# Table of Contents

Overview ....................................................................................................................................................... 3
Introduction .................................................................................................................................................. 4
  Service-Oriented Architecture .................................................................................................................. 4
  Three-Tier Architecture ............................................................................................................................ 4
Technologies Used ........................................................................................................................................ 5
  System ................................................................................................................................................ 5
  Presentation Layer ................................................................................................................................. 5
  Business Layer ................................................................................................................................... 5
  Data Layer ............................................................................................................................................. 5
Adaptor Interfaces ........................................................................................................................................ 6
Core API .................................................................................................................................................... 8
  Kernel ................................................................................................................................................. 8
  Logging ............................................................................................................................................. 9
  Notifications .................................................................................................................................... 9
  Scheduler .......................................................................................................................................... 10
Common Data Structures .......................................................................................................................... 11
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notifications</td>
<td>11</td>
</tr>
<tr>
<td>Database Schemas</td>
<td>13</td>
</tr>
<tr>
<td>Authentication</td>
<td>14</td>
</tr>
<tr>
<td>Historic Data &amp; Errors</td>
<td>15</td>
</tr>
<tr>
<td>Sequence Diagrams</td>
<td>16</td>
</tr>
</tbody>
</table>
Overview

Legend:
- Python Web Service
- .NET Web Service

SOAP Message

SQL Database

Core

Data Adaptor

Binary Adaptor

Numeric Adaptor

Presentation Layer

Business Layer

Web Interface

SMS Interface

User

Language Interface

Layer Hardware Devices

Video Surveillance

X10 Home Automation

Sensors

Keycard Authentication

Data Layer
Introduction

Terminus Security is built on a Service-Oriented Architecture (SOA) with a three-tier architecture influence. Combining the best elements of these two architectures, the result is a highly modular architecture based on services that can be easily broken down in the three basic tiers.

Service-Oriented Architecture

Service-Oriented Architecture is an architectural design pattern that concerns itself with defining loosely-coupled relationships between producers and consumers where existing or new functionalities are grouped into atomic services. These services communicate with each other by passing data from one service to another, or by coordinating an activity between one or more services.

Three-Tier Architecture

Probably the most common architecture design pattern, the 3-Tier Architecture is a client/server architecture model promotes the separation of responsibilities into three distinct layers. These layers are the Presentation layer, the Business layer and the Data layer. Each layer of the system depends only on the layer directly below it; layers may not depend on layers above them.
Technologies Used

System
The system runs on a Microsoft Windows 2003 Web Server Edition. In addition, the server will host Microsoft .NET 3.5 and Python 2.5 frameworks.

Presentation Layer
The Web User Interface allows the user to get an overview of the entire system, add/remove/edit components, arm/disarm the system and add/remove users. ASP.NET is the framework chosen for this component.

Business Layer
The core and adaptors can be written in any language due to the SOA architecture. However, in order to keep moderate consistency, C# and Python are the languages chosen for the development.

Data Layer
The data layer has to store all system configurations and logs, as well as archive module data, including text data (i.e. temperature sensor). Based on functionality needed, cost as well as integration with current technologies used, Microsoft SQL Server Express 2008 is the platform chosen.
Adaptor Interfaces

In order to support the modularity and flexibility required, the use of Adaptors and Modules is made.

Adaptors are web services which are used to hold a collection of sensors. This is how the system (Core) communicates with the devices. As long as the interface below is implemented, any devices can be added to the system:

Adaptor Interface
For the entire system to actually use these adaptors, an *Adaptor* object is created and manipulated:

![Adaptor Object](image)

*Adaptor Object*

The actual hardware interaction is done through *Modules*. They can be viewed as drivers to the system, and their sole purpose is to read the raw data and have it readily available through the *Adaptors* in a standardized manner. Below is a *Module* object:

![Module Object](image)

*Module Object*
Core API

As the main engine of the system, the Core contains four major components implemented in the form of web services.

Kernel

- The main logic system
- In charge of keeping track of adaptors and raising alarms
- The bridge between the User Interface and the Adaptors
- Complex state machine with three states: Armed, Secure and Alert.

Kernel Service Interface
Logging

- Global logging system
- Interacts with the SQL database
- Logs all events (alerts, debugging messages, notifications, errors)

**Logging Service Interface**

Notifications

- System used for sending alerts

**Notification Service Interface**
**Scheduler**

- System used to schedule events and trigger them
- Currently will be used for X10, however it allows any device to be scheduled to perform an action at a given time

*Scheduler Service Interface*
Common Data Structures

In order to make things easier and strongly typed a series of objects are used.

Notifications

In order to send a notification, a `NotificationMessage` object is created. This can be done either by the `Kernel` service or by the User Interface, depending on where the message is initiated from.

![Notification Message Object]

While the message can be any ASCII text, the `Level` is a pre-defined structure used internally.

![Message Level Structure]
The last element necessary for sending a message is the destination. This is in the form of a User which also contains information relevant, such as an email address and/or cell phone number.
Database Schemas
In order to support the features of this system, two databases are deployed: one for authentication, used mainly by the ASP.NET Web Interface, and one for storing historic data and errors.
Historic Data & Errors
Sequence Diagrams

Arm Sequence

Disable Camera
Login

Token Authentication
Power Failure

1. UPS
2. UPS Adaptor
3. Kernel
4. Notifications
5. Logging

- Read Data
- Pol() → RaiseAlert()
- Send() → WriteMessage()