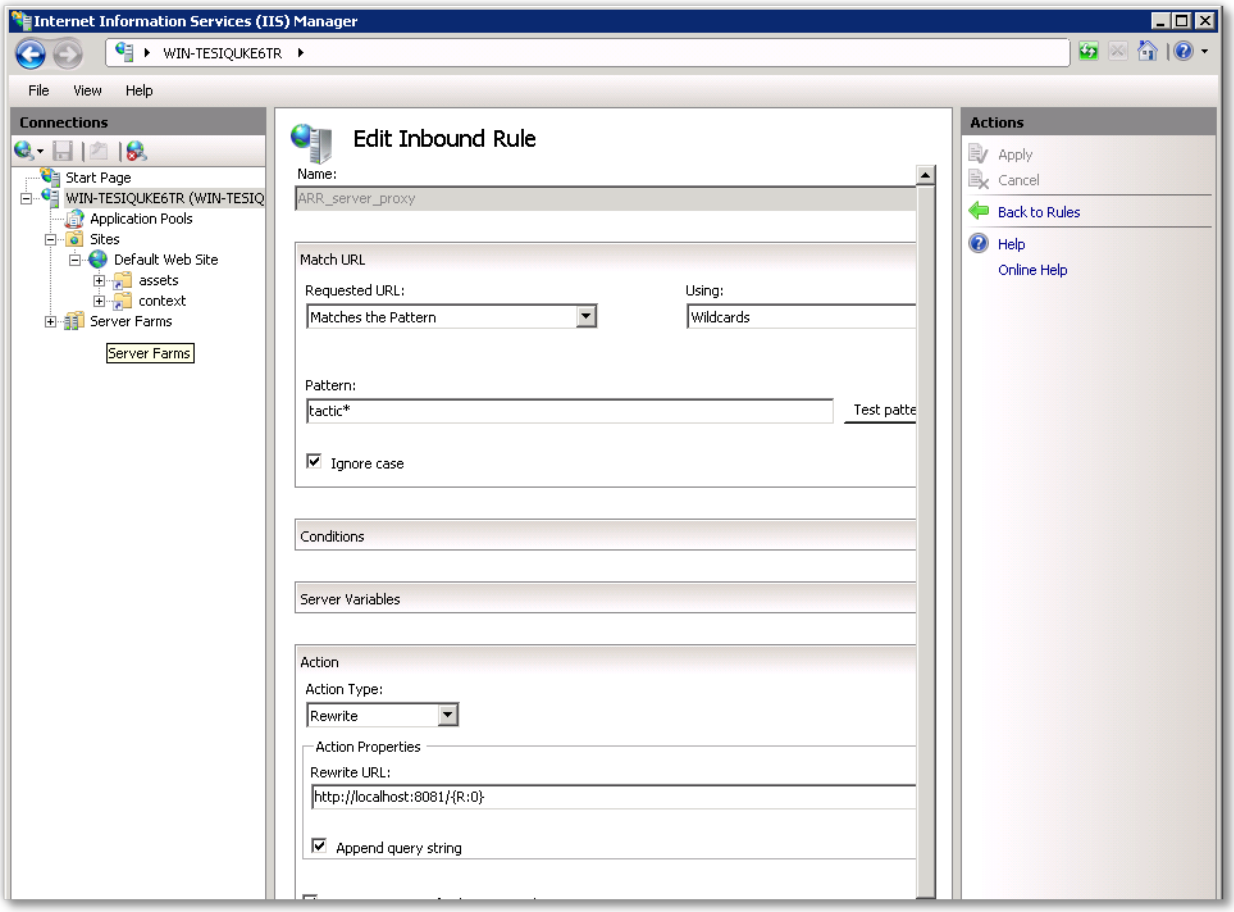
In the ARR options under the proxy settings check the “use URL rewrite to inspect incoming requests&#xfffd;?, And in the “reverse proxy&#xfffd;? text area type “localhost:8081&#xfffd;?. This instructs IIS to proxy everything to this address and port.



From here URL Rewrite can then be instructed to filter proxy requests. Click on the “URL rewrite&#xfffd;? link on the right-hand side to modify rewrite rules. Typically, if the proxy has been created in ARR, then a rule will be created in URL rewrite.



IIS must then be instructed to only proxy TACTIC UI and API requests. To do this, an automatically created rule must be altered to allow TACTIC asset requests to be handled by IIS.



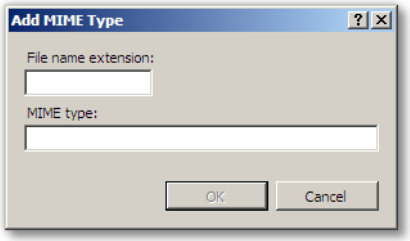
To modify the existing rule for TACTIC, the pattern under "Match URL" should be "tactic*". The action “Rewrite URL&#xfffd;? should be "http://localhost:8081/{R:0}"

TACTIC currently has a requirement to insert custom MIME types into the list of allowed types for the IIS service.

MIME type	Function	Extension
text/plain	Python Script mask	.xx

To add a MIME type in IIS 7

1. Click on the web server entry in the left-hand pane.
2. Click on the MIME typesimage:media/Selection_013.png[image] icon in the centre pane.
3. Click on the "Add. . . " link on the right-hand pane.
4. Add all entries required by the table above.



At this point IIS should be configured to proxy TACTIC service requests to the TACTIC service, while leaving all other requests to IIS.

http://learn.iis.net/page.aspx/485/define-and-configure-an-application-request-routing-server-farm/

http://learn.iis.net/page.aspx/486/http-load-balancing-using-application-request-routing/

http://blogs.msdn.com/nickhodge/default.aspx?p=2

5 Install Database

5.1 TACTIC Database Configuration

By default TACTIC will look at localhost to the database. This is the simplest implementation and most installations will have the database on the same physical server as the TACTIC application servers. However, in scaling up TACTIC, it may become necessary to separate the database server from the TACTIC applications servers.

TACTIC can be configured to look at an external database by connecting through sockets of TCP/IP. This can be set in the TACTIC configuration file.

While it is possible to edit this file manually, it is often simpler to use the provided utility found in: <TACTIC_INSTALL_DIR>/src/bin/ut

This command line script will ask a series of questions and update the appropriate configuration file

Usage:

python change_db_info.py

When executing, the script will ask configuration questions and show a default value in brackets. If the default is sufficient, merely press enter without entering anything in will accept the default.

Below is a sample output where all the default values are accepted:

TACTIC Database Configuration

Please enter the database vendor (PostgreSQL or Oracle):

(PostgreSQL) →

Please enter database server hostname or IP address:

(localhost) →

Please enter user name accessing the database:

(postgres) →

Please enter database password:

(ENTER to keep password, EMPTY for empty password)

Enter Password →

Vendor: [PostgreSQL]

Server: [localhost]

User: [postgres]

Save to config (N/y) →

Once the values have been saved, a restart of TACTIC is required to start using the new database configuration.

6 Server Configuration

6.1 TACTIC Configuration File

The TACTIC config file stores settings such as directory locations and email server information.

The name of the TACTIC config file and the location can be found here:

Filename	Operating System	Location
tactic-conf.xml	Windows or Linux	/config

note: Prior to TACTIC 3.8, the TACTIC config file is named differently for each operating system. See the table below:

Filename	Operating System	Location
tactic_win32-conf.xml	Windows	/project/config
tactic_linux-conf.xml	Linux	/project/config

For Windows, it is C:\ProgramData\Southpaw\Tactic\data by default for Enterprise Edition.

Otherwise, it is usually /home/tactic/tactic_data by default on a Linux machine. Southpaw supplies the TACTIC config file as a template on installation. Once installed, this file can be modified to reflect any of the options described in the sections below. Most of the parameters can be modified in the UI through Global > System Config as well. If an option tag does not exist in a particular section in your config file, TACTIC will assume a default or it can simply be added in.

Install

This section defines the hostname for the server (if different from "localhost") as well as the temp directory to be used for TACTIC. The tmp_dir is where temporary files are stored as well as the TACTIC log files.

```
<install>
    <hostname>localhost</hostname>
    <tmp_dir>/home/tactic/tactic_temp</tmp_dir>
    <default_project>default_project_code</default_project>
    <include_js>/context/some_external_lib.js</include_js>
</install>
```

HOSTNAME	This section defines the hostname for the server (if different from "localhost"). The hostname is what TACTIC listens to.
TMP_DIR	The temp directory to be used by TACTIC.
DEFAULT_PROJECT	Default project when the user browse the TACTIC base url.
INCLUDE_JS	You can include one or more external js files you want to make use of separated by ,.

Services

This section defines information regarding the services external to TACTIC.

```
<services>
    <mailserver>smtp8.sympatico.ca</mailserver>
    <mail_user>some_username</mail_user>
    <mail_password>some_password</mail_password>
    <mail_port>a port number other than 25</mail_port>
    <mail_sender_disabled>true</mail_sender_disabled>
    <mail_tls_enabled>true</mail_tls_enabled>
    <python>python</python>
    <python_path>/home/apache/custom</python_path>
    <render_submit_class>sites.racoon.modules.command.CustomRenderSubmit</ ↩
        render_submit_class>
    <process_count>3</process_count>
    <thread_count>50</thread_count>
    <process_time_alive>30</process_time_alive>
    <system_class></system_class>
</services>
```

MAILSERVER	The URL of the SMTP mail server
MAIL_PASSWORD	The password for accessing the SMTP mail server that requires authentication
MAIL_USER	The user name for accessing the SMTP mail server that requires authentication
MAIL_PORT	The port for the SMTP mail server (if different that 25)
MAIL_SENDER_DISABLED	disable using the sender name in sending of email in case the email server does not allow sender's email not owned by the sender
MAIL_TLS_ENABLED	enable TLS (Transport Layer Security) for the connection to email server
PYTHON	The root path of the Python installation. "python" is usually sufficient.
PYTHON_PATH	The server-side location for client files. This location can also be mounted from a shared volume if you wish to maintain stricter server access for clients. For multiple paths, separate with | .e.g. /home/apache/customl/home/apache/custom_two
RENDER_SUBMIT_CLASS	The class used for render submissions.
PROCESS_COUNT	The number of processes the TACTIC service would spawn. It needs to match the number of ports used in the load balancing scheme in the Apache configuration.
THREAD_COUNT	The number of worker threads generated for each instance of the TACTIC process. If not set, it defaults to 10 which is too low to handle rapid requests.. TACTIC's default is 50 on new installation. A good balance of process_count and thread_count can improve response time of the server.
PROCESS_TIME_ALIVE	The number of minutes a TACTIC process gets respawned. It helps with the memory consumption inherent with a long-running Python process.
SYSTEM_CLASS	Allows for an override some of the low level system functionality. For example *mkdirs* and *exists*

Security

This section defines information regarding the services external to TACTIC.

```
<security>
    <version>2</version>
    <ticket_expiry>10 hour</ticket_expiry>
    <authenticate_mode>default</authenticate_mode>
    <authenticate_class></authenticate_class>
    <authenticate_version>2</authenticate_version>
    <case_insensitive_login>>false</case_insensitive_login>
    <max_login_attempt>3</max_login_attempt>
    <account_lockout_duration>30</account_lockout_duration>
    <auto_create_user>false</auto_create_user>
    <api_require_password>true</api_require_password>
    <api_password></api_password>
    <allow_guest>false</allow_guest>
    <guest_mode>restricted</guest_mode>
    <guest_url_allow>/guest_view</guest_url_allow>
</security>
```

TICKET_EXPIRY	The number of hours a login ticket expires after
AUTHENTICATE_MODE	default: This is basically just looks at the tactic database for information. autocreate: This autocreates the first time and then leaves the information alone. cache: This caches the information to the tactic database on every login

AUTHENTICATE_CLASS	Full class path to override the default class "pyasm.security.TacticAuthenticate". Note: Your custom class needs to override the method verify() which takes two arguments: login and password.
AUTHENTICATE_VERSION	This is the old way of authentication. 2 is the new way.
CASE_INSENSITIVE_LOGIN	If set to true, it allows case insensitive login name. When autocreate mode is used, all login entries created will have a lowercase login name. It can be used in combination with Active Directory setup.
MAX_LOGIN_ATTEMPT	Number of times login attempt can fail before account is locked out.
ACCOUNT_LOCKOUT_DURATION	Duration of time the user account is locked out for failed login attempt if specified.
AUTO_CREATE_USER	Autocreate user in TACTIC during authentication phase if it does not exist. (Deprecated: use "authenticate_mode" in new way of authentication)
API_REQUIRES_PASSWORD	This setting requires password to login or not
API_PASSWORD	A generic Client API password can be set here
ALLOW_GUEST	True or false can be set to allow guest to access without login
GUEST_MODE	Full or restricted can be set. In restricted mode, a /guest relative URL is expected to be defined in Custom URL to restrict the guest to only see a particular view
GUEST_URL_ALLOW	In restricted mode, one can have multiple relative URLs predefined for guest, separated by |.

Database

```
<database>
    <vendor>PostgreSQL</vendor>
    <server>localhost</server>
    <port></port>
    <user>postgres</user>
    <password>none</password>
    <sobject_database>sthpw</sobject_database>
    <pool_max_connections>0</pool_max_connections>
</database>
```

VENDOR	The database vendor (software) the database will be installed on.
SERVER	The hostname of the server. This is localhost if TACTIC and the database are on the same server
PORT	The database connection port
USER	The user name for the database connection
PASSWORD	The password for the database connection.
SOBJECT_DATABASE	The database where SObject definitions will be stores
POOL_MAX_CONNECTIONS	The pool of connections available for connecting to the database. 0 is recommended for PostgreSQL implementation

Perforce

```
<perforce>
    <web_dir>perforce</web_dir>
    <p4>p4</p4>
    <port>1666</port>
</perforce>
```

WEB_DIR	The webdir for the perforce connection.
P4	
PORT	The port to be used for connection to perforce.

Look

This setting provides a method of setting the TACTIC skin in the server for all users. In this example, the *BON_NOCHE* palette specified:

```
<look>
    <palette>BON_NOCHE</palette>
</look>
```

Other available palettes are *AQUA*, *DARK*, *BRIGHT*, *DEFAULT*, *SILVER*, *AVIATOR*, and *ORIGAMI*. Alternatively, the whole palette can be customized as follows:

```
<look>
     <palette>{
             'color':        '#000000',          # main font color
             'color2':       '#FFFFFF',          # secondary font color
             'color3':       '#FFFFFF',          # tertiary font color
             'background':   '#FDEEA7',          # main background color
             'background2':  '#1A9481',          # secondary background color
             'background3':  '#003D5c',          # tertiary background color
             'border':       '#666666'           # main border color
    }</palette>
</palette>
</look>
```

The side bar color may not change right away until the next TACTIC service restart.

PALETTE	The default palette setting for all TACTIC users.

Checkin

TACTIC uses the following directory and path settings for internal and client interaction. They are included in the tag (for checkins).

VERSIONLESS_MODE	COPY or SYMLINK: Turn on versionless mode for checkins for all projects. To set the versionless mode per project, go to PROJECT ADMIN → PROJECT SETTINGS and add a the key VERSIONLESS_MODE and the value: COPY or SYMLINK.
ASSET_BASE_DIR	The directory where the assets are stored in the TACTIC server.
WEB_BASE_DIR	The root URL that maps the *asset_base_dir* directory
WIN32_LOCAL_BASE_DIR	The base directory in Windows client machines
LINUX_LOCAL_BASE_DIR	The base directory in Linux client machines
WIN32_SANDBOX_DIR	The default sandbox directory in the Windows client machines (it can be overridden by Remote Repo)
LINUX_SANDBOX_DIR	The default sandbox directory in the Linux client machines (it can be overridden by Remote Repo)
WIN32_CLIENT_REPO_DIR	Maps the asset_base_dir directory as seen by the Windows client. For example, if asset_base_dir is on a Linux server with a path like "/home/apache/assets" but from the Windows client, it is mapped as "Z:/assets", then "Z:/assets" should be the value for this setting. By default, this path is empty because the system assumes the client and server are on the same Windows machine.
LINUX_CLIENT_REPO_DIR	Same as *win32_client_repo_dir* except it is from the perspective of a Linux client machine
WIN32_CLIENT_HANDOFF_DIR	Windows client-side handoff directory for Client API transactions. (Find out more about the handoff directory below.)
WIN32_SERVER_HANDOFF_DIR	Windows server-side handoff directory for Client API transactions
LINUX_CLIENT_HANDOFF_DIR	Linux client-side handoff directory for Client API transactions
LINUX_SERVER_HANDOFF_DIR	Linux server-side handoff directory for Client API transactions
SUDO_NOPASSWORD	It controls whether sudo can be run to change the user id and group id of the files checked in. It is particularly important if you want to ensure files checked in to the TACTIC repository are owned by TACTIC and not overwritable by just any users. If set to true, "no password" should be enabled for the user TACTIC is run as in the OS. e.g. For Fedora, assuming you have sudo installed: In the file /etc/sudoers, the following line should be uncommented: %wheel ALL=(ALL) NOPASSWD: ALL In the file /etc/group, apache should be added to the group wheel wheel:x:10:root,apache
VERSION_PADDING	padding of 3 or more can be set for checked-in files

THE HANDOFF DIRECTORIES

Handoff directories can be seen by both the server and the client machines. They are used for 3D checkins and client API interactions, and are important for specifying how the client and server sides see the same location.

For example, if you have the location //192.168.0.105/handoff available on your network and it is mounted as /home/apache/handoff on a server, then it would be important to include the following entries:

```
<win32_client_handoff_dir>//192.168.0.105/handoff</win32_client_handoff_dir>
<win32_server_handoff_dir></win32_server_handoff_dir>
<linux_client_handoff_dir></linux_client_handoff_dir>
<linux_server_handoff_dir>/home/apache/handoff</linux_server_handoff_dir>
```

DIRECTORY CONFIGURATION EXAMPLES

Example 1

The assets directory is located on the TACTIC server and allows for read-only access from client machines in the local subnet.

- The assets directory is located on the TACTIC server and allows for read-only access from client machines in the local subnet.
- The handoff directory is located on the TACTIC server and allows for read/write access from client machines in the local subnet
- The Windows and Linux *client_repo_dir* looks directly to the server for the available "assets" share
- The Windows and Linux *client_handoff_dir* looks directly to the server for the available "handoff" share

```
<checkin>
    <asset_base_dir>/home/apache/assets</asset_base_dir>
    <web_base_dir>/assets</web_base_dir>
    <win32_local_base_dir>C:/sthpw</win32_local_base_dir>
    <linux_local_base_dir>/tmp/sthpw</linux_local_base_dir>
    <win32_sandbox_dir>C:/sthpw/sandbox</win32_sandbox_dir>
    <linux_sandbox_dir>/tmp/sthpw/sandbox</linux_sandbox_dir>
    <win32_client_repo_dir>//192.168.0.105/apache/assets</win32_client_repo_dir>
    <linux_client_repo_dir>/usr/assets</linux_client_repo_dir>
    <win32_client_handoff_dir>//192.168.0.105/apache/handoff<win32_client_handoff_dir>
    <win32_server_handoff_dir></win32_server_handoff_dir>
    <linux_client_handoff_dir>/home/apache/handoff</linux_client_handoff_dir>
    <linux_server_handoff_dir>/home/apache/handoff</linux_server_handoff_dir>
    <version_padding>3</version_padding>
</checkin>
```

Example 2

- The assets directory is located on another server and mounted locally on the TACTIC server to /mnt1/assets.
- The Windows and Linux *client_repo_dir* is mapped/mounted to the TACTIC *asset_base_dir*
- The Windows and Linux *client_handoff_dir* is mapped/mounted to the TACTIC *server_handoff_dir*

```
<checkin>
    <asset_base_dir>/mnt1/assets</asset_base_dir>
    <web_base_dir>/assets</web_base_dir>
    <win32_local_base_dir>C:/sthpw</win32_local_base_dir>
    <linux_local_base_dir>/tmp/sthpw</linux_local_base_dir>
    <win32_sandbox_dir>C:/sthpw/sandbox</win32_sandbox_dir>
    <linux_sandbox_dir>/tmp/sthpw/sandbox</linux_sandbox_dir>
    <win32_client_repo_dir>z:/assets</win32_client_repo_dir>
    <linux_client_repo_dir>/assets</linux_client_repo_dir>
    <win32_client_handoff_dir>z:/tactic_handoff<win32_client_handoff_dir>
    <win32_server_handoff_dir></win32_server_handoff_dir>
```

```
    <linux_client_handoff_dir>/tactic_handoff</linux_client_handoff_dir>
    <linux_server_handoff_dir>/home/apache/tactic_handoff</linux_server_handoff_dir>
    <version_padding>3</version_padding>
</checkin>
```

6.2 Configure the TACTIC Service

For development, test, or evaluation purposes, there is no need to run TACTIC as a service. The `startup_dev.py` script should suit these purposes adequately. However, for production, it is highly recommended to run TACTIC as a service.

In the `<TACTIC_INSTALL_DIR>/src/install` directory, there is a directory called `service`, with two files:

[multiblock cell omitted]
[multiblock cell omitted]

6.3 TACTIC Configuring SSL

This document discusses security layers that can be implemented on top of the TACTIC service.

Services covered

- Apache HTTP server
- IIS HTTP server
- PPTP server
- IPsec

Southpaw Technology does not provide any support, either direct or implied, for remote access.

Any application that is put on a public network can potentially be compromised, even with rigorous security testing. TACTIC, in its current version, has not been production tested for security scenarios outside of completely trusted networks like intranets. This document does not make any claims that TACTIC has any level of security beyond what its current architecture design allows.

TACTIC is a bandwidth-intensive application, and client-server performance will be affected by any number of network issues between the client and the server.

There are numerous elements of functionality that go beyond utilizing HTTP port access in enterprise applications of TACTIC. Many 3rd party applications that communicate with TACTIC will produce unexpected behavior when utilized in a fashion that was not intended for the product, such as use in remote applications.

TACTIC on public networks

HTTP transport co-services can be configured to provide authentication services to TACTIC.

HTTP zone access

The simplest form of HTTP security authentication zone access. To enable different security scenarios, the HTTP service must be configured for use of these scenarios. Configuration files (usually called .htaccess files) contain a number of settings that can be used for integrating the application with the capabilities of the Web server.

When enabling password zone access, the password is transmitted to the server in cleartext, unless TLS is enabled.

Apache

Apache uses either directives for directory access in the httpd.conf file, or can be enabled on a per-directory basis, with the use of a .htaccess file. Either way can be used. Apache provides modules for LDAP, MySQL, flat-file, ADS, and many other authentication mechanisms.

IIS

IIS can be configured at the top level through the IIS snap-in, or by individual configuration file. IIS 7 uses a file called Web.config to hold settings for integration with applications. The Web.config file contains information that control module loading, security configuration, session state configuration, and application language and compilation settings.

Overview

TACTIC can be configured to use the SSL transport layer. This layer is independent of the TACTIC service, and can be tailored to the needs of the deployment without major changes to the TACTIC service.

Both major HTTP servers can be configured for SSL support. There are many materials online available for configuration of the various flavors of these two major HTTP service projects.

Since there are many different versions of these servers, a simple search of Windows 2003 IIS SSL as an example will yield many HOWTOs and configuration references. Please consult the documentation for these services.

The configuration examples that have been set out below are by no means complete. Depending on the HTTP service used, and the platform used, these examples may not be enough set directives or steps to properly complete the SSL process. They are given as guides.

Steps

The major steps to utilizing SSL in with TACTIC are

- Enabling TACTIC to converse with SSL
- Enabling the SSL service
- Creating a Virtual Server, or extending the existing server to accept SSL
- Providing CA certificates

Configuration

TACTIC configuration

The SSL HTTP layer is kept separate from TACTIC via a proxy, so the only configuration change required of TACTIC is in the tactic_(OS).conf file. The <security><protocol> directive must tell TACTIC to expect an SSL delivery.

```
<security>
    <protocol>https</protocol>
</security>
```

The setting can be set to either “http&#xfffd;? or “https&#xfffd;?.

Apache

The apache project uses mod_ssl as a modular way of inserting SSL capabilities into the HTTP service.

The example OS is Fedora 11. “Yum&#xfffd;? is used to add SSL to apache.

```
[root@lindsay conf.d]# yum install mod_ssl
Loaded plugins: refresh-packagekit
...
Complete!
[root@lindsay conf.d]#
```

Once mod_ssl is added, the Apache configuration should contain at least these directives. The example directives are contained in conf.d/ssl.conf in the Fedora 11 example. File locations will vary according to OS.

```
LoadModule ssl_module modules/mod_ssl.so
Listen 443

SSLPassPhraseDialog  builtin
SSLSessionCache         shmcb:/var/cache/mod_ssl/scache(512000)
SSLSessionCacheTimeout  300
SSLMutex default
```

```
SSLRandomSeed startup file:/dev/urandom  256
SSLRandomSeed connect builtin
SSLCryptoDevice builtin
<VirtualHost _default_:443>
SSLEngine on
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

As soon as its added and the server is restarted, the SSL service becomes available on the network interface that apache is running on.

IIS

IIS can be configured at the top level through the IIS snap-in, or by individual configuration file.

To configure SSL on a Web server or a Web site

\1. In IIS Manager, double-click the local computer, and then double-click the Web Sites folder. 2. Right-click the Web site or file that you want to protect with SSL, and then click Properties. 3. Under Web site identification click Advanced. 4. In the Advanced Web site identification box, under Multiple identities for this Web site, verify that the Web site IP address is assigned to port 443, the default port for secure communications, and then click OK. Optionally, to configure more SSL ports for this Web site, click Add under Multiple identities of this Web site, and then click OK. 5. On the Directory Security or File Security tab, under Secure communications, click Edit. 6. In the Secure Communications box, select the Require secure channel (SSL) check box. 7. To enable SSL client certificate authentication and mapping features, select the Enable client certificate mapping check box, click Edit, add the 1-to-1 or many-to-1 mappings you need, and then click OK three times.

Secure transaction processing

Processing transactions securely on the web means that there is a need to be able to transmit information between the web site and the customer in a manner that makes it difficult for other people to intercept and read. SSL, or Secure Sockets Layer, takes care of this. It works through a combination of programs and encryption/decryption routines that exist on the web services host, and in browser programs (like Firefox and Internet Explorer)

Performance

TLS has encryption/decryption routines as part of its security. These routines can be bandwidth/CPU intensive. Any usage of TLS can compromise TACTIC, if the routines are incorporated into the same host as the TACTIC service. See guides on load-balancing for details on offsetting this.

Overview

A VPN will provide the most trouble-free access to TACTIC remotely. In this scenario, the authentication/encryption routines are completely removed from the realm of TACTIC configuration. This not only helps to isolate TACTIC from complex configuration issues, but also allows for isolated troubleshooting of remote access issues.

PPTP

PPTP is Microsoft supplied product. If an enterprise deployment of TACTIC includes ADS authentication, then PPTP can be used as the VPN transport layer. Usually, deployment is quite easily done. In PPTP, usernames and passwords are used to complete the VPN link. PPTP can be considered the “road-warrior&#xfffd;? VPN, meaning that it is easily deployed to users.

IPsec

Ipsec is a suite of protocols used to secure data between hosts. A VPN can be transported on top of this protocol. Typically, two remote hosts are configured to communicate with each other, such as routers. This type of VPN is typically used to connect two offices together.

Hardware VPNs

Hardware VPNs such as offerings provided by companies like Cisco, can be easily implemented. These Systems are designed for minimal ramp-up, and can be implemented quickly.

Security Terms

“is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all messages going between the two victims and inject new ones, which is straightforward in many circumstances (for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle).&#xfffd;?

SSL (Wikipedia)

“Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide security for communications over networks such as the Internet. TLS and SSL encrypt the segments of network connections at the Transport Layer end-to-end.&#xfffd;?

Cleartext (Wikipedia)

Cleartext is transmitted, unencrypted text.

“In a cryptosystem, weaknesses can be introduced through insecure handling of plaintext, allowing an attacker to bypass the cryptography altogether. Plaintext is vulnerable in use and in storage, whether in electronic or paper format. “

PPTP (wikipedia)

The Point-to-Point Tunneling Protocol (PPTP) is a method for implementing virtual private networks. PPTP uses a control channel over TCP and a GRE tunnel operating to encapsulate PPP packets.

The PPTP specification does not describe encryption or authentication features and relies on the PPP protocol being tunneled to implement security functionality. However the most common PPTP implementation, shipping with the Microsoft Windows product families, implements various levels of authentication and encryption natively as standard features of the Windows PPTP stack. The intended use of this protocol is to provide similar levels of security and remote access as typical VPN products.

IPSEC (Wikipedia)

Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a data stream. IPsec also includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used to protect data flows between a pair of hosts (e.g. computer users or servers), between a pair of security gateways (e.g. routers or firewalls), or between a security gateway and a host.[1]

6.4 Active Directory Integration

TACTIC provides the ability to easily connect to any active directory installation for both authentication and for synchronization of user data. With a set of directives in the TACTIC configuration file, it is possible to connect to Active Directory for authentication and user information.

This synchronization takes place at login time. At this point, TACTIC takes the desired information for a particular user and caches it into the "sthpw/login" search type. Subsequent requests would normally use an issued ticket given at login time. On these requests, no further querying of active directory is needed until the ticket expires or the user signs out the application.

The active directory modules make use of win32 libraries for python. These must be installed in order for the connection to active directory to function properly

There are a number of directives in the TACTIC config file that can be used to configure the active directory settings. These allow you to adjust TACTIC behavior to suit the needs of the facility.

In order to turn on active directory authentication, you must change the authenticate class to the following:

authenticate_class: tactic.active_directory.ADAuthenticate

The following directives can be set under the active directory category:

domains: This is a "|" delimited list of the domains that exist in the network. If specified, a selection box for domains will be added to the login page.

*allow:*can be "all", which allows everyone to log in if authentication is approved or it can point to the name of a specific Active Directory attribute that must be set to True. If a person is denied access, the will receive the error: "Permission denied due to insufficient Active Directory clearance".

*default_groups:*defines the default groups that a user will belong to if none is specified. Multiple groups are delimited by "|".

*default_license_type:*determines the default license type for a user if none is specified in the Active Directory attribute "tacticLicenceType".

Below is an example of a typical entry in the TACTIC config file:

<active_directory>

<domains>xxx|yyy|zzz</domains>

<allow>tacticEnabled</allow>

<default_groups>client</default_groups>

<default_license_type>user</default_license_type>

</active_directory>

Allow anyone to login:

<active_directory>

<allow>all<allow>

</active_directory>

Allow anyone to login and will be put in the "client" group if user has no groups specified.

<active_directory>

<allow>all</allow>

</active_directory>

Only allow thos with the attibute tacticEnabled in Active Directory set to "true"

<active_directory>

<allow>tacticEnabled</allow>

</active_directory>

Enable users to select a domain (xxx, yyy or zzz) in the login screen

<active_directory>

<allow>all</allow>

<domains>xxx|yyy|zzz</domains>

</active_directory>

Active Directory attributes use camel case notation (aaaBbbCcc), while TACTIC users lowercase with underscore separators for columns(aaa_bbb_ccc). In order to maintain consistency within the TACTIC application, a mapping of columns from active directory to TACTIC is provided. The following mappings are made by default:

mail	email
telephoneNumber	phone_number
department	department
displayName	first_name, last_name (broken up)
tacticLicenceType	license_type

The Active Directory variable "tacticLicenseType" is a custom variable that indicates which type of license a particular user can occupy in TACTIC. If this attribute is missing from a users active directory profile, then they will be denied a login. This attribute can be used to determine if a particular user in active directory is allowed to login to TACTIC.

The only supported license for this attribute are "user" and "default". Other license types have not yet been implemented yet.

On log in, TACTIC will look at all of the groups that a user belongs to in Active Directory and match those group names to the "ad_login_group" column in the "sthpw/login_group" search type. This grouping list will synchronized at this time, removing

the users from groups not specified in Active Directory and add those that are specified. This means that Active Directory is in full control of the groups that a user is part of and therefore must be managed entirely in Active Directory.

For the name of the group, TACTIC only looks at the root of the path to map the group name. For example, an active directory group with the following distinguished name:

memberOf: CN=supervisor,OU=Users,OU=EIS,DC=domain,DC=us,DC=xxxx,DC=com

TACTIC will need only "supervisor" to be entered in the "ad_login_group" column.

If on logging in, the number of users exceeds the number of users in the license, then that user will be denied access and an entry in the "sthpw/login" search type will not be made. However, all other users currently registered can continue to work normally.

6.5 Configuring TACTIC and Co-Services

TACTIC Service Configuration

Overview

Multiple TACTIC servers can be leveraged in environments where there is heavy report analysis, and where custom TACTIC environments are making heavy use of API and GUI calls. To spread out the load of the requests made, custom API scripts can be run on one TACTIC server, while another TACTIC server can be used to serve GUI requests. The only consideration then with multiple TACTIC machines will be where the database and the asset directories.

Database

TACTIC needs to know where to find assets and asset metadata. To do this, each installation of TACTIC must be able to have direct file system access to assets storage, and network access to the database co-service.

tactic_(os).conf

needs to contain this information.

```
<database>
<server>[DB server IP]</server>
</database>
```

All other settings, covered previously, can be set according to individual requirements of the host machine and environment.

Assets Storage

TACTIC needs to know where to find assets. To do this, each installation of TACTIC must be able to have direct file system access to assets storage. The details of file system management are beyond the scope of this document, but typically are within the realm of the system administrator.

Processes

If the TACTIC server has no other services attached, there is probably room to increase the number of processes running on each machine.

```
<services>
<process_count>6</process_count>
</services>
```

Refer to Reference: TACTIC Default Configuration for the complete sample configuration file.

Refer to Reference: TACTIC Service Configuration Directives for configuration directives.

HTTP Co-service Configuration

Apache

Apache is used for the following sample HTTP configuration for TACTIC. Some configuration knowledge of apache is required.

Permissions – Allowing TACTIC to store and manipulate assets

This section defines the location and availability of the assets directory, which is the primary source of data for the apache server. There is also the declaration of an alias to the TACTIC source directory, which contains various objects that TACTIC uses, such as widget elements.

```
Alias /context         /home/apache/tactic/src/context
Alias /assets       /home/apache/assets
Alias /doc/           /home/apache/tactic/doc/
```

The section with <Directory> directives defines the access rules for the assets directory and the "/tactic" directory, which is just a conveniently named alias.

```
<Directory "/home/apache/tactic" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
<Directory "/home/apache/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
```

Proxying/Rewriting a single process – Redirecting requests to TACTIC

Apache needs to know where to find the proxied TACTIC service in the httpd.conf file. The below configuration takes advantage of only one process being served from port 8081 on the local machine.

The second section, with RewriteRule directives, defines the re-write rules for the TACTIC service, to segregate requests handled by the TACTIC server, from static asset delivery through the HTTP service. These rules channel any requests that are prefixed with the "/tactic" path to the TACTIC server on port 8081.

```
RewriteRule   ^/tactic/(.+)$ http://localhost:8081/tactic/$1 [P,L]
RewriteRule   ^/tactic http://localhost:8081/tactic/ [P,L]
RewriteRule   ^/projects/(.+)$ http://localhost:8081/tactic/$1 [P,L]
RewriteRule   ^/projects http://localhost:8081/tactic/ [P,L]
```

Proxying/Rewriting multiple processes – Redirecting requests to TACTIC

The proxy configuration can be enhanced with a load balancing scheme for one or more machines. Apache has the ability to randomly select from a list of locations via a rewrite map.

```
RewriteMap    lb    rnd:/home/apache/sites/load_balance.txt
```

This map can feed the rewrite rules with a dynamically assigned host name.

```
RewriteRule   ^/tactic/(.+)$ http://${lb:dynamic}/tactic/$1    [P,L]
RewriteRule   ^/projects/(.+)$ http://${lb:dynamic}/tactic/$1  [P,L]
RewriteRule   ^/tactic http://${lb:dynamic}/tactic/            [P,L]
RewriteRule   ^/projects http://${lb:dynamic}/tactic/          [P,L]
```

The file load_balance.txt is an arbitrarily named and located file that contains the names of servers that will be referred to by the "rnd" function in the Rewrite rules. The "lb:dynamic" reference will be replaced by the name of the server file

The load_balance.txt contains a pipe separated list of hosts named "dynamic"

```
dynamic localhost:8081|localhost:8082|localhost:8083
```

This list is dependent on the number of ports that TACTIC is running on, specified by tactic_(OS).conf. Note that apache can proxy from IP addresses or hostnames external to the host that the service is parked on.

On this single line list, add all machines and ports that are running TACTIC. Since the scheme algorithm is random, it does not matter what order the machines/ports are listed in, just that they are actually on the list. In this case, there are multiple TACTIC machines, named "tacticserver01" and "tacticserver02". The example assumes the machine has either DNS entries for these machines or entries in the "hosts" file. IP addresses can also be used.

```
dynamic tacticserver01:8081|tacticserver02:8081|tacticserver01:8082
# and so on, until all machines/ports are covered
```

Refer to Reference: Apache Configuration for TACTIC for the complete sample configuration file.

Database Co-service Configuration

PostgreSQL

PostgreSQL is used as the database co-service in the following sample configuration. PostgreSQL has only two configuration files that are required to be examined in order to function with TACTIC; pg_hba.conf and postgresql.conf

Network trust - pg_hba.conf

The pg_hba.conf configuration file contains a list of users and hosts with clearance levels. In the default pg_hba.conf file that comes with TACTIC, the network trust level is set for the most open access by the localhost;

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local      all  all  trust
# IPv4 local connections:
host    all         all         127.0.0.1/32          trust
# IPv6 local connections:
host    all         all         ::1/128               trust
```

In this example, all local connections to PostgreSQL are trusted. This configuration matches the correct configuration required by a single machine.

```
host   all     all  <TACTIC_HOST_IP_ADDR>     trust
```

Network interface - postgresql.conf.

The postgresql.conf configuration file has a setting that will allow the PostgreSQL service to bind to the particular interface required.

Of concern is the "listen_addresses" attribute.

```
listen_addresses = 'localhost'
```

By default, the PostgreSQL service is only bound to the localhost. This is fine for single machine operation of the TACTIC service.

```
listen_addresses = '*'
```

This setting or specific IP addresses can be used if the DB service is not on the same machine as the TACTIC service

6.6 Setup Email

This document describes how to setup the mail server settings in TACTIC in order for notifications to be sent out as emails.

To setup the SMTP mail server, open the TACTIC config file.

The TACTIC config file is located here:

On Linux:

```
<TACTIC_INSTALL_DIR>/projects/config/tactic_linux-conf.xml
```

On Windows:

```
<TACTIC_INSTALL_DIR>/projects/config/tactic_win32-conf.xml
```

Go to the **<services>** section of the config file.

Add the following settings with the prefix **<mail...>**.

Below is an example of a completed mail config section in **bold**:

```
<python>python</python>
<python_path></python_path>
<render_submit_class></render_submit_class>
<render_dispatcher></render_dispatcher>
<system_class></system_class>
<pool_max_connections>3</pool_max_connections>
<process_count>3</process_count>
```

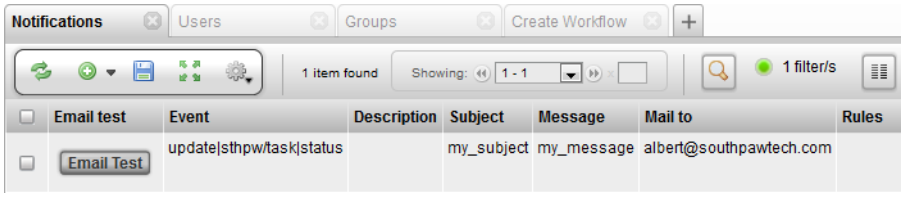
mailserver	The URL of the SMTP mail server
mail_password	The password for accessing the SMTP mail server that requires authentication
mail_user	The user name for accessing the SMTP mail server that requires authentication
mail_port	The port for the SMTP mail server (if different that 25)
mail_sender_disabled	disable using the sender name in sending of email in case the email server does not allow sender's email not owned by the sender
mail_tls_enabled	enable TLS (Transport Layer Security) for the connection to email server

Go to the Notifications view under:

Admin Views → Site Admin → Notifications

Click on the green plus button on the tool shelf to insert new a notification.

Fill in the following the minimum fields to create a test notification:



Click on the **Email Test** button to send out a test email to the recipient.

For further details on setting up advanced notifications, please refer to the doc titled: **Advanced Notification Setup**.

6.7 HTTP Co-Service Installation/Configuration

The aim of this document is to be able to configure Apache to proxy TACTIC on a UNIX machine. For Apache on Windows machines, there may be some slight differences in behavior or steps.

1. User must have sufficient technical knowledge to navigate the OS directories, edit text files, and start/stop services. Knowledge of the target web server is required.
2. TACTIC must be installed on the host machine. It should be runnable to the point where all diagnostics presume no error in the TACTIC configuration. In the below examples, TACTIC is installed with all default options.
3. The web server must be installed on the host machine. Since different OS's can have vastly different HTTP service default configurations, configuration files will need to be found by command or appropriate documentation consultation.

The TACTIC host machine should have the following prerequisites

- A working TACTIC installation
- An OS capable of running the target HTTP co-service

The HTTP configuration assumes that the main server is being used to proxy TACTIC requests, without SSL or virtual servers. TACTIC requires that the appropriate modules or 3rd party modules are activated.

This document assumes that TACTIC has been configured correctly and is functioning as expected. It is assumed that TACTIC is installed with default ports 8081 through 8083. This is the typical default TACTIC configuration.

6.8 Database Resource

TACTIC can map to any supported database platform. A single TACTIC installation can map on to any number of external databases and use them as a database resource. These database resources are treated as first class citizens in the TACTIC. All of the features that are inherent to any native TACTIC data are also available to these external database resources. To implement this functionality, db_resource can be used to connect a project to a database resource. It defines the location and connection credentials to access a database server. Database resources can be added under the side bar menu **Global** → **Database Resource**.

Specify (or look up) the Database Resource under **Admin Views** → **Global**.

The database resource entry in Database Resource contains all the information needed to connect to a database. In order for TACTIC to access another database, it has to have a database resource registered and then it has to be mapped to a project.

In the example below, the new database's code is named: **my_new_db_resource**, which is the reference of this entry. Host, which refers to the IP address of the server, is set to be a tested address. The Login and Password are the information used to log in the database.

Since all TACTIC projects have a single database resource that it uses as a persistent store for sObjects. By default, the project will use the same resource as the sthpw database and is defined in the TACTIC config file. However, this can be configured to point to another database resource.

Note that the search type uses a project to find the corresponding database. Any project configured to look at a database resource with the db_resource and database columns. The db resource defines the connection criteria and the column determines which database to connect to that resource.

To add new a database as pure resource database, Specify (or look up) the *Add Project* and *Projects* columns under **Admin Views** → **Global** → **Database Resource**.

In the example below, the new database called **test_database001** from the db_resource **my_new_db_resource** has been added to a Project called **Testing**.

For a project to become a full TACTIC project (complete with separate URL space: `http://<server>/<project>` and separate themes an interfaces), all that is required is the config tables to exist in that database resource. Thus a database resource can start off as a data source and then, at any time, be upgraded to a full TACTIC project if that is ever required by importing the appropriate config tables that the search type uses in the project to find the corresponding database. The project is configured to look at a database resource with the db_resource and database column. The db resource defines the connection criteria and the column determines which database to connect to on that resource.

In order for TACTIC to access another database, it has to have a database resource registered and then it has to be mapped to a project. This mapping is currently required because all search types are scoped and searched using the notation:

```
<namespace>/<table>?project=<project>&code=<code>
```

An example search_type entry would be:

```
jobs/media?project=my_test_project
```

6.9 RemoteAccessTactic

7 Fileserver

7.1 Relocating the *assets* directory

proper management of assets and asset metadata are essential to managing the TACTIC service. Versioned assets are by far the most disk space consuming element of asset management. TACTIC can store its asset data in any filesystem path location. All that is required is that the tactic configuration file contains the path to a directory writable by the TACTIC service user.

- **Local Access** in a simple configuration, the TACTIC service is enabled on the TACTIC host, and all assets checked in and checked out of TACTIC are stored on the local filesystem. Typically a user is created to run the TACTIC service, or an already assigned user is simply used to run the TACTIC service.
- **Network Access** in an addition to the simple configuration, the TACTIC service is enabled on the tactic host and all assets checked in and checked out tactic are stored on a network mount. Typically a user is created to run the TACTIC service and be able to write to the network mount.
- *TACTIC user permissions*It is important to note that in a network accessible filesystem, allocation of permissions are important only for the TACTIC user. Regular system users should not have access to the TACTIC assets directory.
- **Browsable access to assets** During some conversions to the management of assets by TACTIC, a frequent request is for regular user access to the assets directory managed by TACTIC. There really is not the much to see in these directories as tactics naming convention for versioning of assets creates files and directories that are largely mysterious in their arrangement.

  Is very important that the files that are managed by tactic are not touched by regular users. Manipulating files or directories managed by tactic will definitely result in data loss. Regular users should not have writable access to tactic asset directories.

Before you begin this process, you should first move or copy your assets to their new location using a network share ideally named *assets*. For a new installation, you should create an *assets* share or directory in your new location.

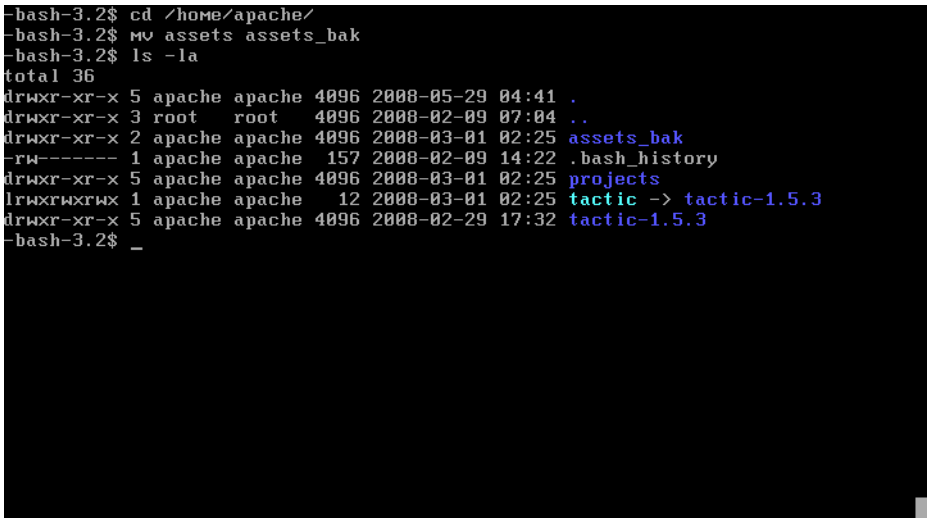
In most cases, it is ideal to relocate the `assets` directory to a dedicated file server. If you already have an assets directory, you can rename it to "assets_temp" to break the current connection. You will know the connection is broken when your asset and project thumbnails are no longer visible in the user interface. Once the assets are successfully moved to the new location you will know they are accessible when the assets and thumbnails are visible again.

In Linux, the steps to rename the current assets will be similar to the example below:

\1) Navigate to the apache directory `cd /home/apache`

\2) In the shell run `mv assets assets_bak`

\3) If you run `ls -la`, you should get a listing confirming the change



\4) To confirm the operation was successful, log in to TACTIC and you should find that none of the thumbnails or assets are available.

The config file is named differently on each operating system:

Filename	Operating System	Location
tactic_win32-conf.xml	Windows	TACTIC_INSTALL_DIR/projects/config
tactic_linux-conf.xml	Linux	TACTIC_INSTALL_DIR/projects/config

Updating the tactic_linux-conf.xml file (Linux)

In the `<checkin/>` tag in the `tactic_linux-conf.xml` config file, the `asset_base_dir` location defines where assets are stored by TACTIC. This location could be on the TACTIC server (the default is `/home/apache/assets`), or preferably on a network shared drive.

This path is defined from the perspective of the TACTIC server. An example of this entry where it looks to a share on another server would be:

```
asset_base_dir="/home/apache/assets"
```

This location on the server would be mounted to the central assets server which in this case would be:

```
//192.168.0.105/assets
```

This share would have to allow the *apache* user (which is TACTIC) to read and write to it.

There are also two variables that need to be set in the following Apache config file:

`/etc/httpd/conf.d/tactic.conf`

Change this:

```
<Directory "/home/apache/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
```

To:

```
<Directory "/mnt/d1/tactic/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All

</Directory>
```

You will also need to change the assets alias further down in the config file:

Change this:

```
Alias /assets "/home/apache/assets"
```

To:

```
Alias /assets ""
```

For the `win32_client_repo_dir` location, it should include the `asset_base_dir` path as seen by the client machine. For example, suppose the `asset_base_dir` path is "/home/apache/assets" on a Linux box. However, from a Windows client, it is mapped as:

"Z:/assets" or "//192.168.0.105/assets"

The `win32_client_repo_dir` location would then be:

win32_client_repo_dir="//192.168.0.105/assets"

The linux_client_repo_dir location is analogous to win32_client_repo_dir, except it is meant for a Linux client.

Updating the tactic_win32-conf.xml file (Windows)

In the `<checkin/>` tag in the `tactic_linux-conf.xml` config file, the `asset_base_dir` location defines where assets are stored by TACTIC. This location could be on the TACTIC server (the default is `/home/apache/assets`), or preferably on a network shared drive.

This path is defined from the perspective of the TACTIC server. An example of this entry where it looks to a share on another server would be:

```
asset_base_dir="/home/apache/assets"
```

This location on the server would be mounted to the central assets server which in this case would be:

```
//192.168.0.105/assets
```

This share would have to allow the *apache* user (which is TACTIC) to read and write to it.

There are also two variables that need to be set in the following Apache config file:

`/etc/httpd/conf.d/tactic.conf`

Change this:

```
<Directory "/home/apache/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All
</Directory>
```

To:

```
<Directory "/mnt/d1/tactic/assets" >
    Options FollowSymLinks
    AllowOverride None
    Order Allow,Deny
    Allow from All

</Directory>
```

You will also need to change the assets alias further down in the config file:

Change this:

```
Alias /assets "/home/apache/assets"
```

To:

```
Alias /assets ""
```

For the `win32_client_repo_dir` location, it should include the `asset_base_dir` path as seen by the client machine. For example, suppose the `asset_base_dir` path is "/home/apache/assets" on a Linux box. However, from a Windows client, it is mapped as:

"Z:/assets" or "//192.168.0.105/assets"

The `win32_client_repo_dir` location would then be:

win32_client_repo_dir="//192.168.0.105/assets"

The linux_client_repo_dir location is analogous to win32_client_repo_dir, except it is meant for a Linux client.

7.2 Configure the Remote Repo

The remote repository (repo) setting describes the specific environment that remote users or a remote facility may use, in the situation where it differs from the environment used by the master repository.

To set up a remote repo, you can insert an entry that represents users or a group of users that fall under a certain IP address or IP mask combo.

> **Note**
>
> If you want everyone at an IP address to use the same remote repo, a typical IP mask value is `255.255.255.255`

The login name is only required when you are not using an IP address or IP mask combo to specify users. The sandbox path and local repo path can be a network path like:

```
\\<server_name>\<share_name>\..\repo
```

or (for Windows)

```
Z:/repo
```

8 Scalability

8.1 TACTIC Scalable Configuration Examples

TACTIC and its co-services can be distributed over multiple machines. This can be specific combinations of TACTIC and co-services.

For purposes of instruction and brevity, the configuration examples are not targeted towards particular multiple machine solutions; rather, the configuration examples can be altered to suit the individual environment.

Here are some examples:

- HTTP server, DB, and TACTIC server
- HTTP server, HTTP server, DB server, and TACTIC server
- HTTP server, TACTIC server 1, TACTIC server 2, and DB server,

and so on.

Example Configuration: One Server

Single machine for small scale environment, limited usage

- all services on one machine

Example Configuration: Two Servers

Two machines to spread workload.

- HTTP and TACTIC server
- DB server

Example Configuration: Three Servers

For larger environments where automated usage and heavy reporting usage by users through the GUI.

- HTTP & DB server
- TACTIC server 1
- TACTIC server 2

8.2 Load-Balancing TACTIC

TACTIC runs on a multi-threaded application server ("Cherrypy"), but this application server uses Python threads and not system threads. This means that all of the threads in a single python process run on the same processor.

To overcome this limitation, you should run a number of full TACTIC processes across a number of different ports and allow the webserver to load-balance requests across these ports. By default in the the tactic installation config file, ` /home/apache/projects/config/ta
conf.xml`, the process count is set to 3. Add this entry if it does not exist.

```
<services>
    ...
    ...
    <process_count>3</process_count>
</services>
```

There are many ways to load-balance TACTIC. Two different ways will be described in the following:

> **Note**
>
> Warning: For load balancing, only use either:
>
> \1) the Proxy Balancer method (recommended)
>
> **or**
>
> \2) the RewriteRule method (not recommended).
>
> Do **not** use both methods at the same time.

\1. The first way (recommended way) is to make use of Apache's `proxy_balancer_module` . Ensure it is already enabled in the main Apache config file in /etc/httpd/conf/httpd.conf. In the `tactic.conf` file (provided with the TACTIC installation usually placed near the Apache 2 installation config area like /etc/httpd/conf.d/.. or /etc/apache2/..), make sure these lines below are present. This set-up corresponds to a process_count of 3 in the `tactic.conf` file. If there are more, you just have to add more*BalancerMember* line accordingly.

```
ProxyPreserveHost on

# Using the ProxyPass directives
<Proxy balancer://tactic>
  BalancerMember http://localhost:8081/tactic
  BalancerMember http://localhost:8082/tactic
  BalancerMember http://localhost:8083/tactic
</Proxy>
ProxyPass /tactic balancer://tactic
ProxyPass /projects balancer://tactic
```

This method is more desirable as it doesn't need the load_balance.txt required for the second method. In addition, it is mandatory if the process_time_alive directive is used under services in the TACTIC config file in /home/apache/projects/config/tactic_linux-conf.xml.

\2. The second way (old way) is to make use of Apache's `rewrite_module`. In the `tactic.conf` file (provided with the TACTIC installation usually placed near the Apache 2 installation config area like /etc/httpd/conf.d/.. or /etc/apache2/..), comment out the lines under "for cherrypy":

```
# for cherrypy
#RewriteRule ^/tactic/(.+)$ http://localhost:8081/tactic/$1 [P,L]
#RewriteRule ^/tactic http://localhost:8081/tactic/ [P,L]
#RewriteRule ^/projects/(.+)$ http://localhost:8081/tactic/$1 [P,L]
#RewriteRule ^/projects http://localhost:8081/tactic/ [P,L]
```

Then in the same file uncomment the lines for the random load balancing scheme:

```
# This is for using a random load_balancing scheme
RewriteMap lb rnd:/home/apache/sites/load_balance.txt
RewriteRule ^/tactic/(.+)$ http://${lb:dynamic}/tactic/$1 [P,L]
RewriteRule ^/projects/(.+)$ http://${lb:dynamic}/tactic/$1 [P,L]
RewriteRule ^/tactic http://${lb:dynamic}/tactic/ [P,L]
RewriteRule ^/projects http://${lb:dynamic}/tactic/ [P,L]
```

These lines will look at the file "`/home/apache/sites/load_balance.txt`" to find out how to replace the variable ${lb:dynamic} with a randomly chosen value from a list.

Note: The path for load_balance.txt cannot have spaces in it or Apache service will not start.

A sample `load_balance.txt` file looks like the following:

```
##
## load_balance.txt -- rewriting map
##
dynamic localhost:8081|localhost:8082|localhost:8083
```

The process_count being set to 3 corresponds to the three processes running at port 8081, 8082, 8083 respectively. You can see the three startup.py processes by running:

```
ps -wef | grep python
```

The higher the process_count is set, the more memory is needed in the system. If you set it to 10 for heavier usage, the load_balance.txt should look like this:

```
## this is in one line in the file
dynamic localhost:8081|localhost:8082|localhost:8083|localhost:8084|localhost:8085| ↩
    localhost:8086
        |localhost:8087|localhost:8088|localhost:8089|localhost:8090
```

Restart the apache service and tactic service after changing their config files.

To verify if you have set up load-balancing correctly, please follow this tutorial in our support site:

```
http://support.southpawtech.com/tutorial/simple-load-balance-test
```

For more information on load balancing using mod_rewrite, refer to the mod_rewrite documentation located at:

http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html

9 Troubleshooting

9.1 TACTIC Service Troubleshooting

1. **I get a proxy error in my browser that looks like the following:**

   ```
   Proxy Error

   The proxy server received an invalid response from an upstream server.
   The proxy server could not handle the request /POST /tactic/admin/
   <http://192.168.14.113/tactic/admin/>/

   Reason: *Error reading from remote server
   ```

   This message tells you the TACTIC service has stopped. If you are running other processes on port 80 (for example, Skype), this can cause a problem with the TACTIC service. Stop the conflicting service. Then restart TACTIC in the TSI by logging in as root and running the command:

`service tactic start ` 2. **What if the TACTIC host memory is full?**

If the memory is full, restarting the TACTIC service will remove this memory. If the problem persists or occurs again, contact Southpaw. A memory leak can occasionally occur when a user is repeatedly viewing a very large table with a huge number of entries (>5000). Restarting the service will clear this memory. 3. What if the TACTIC host CPU is at 100%?

Use the Windows Task Manager to kill the process that is taking 100% of the CPU. The TACTIC service will automatically recreate the process. Normally, this will clear the problem. 4. What if the user sees a Proxy Error?

This means that the IIS service cannot contact the TACTIC application server. Look in the Windows services and restart the TACTIC service. If the problem persists, contact Southpaw.

To diagnose issues with TACTIC, several tools can be used.

This command line tool reveals active ports on the local machine.

```
leowiz:~ root# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.80.128.ssh     192.168.80.1.46575     ESTABLISHED
tcp4       0      0  *.*                    *.*                    CLOSED
tcp46      0      0  *.http                 *.*                    LISTEN
tcp4       0      0  192.168.80.128.ssh     192.168.80.1.50126     ESTABLISHED
tcp46      0      0  *.vnc-server           *.*                    LISTEN
tcp4       0      0  *.ssh                  *.*                    LISTEN
tcp6       0      0  *.ssh                  *.*                    LISTEN
tcp4       0      0  localhost.ipp          *.*                    LISTEN
tcp6       0      0  localhost.ipp          *.*                    LISTEN
udp4       0      0  *.net-assistant        *.*
udp4       0      0  192.168.80.128.ntp     *.*
udp6       0      0  localhost.ntp          *.*
udp4       0      0  localhost.ntp          *.*
udp6       0      0  localhost.ntp          *.*
udp6       0      0  *.ntp                  *.*
udp4       0      0  *.ntp                  *.*
udp6       0      0  *.mdns                 *.*
udp4       0      0  *.mdns                 *.*
Active LOCAL (UNIX) domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
41ad880 stream      0      0  46ad830        0        0        0 /private/tmp/ARD_ABJMMRT
41ce770 stream      0      0        0  3e22e58        0        0 /var/run/mDNSResponder
3e22e58 stream      0      0        0  41ce770        0        0
41ce5d8 stream      0      0        0  41ceb28        0        0 /var/tmp/ ↩
    SCDynamicStoreNotifyFileDescriptor-20739
41ceb28 stream      0      0        0  41ce5d8        0        0
41cef68 stream      0      0        0  3e22dd0        0        0
3e22dd0 stream      0      0        0  41cef68        0        0
3e22088 stream      0      0        0  41ad330        0        0
41ad330 stream      0      0        0  3e22088        0        0
41ce6e8 stream      0      0        0  41ce908        0        0
leowiz:~ root#
```

This command line tool allows port-scanning of an IP address. It can be used to remotely determine if a host is running HTTP.

```
[root@espresso ~]# nmap 192.168.80.128

Starting Nmap 4.76 ( http://nmap.org ) at 2010-03-23 15:58 EDT
Interesting ports on 192.168.80.128:
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
5900/tcp open  vnc
MAC Address: 00:0C:29:9D:E8:46 (VMware)
```

```
Nmap done: 1 IP address (1 host up) scanned in 6.66 seconds
[root@espresso ~]#
```

In the above example, an http service seems to be running on the machine with the IP of 192.168.80.128

```
[root@tactic-tsi ~]#  nmap localhost
Starting Nmap 4.52 ( http://insecure.org ) at 2010-03-23 16:10 EDT
Interesting ports on tactic-tsi.localdomain (127.0.0.1):
Not shown: 1705 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5432/tcp  open  postgres
8081/tcp  open  blackice-icecap
8082/tcp  open  blackice-alerts
10000/tcp open  snet-sensor-mgmt

Nmap done: 1 IP address (1 host up) scanned in 0.101 seconds
[root@tactic-tsi ~]#
```

In this example, TACTIC and Apache are running on the same local machine, with TACTIC taking up 2 processes, and 2 ports, 8081, and 8082

9.2 Monitor Server Performance

10 Maintenance

10.1 TACTIC Backup and Restore

Like any other data producing service, TACTIC requires a backup regimen. The data in that TACTIC produces and manages must be backed up in accordance with the backup policy of the departments that manage the TACTIC service.

TACTIC managers should create a backup policy for TACTIC data. A backup policy is extremely important, as data loss within the TACTIC service is catastrophic.

TACTIC manages data in three parallel scopes:

- `versioned assets` Versioned assets are all the files that are checked in and out of the tactic service.
- `asset metadata` Asset metadata are all the data produced by user interaction with the tactic service.
- `project metadata` Project metadata are all data produced by the creation of a project within tactic the tactic service.

Tactic data is all of equal importance. The data is all interrelated, and thus all data must be treated with equal importance in the backup regimen.

The frequency of backup of tactic data depends on the policies of the TACTIC managers. An examination of the frequency of use of the tactic service, combined with the down time potentially required to back up tactic data should be considered.

the only consideration that tactic managers should take into account when creating a backup policy for tactic data is that tactic data produced is synchronized. This is accomplished via a transaction system within tactic. This transaction system relies on co-services to store transacted data. During the backup process, tactic and its co-services should be stopped to prevent any loss of synchronization to these co-services.

*Versioned assets*Versioned assets are all the files that are checked in and out of the tactic service. Typically the tactic service works in conjunction with a co-service such as a network file storage service to accomplish its tasks of asset management. Thus

the task of backing up tactic of versioned assets onto these systems usually overlaps with the file management policies of the IT department. Most enterprise class network file storage systems, and third party file backup systems can accomplish the task of backing up tactic assets. Redundant raid storage, snapshots, and offsite backups can all be utilized to accomplish version and asset backups.

It is highly recommended that you back up your TACTIC server using a regular schedule. In the event of a hardware failure, you will be able to restore your TACTIC server fully from the latest backup if you have taken the following three backup steps:

1. Back up the *assets* directory
2. Dump the TACTIC database
3. Back up the TACTIC program directory

> **Note**
>
> You may want to stop the tactic service while you do your backup process. To do so, run the following commands:
>
> ```
> service tactic stop
> service tactic start
> ```

The *assets* directory is the repository where TACTIC stores all of the asset files. By backing up this directory, you can restore all of your asset files in the event of a database crash into a clean TACTIC-managed directory structure.

> **Note**
>
> You may need to contact your TACTIC server administrator if your *assets* directory has been redirected to another location.

Windows

The *assets* directory on a Windows install is located by default in:

```
C:\assets
```

Linux

The *assets* directory on a Linux install is located by default in:

```
/home/apache/assets
```

To restore the TACTIC *assets* directory, you have to restore your backup to the current *assets* location.

To dump the TACTIC data base you need to log in to the PostgreSQL database and perform a database dump.

> **Note**
>
> For both Linux and Windows the database file is in the *assets* directory, although you may redirect the location to your backup location. Also, the TACTIC default for the database is to have no password. If you have added a password, you may need to enter it into your postgres commands with the `-P` tag.

Windows

```
pg_dumpall -U postgres -c > c:\assets\tacticDatabase.sql
```

Linux

```
pg_dumpall -U postgres -c > /home/apache/assets/tacticDatabase.sql
```

To restore the TACTIC database run the command:

```
psql -U postgres < tacticDatabase.sql
```

This last step, although not completely necessary, is recommended because it takes a snapshot of your TACTIC source code and project settings at the time of backup.

Windows

The folders to back up are:

`C:\Program Files\southpaw\tactic`

`C:\Program Files\southpaw\projects`

Linux

The folders to back up are:

`/home/apache/tactic`

`/home/apache/projects`

To restore the TACTIC program files, you have to restore the `tactic` and `projects` directories to their original locations.

10.2 Database Backup Automation

Overview

The TACTIC database needs backing up as part of a routine maintenance on the TACTIC system.

Procedure

1. Log into the TACTIC server as root
2. Make a backup of the existing /etc/crontab -the original crontab should look something like this:

```
SHELL=/bin/bash PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

3. The following line is an example of code that will write the date out to a file and run on the 15th minute of every hour:

   15 * * * * root date > /root/out

   Modify this example code to run in the next few minutes and add it to the crontab. To see if the cron script ran, after the time has passed, verify that the script wrote out the time to the file` /root/out`. The following line is an example of how to backup the TACTIC database every day at 3:45am yum

   45 3 * * * root /usr/bin/pg_dumpall -U postgres > /tmp/my_tactic_db_my_date

   > **Note**
   > Be careful as this example will overwrite the backup every day . Another more clever way is the name the file with the date:

   45 3 * * * root /usr/bin/pg_dumpall -U postgres > /root/my_tactic_db_`date +\%Y-\%m-\%0e_\%H:\%m:\%S`

   or

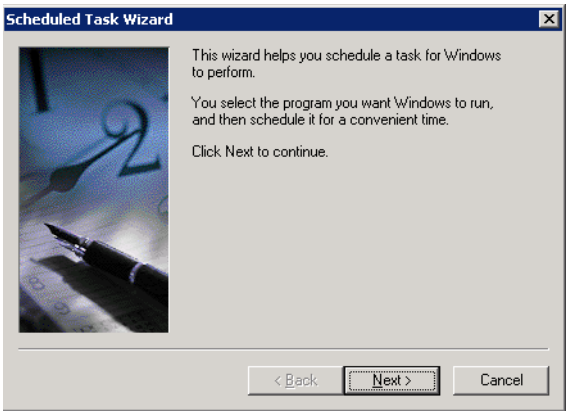
   45 3 * * * root my_date=`date +\%Y-\%m-\%0e_\%H:\%M:\%S`; /usr/bin/pg_dumpall -U postgres > /tmp/my_tactic_db_${n

   > **Important**
   > *From crontab manpage:*The “sixth” field (the rest of the line) specifies the command to be run. The entire command portion of the line, up to a newline or % char- acter, will be executed by /bin/sh or by the shell specified in the SHELL variable of the cronfile. Percent-signs (%) in the command, unless escaped with backslash (\), will be changed into newline characters, and all data after the first % will be sent to the command as standard input.

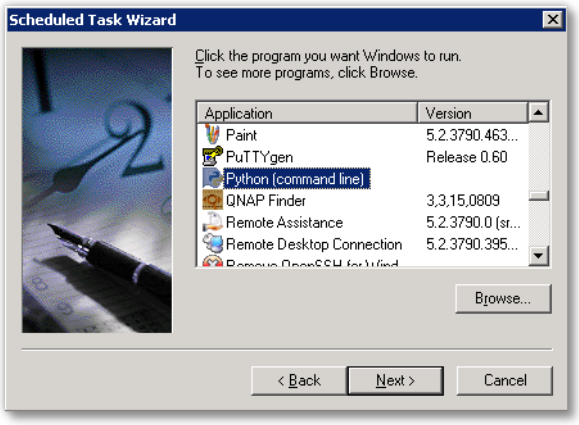
By using Task Scheduler can schedule tasks such as to automate DB backup to run at a time that is most convenient. Task Scheduler starts each time Windows is started, and runs as a background process. With Task Scheduler, you can: Schedule a task to run daily, weekly, monthly.

To add a scheduled task:

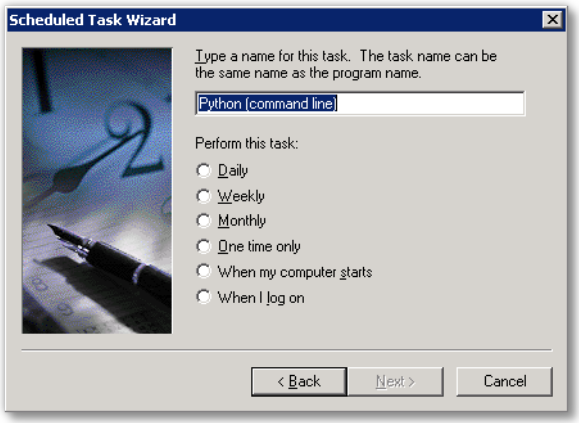
1. Create a windows batch script to back up the database
2. Click Start→Control Panel→Scheduled Tasks→Add scheduled Task
3. Create the task.



4. Use the provided python script to dump the database.



5. Run the task according to the schedule required.



6.

7. At the concluding stage of the scheduled tasks wizard, be sure to check the "Open advanced properties for this task when I click Finish"

Edit the command line to read "<path to python>\python dbbackup.py", and click "OK"

10.3 TACTIC logs

In a default installation tactic stores its logs in a single file. For proper maintenance, this file must be rotated by the host operating system. This action must be set in the host operating system. As tactic has no facility to control the log files it makes as of yet, it may be necessary to configure an external tool to rotate logs.

The host operating system must have a facility to rotate text logs on a regular chronological basis.

Logrotate is used as a tool to rotate text files on a chronological basis. The tool allows automatic rotation, compression, removal and mailing of log files.

Directives

This is a partial list of logrotate directives.

- **missingok**: If the log file is missing, go on to the next log file without issuing an error message.
- **copytruncate**: Truncate the original log file to zero size in place after creating a copy, instead of moving the old log file and optionally creating a new one
- **rotate 7** : Log files are rotated 7 times before being removed or mailed to the address specified in a mail directive. If count is 0, old versions are removed rather then rotated.
- **compress**: Old versions of log files are compressed with gzip to save disk space.
- **notifempty**: Do not rotate the log if it is empty
- **sharedscripts postrotate /etc/init.d/lighttpd reload endscript**:

Default configuration

Logrotate's main configuration file `/etc/logrotate.conf`

Applies the main configuration settings from this file to all log rotation settings, unless overridden by individual directives on a per-log basis.

```
# see "man logrotate" for details
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
```

```
include /etc/logrotate.d
# no packages own wtmp -- we'll rotate them here
/var/log/wtmp {
monthly
create 0664 root utmp
rotate 1
}
```

Example Service Configuration

This is an example logrotate configuration for HTTP. The configuration file is `/etc/logrotate.d/httpd `

```
/var/log/httpd/*.log {
 weekly
 rotate 52
 compress
  missingok
  notifempty
  sharedscripts
  postrotate
      /bin/kill -HUP `cat /var/run/httpd.pid 2>/dev/null` 2> /dev/null || true    endscript
}
```

On the example linux system, service or server specific configurations are stored in `/etc/logrotate.d` directory. For example here is sample apache logrotate configuration file:

In this example, HTTP logs are not deleted until they are at least a year old, and are rotated on a weekly basis. There are other configuration options available, but they will not be discussed in this tutorial. Please look online for more information on these other options. Depending on the requirement, each log file may be handled at different intervals.

The lines between postrotate and endscript (both of which must appear on lines by themselves) are executed after the log file is rotated. These directives may only appear inside a log file definition. In our case we are reloading lighttpd.

Example TACTIC configuration

The configuration goal is get logrotate to get the tactic log files rotated for our needs and requirements.

This is an example logrotate configuration for HTTP. The configuration file is `/etc/logrotate.d/tactic `

Remember to restart the logrotate services after any additions to logrotate configuration.

```
/home/apache/tacticTemp/log/stdout.log {
    notifempty
    daily
    rotate 365
    missingok
    copytruncate
    create 0600 apache apache
}
```

11 TACTIC Server VM

11.1 TACTIC VM Setup and Install

The TACTIC VM lets you run a virtual machine on Windows or Linux, and provides an easy way to evaluate the TACTIC platform without changing your operating system. The VM requires VMware Player for Windows or Linux,

Q. Will my computer be adversely affected if I run VM ?

A. Your computer will not be affected by the VM. Aside from requiring a large amount of memory (the TSI unzips into a folder needing at least 5-10 GB), the TACTIC virtual machine does not interact with your host machine. At the end of the evaluation

period, you can either continue using the TSI for production, migrate your data to a new full server install or simply delete the TSI directory. (Note that if you delete the TSI directory, all evaluation data you entered into the TACTIC evaluation is destroyed.)

Request and download the TACTIC Server Image (TSI)

The TSI is available on request from Southpaw Technology. When you contact the Southpaw sales or support representatives and set up an evaluation account, they will send you a URL and login/password information so you can begin the trial.

For more information, visit:

http://www.southpawtech.com

Download VMware Player or server

The VMware player is a license-free player you can download from

http://vmware.com/download/player/ [http://vmware.com/download/player/]

VMware currently supports Microsoft Windows XP, Windows Vista and many flavors of Linux. To streamline the evaluation process, we chose the Fedora Linux-based operating system to create the TSI virtual machine image. (Although you will be running the VM on any of many supported operating systems, you will see the Fedora Linux when you use the VM.)

Download VMware Server

The VMware server is a full-featured service for Virtual machines

VMware server is a service that allows users to run VMs silently in the background. No window is needed, and the server can be configured to run the VM on host machine startup. Additionally, there are several interesting options that allow for more flexible operation of a VM.

- **Web UI**
- **Pause** The ability to pause a VM is useful for cycling the VM host.
- **Snapshot a VM** The VMs current state can be saved and recalled in the future. Note that if the VM is reverted to a snapshot, then any changes that have occurred since the snapshot was taken will be permanently deleted.
- **Change hardware** The VM can be modified to change hardware. For example, another network interface, or a USB interface

> **Note**
>
> The instructions for running a VM can be found in the VMware server documentation. Running the VM from VMware player is covered in this document.

Web Browser

We recommend you use Mozilla Firefox as your web browser to access the TSI. You can download Firefox at:

http://www.mozilla.com/en-US/firefox/ [http://www.mozilla.com/en-US/firefox/]

Login Info

There are three sets of login info to remember:

1. VM Linux user

   user: root password: south123paw 2. TACTIC UI in browser

   user: admin password: tactic 3. Windows Samba share

   user: apache password: south123paw

Networking

When VMware software is installed, several new network interfaces can become available, depending on the options selected during the installation process. Although any one of the interfaces can be used to connect to the VM, it is advised to install all three network interfaces, as this allows the most flexibility

Note

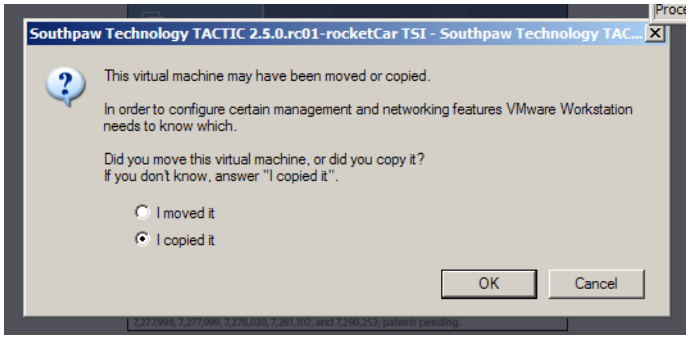
Successful operation of a virtual machine may require some input from system administrators within a network environment.

- NAT
  - If it becomes necessary to isolate the VM on a host machine, then the NAT interface can be used to connect to the VM.
  - Internet and LAN connectivity can be utilized through the NAT interface if the host machine is connected to an external network such as the internet or a LAN. Allowing the VM to communicate with a LAN may require changes to the routing system on the LAN. If this is a requirement, then bridged networking may be the better option.
  - NAT is a complete network that originates at the host machine, and is a self-contained router with DHCP services. The VM, once run, will usually ask the host machine for an IP address to assign to a network interface. The host machine can then connect to the VM interface through this IP.
- Bridged
  - If there are machines other than the host machine that are required to connect to the VM, then the bridged connection should be used.
  - The only requirement that bridged networking has is that there are appropriate services for the operation of the interface, such as DHCP on the host machines network. If these services do not exist, then the VM network interface needs to have manually set options.
- Host Only
  - To isolate the TSI completely, the “Host Only&#xfffd;? option can be used.
  - Host Only is a complete network that originates at the host machine, and is a self-contained router with DHCP services. The VM, once run, will usually ask the host machine for an IP address to assign to a network interface. The host machine can then connect to the VM interface through this IP.
  - The VM is completely isolated from the host machines LAN. There is no network connectivity, therefore no LAN or internet will be available on the VM.

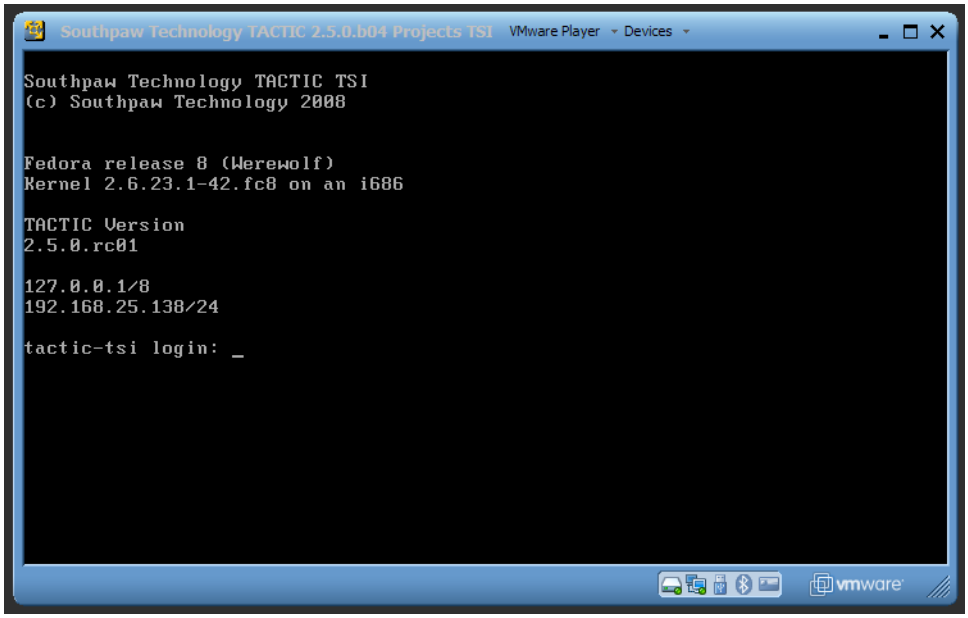
After you start up the VMware Player, click on the **Open** option to browse for a virtual machine.

Browse to where the unzipped VM is. Open the `TACTIC_VM_3.8.0.v04.vmx` file. The TSI will start up in the VMware Player.

When you first start the VM, you are presented with a window asking whether the virtual machine was moved or copied. Click the "I copied it" button.



Start-up messages scroll by after the Linux kernel boots in the VM window. After the boot-up is completed, you will see this prompt:



If you see this prompt, the TACTIC VM has started up correctly. If it does not show an IP address (for example, a number like "192.168.25.138"), it means the virtual machine has not picked up an IP address from a DHCP server. This IP address is required, so you must contact your IT department to fix this situation. (Refer to the Troubleshooting Section.)

Write down the IP address and any names you see such as "localhost.localdomain". (You will need this information in a next step.)

In special evaluation situations, there is a requirement for more disk space than has been allocated for a standard evaluation (i.e. a large amount of assets are to be stored, then the TSI "assets partition" can be expanded.

Steps to expand disk space on a VM

1. Shut down TACTIC

   To shut down TACTIC, issue the command

   service tactic stop 2. Unmount the "TACTIC" drive

   umount /home/apache 3. Run the logical Volume expansion tool

   lvextend -L+2G /dev/mapper/VolGroup00-LogVol00

   This example command resizes the logical volume by +2 G, as seen in the command itself. The volume can be expanded to the total size of the virtual disk. 4. Run the filesystem consistancy checker to make sure the filesystem is ok

e2fsck -f /dev/mapper/VolGroup00-LogVol00 5. Resize the filesystem on the logical volume

resize2fs /dev/mapper/VolGroup00-LogVol00 6. Re-mount the "assets" drive

```
mount /home/apache/assets
```

2. Restart TACTIC

```
service tactic start
```

To make your evaluation easier, the TSI uses passwords that are easy to guess and a Samba share. This could cause security issues, so your IT department must be aware of your use of the TSI.

Tactic Samba Network Details

Workgroup	Southpawtech
Machine	TSI 2.0.0
Share	Tactic
Share User	Apache
Share Password	south123paw

If at any point during the usage of the VM there is a need to reset the VM back to its original state, simply delete or archive the directory with the TSI data, and re-expand the zip file that was downloaded from the community site.

Your TACTIC VM server is set up.

Contact your TACTIC Representative for support details at support@southpawtech.com

11.2 TACTIC VM Troubleshooting

1. **The IP address does not show up when I start up the VM.**

   It is possible that an IP address was not assigned because your host machine is not connected to a network. Or, you have a special network configuration that may require one of the following settings: * Network with DHCP: bridged * Network without DHCP: NAT * No network: host-only

   *_*Changing Windows Special Network Configurations*_*

   In the top right corner of your VMware player, click on the down arrow beside the Ethernet button and choose **host-only** , **NAT**, or **bridged**. Make sure the **Connected** menu item is checked.

   Changing Linux Special Network Configurations

   In the top right corner of your VMware player, right-click on the Ethernet button and choose **host-only**, **NAT**, or **bridged**. Make sure the **Connected** menu item is checked.

   If the problem persists, refer to the VMware Player documentation by pressing F1. 2. **I do not remember the IP address of the VMware virtual machine.**

   In the VMware player, run the following command in the shell:

   ```
   ifconfig
   ```

   The IP is the internet address; for example, "192.168.14.101."

   Refreshing the IP Address

   If you change the Ethernet settings, you may need to restart the network card in the VM. First, you will need to know its number: run the command `ifconfig` in the VM shell. You should see a name similar to "ethX", where X is the arbitrary number of the interface. To restart "ethX", run the following commands:

   ```
   ifdown ethX
   ifup ethX
   ```

2. **I get a proxy error in my browser that looks like the following:**

```
Proxy Error

The proxy server received an invalid response from an upstream server.
The proxy server could not handle the request /POST /tactic/admin/ <http ↩
    ://192.168.14.113/tactic/admin/>/

Reason: *Error reading from remote server
```

This message tells you the TACTIC service has stopped. If you are running other processes on port 80 (for example, Skype), this can cause a problem with the TACTIC service. Stop the conflicting service. Then restart TACTIC in the VM by logging in as root and running the command:

`service tactic start `

11.3 Moving VM Data

To transfer data from one VM to another, you need to do the following:

In the old TSI VM shell:

```
cd /home/apache
```

```
pg_dumpall -U postgres -c > tactic_database.sql
```

In your host machine assuming windows, start a new window of Windows Explorer in the address, type in:

```
\\<your old VM IP address>
```

when prompted for uid, password, type in :

```
apache
```

```
south123paw
```

navigate into the apache folder, you should see the file **tactic_database.sql**, **assets folder** and **projects folder**

You need to move the old VM's **tactic_database.sql**, **assets folder** and **projects** folder to the same location of the VM you just downloaded and installed.

Same as above, open a new windows explorer, but type in the new VM IP address and carry out the copying

In the new TACTIC VM shell, assuming you have copied the **tactic_database.sql**, **assets folder** and **projects folder** in the new IP from the VM, Type in:

```
dropdb -U postgres sthpw
```

```
psql -U postgres < tactic_database.sql
```

Upgrade the TACTIC database in the VM according to the version you are upgrading to

Now you can just shut down the old VM by closing the VMware player window

12 Client Setup

12.1 Client Computer Setup Requirements

The following are requirements for a client using TACTIC.

TACTIC Temp Directory Creation

TACTIC needs to be able to create the TACTIC temporary directory. TACTIC uses this directory as scratch pad space to store temporary files.

The location of this directory depends on the client operating system:

- Linux: `/tmp/sthpw`
- Windows: `C:/sthpw`
- Mac OS X: `/tmp/sthpw`

Set environment variables

`$TACTIC_ASSET_DIR`

This environment variable is used to point to the root of `asset_base_dir` (the directory where assets are stored). This value defaults to "/home/apache/assets" in the TACTIC server. On Windows machines, this path is usually mapped as a letter drive (for example, Z:\) or as a UNC path (for example, \\server\volume\directory\). This environment variable is set by the server on the client machine during loading if it does not already exist.

On a per-project basis, you can set the project setting with the keys "win32_client_repo_dir" and "linux_client_repo_dir" so one project can have its asset directory mapped to Drive Z: and another mapped to Drive T:.

Browser Plugins

With the security concerns over Java plugin, TACTIC has lowered the use of Java plugin in many cases to communicate with the client computer's operating system. HTML5 features, where applicable, is adopted instead. A good example is uploading of files.

To download the latest version of the Java plugin, go to www.java.com