Chapter 8

# Using the PDSA Security System

The PDSA Framework security system provides a robust, extensible security model that is designed to serve the needs of enterprise customers by using application authentication, Active Directory authentication, mixed-mode authentication or custom authentication, along with an extensible authorization model based on customizable roles and permissions. The security system supports a variety of application scenarios, including stand-alone applications, multiple applications accessible to the same group of users, or even a single application deployed as a software-as-a-service system that supports multiple organizations.

## Overview of the PDSA Security System

Our security system is based on the *provider model*, which allows you to use Active Directory, application security or a “roll-your-own” approach. If you choose, you can even blend these two approaches as we have a mechanism to synchronize your Active Directory with the PDSA security tables that live in a database.

There are a few major components to the PDSA Framework to support our unique security system (Figure 1). First, we have a set of users that is global to your organization (or could be for just one application, you decide). Then for each Application you may create Roles and Permissions (described later in this chapter). You may also create an Application Service Provider (ASP) model by defining different Entities (companies) that are allowed to use the application. You can even assign roles and permissions to these entities, thus allowing you to turn on and off features for each entity.

The PDSA Security Provider library delivers the ability to authenticate users and obtain valid credentials that identifies each user's authorization level using roles and permissions. The illustration in Figure 1 depicts how different types of application security scenarios are able to leverage the PDSA Security Provider in concert with different kinds of security repositories. The PDSA Security Provider library will support desktop, Web and mobile applications.

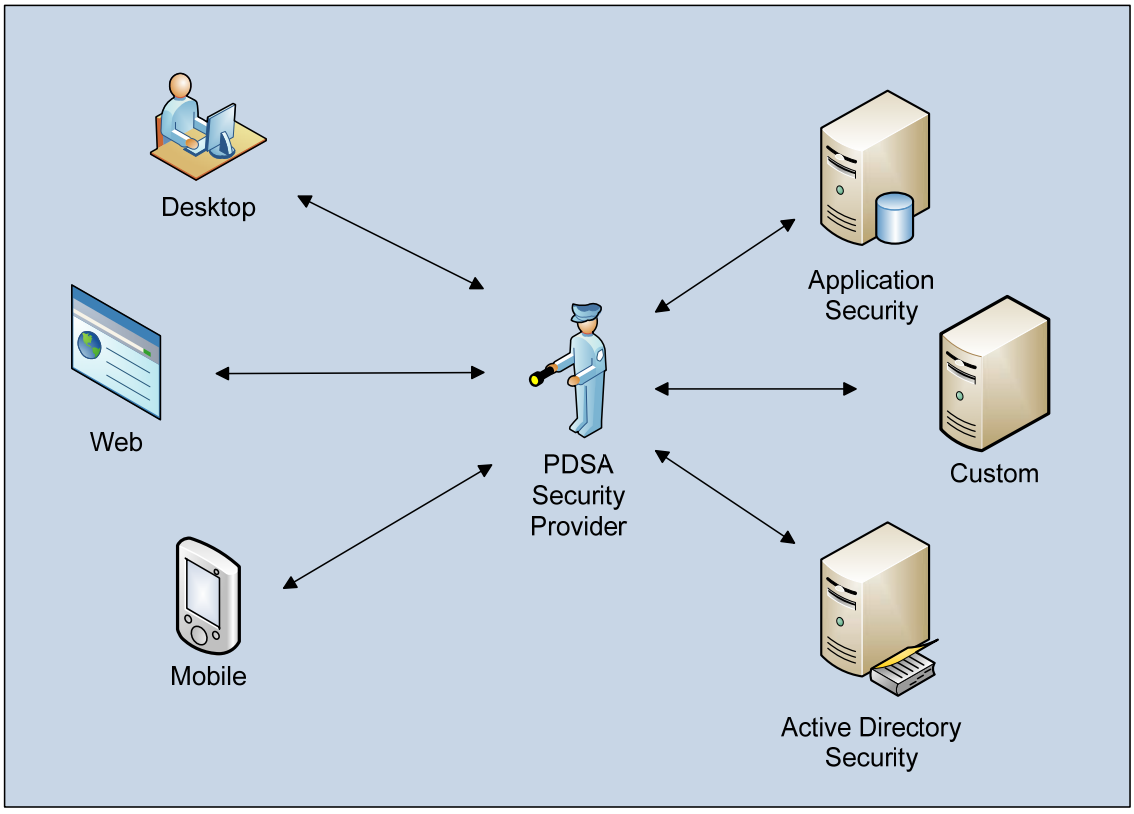


Figure 1: Security Scenarios

There are four modes supported by the PDSA Framework Security Providers: Application, Active Directory, Application Single Sign On, and Active Directory with PDSA Roles. Each of these is scenarios is described in detail in this document.

# Multi-Tenant Architecture

PDSA Framework supports multiple tenants on the same instance of an application created using the PDSA Framework.

# Application (Forms-Based) Security

Application security (most commonly known as “forms-based”) is the most common approach to securing custom applications. When using the PDSA Security Provider in conjunction with Application security, the application presents a login form, the user enters a login name and password, and then the PDSA Security Provider validates the user's credentials against the login name and password stored in a database table named ‘pdsaUser’.

After the provider validates the user’s credentials, it gathers the user’s roles and permissions from the database and then returns a principal object back to the application.

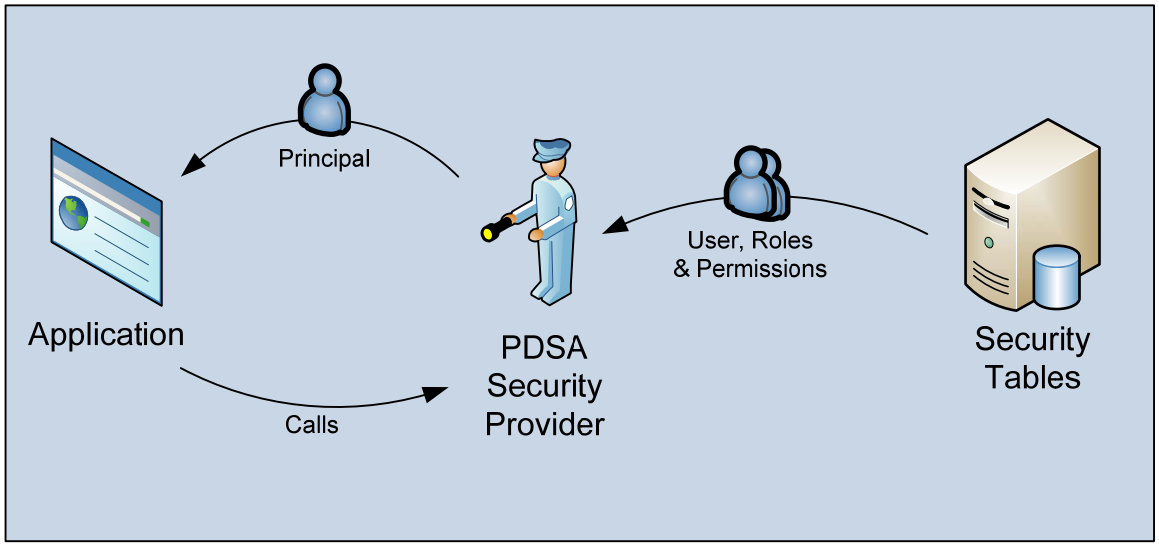


Figure 2: Application Security

# Active Directory Security

Active Directory security can be integrated into an application using the PDSA Security Provider in two ways. One, the application may use a login form where the user enters his or her windows login and password, or two, the application can automatically authenticate the user without presenting a login form.

In either case, if the user identity is valid, the provider creates a principal object that contains the user’s identity and can be used to determine whether the user belongs to a particular role, in this case an Active Directory group.

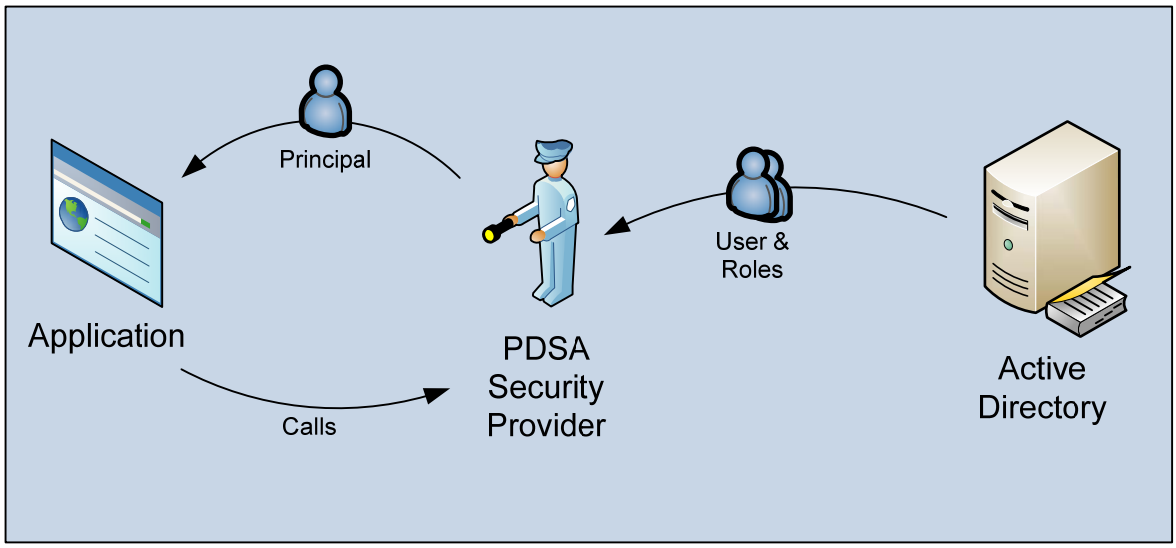


Figure 3: Active Directory Security

# Application Single Sign-on Security

Application Single Sign-on provides the ability to use Active Directory and Application security together.

When the provider identifies that the user is a valid Active Directory user, then it looks up the login name in the pdsaUser table. If the login name matches an entry in the pdsaUser table, then the user is signed in without presenting a login form, and the provider retrieves the user's roles and permissions from the security tables.

Otherwise, the login form is displayed and the user must enter a login name and password, which the provider uses to validate the credentials following the same approach described under Application Security.

All Active Directory users permitted to use the application must be registered in the pdsaUser table.

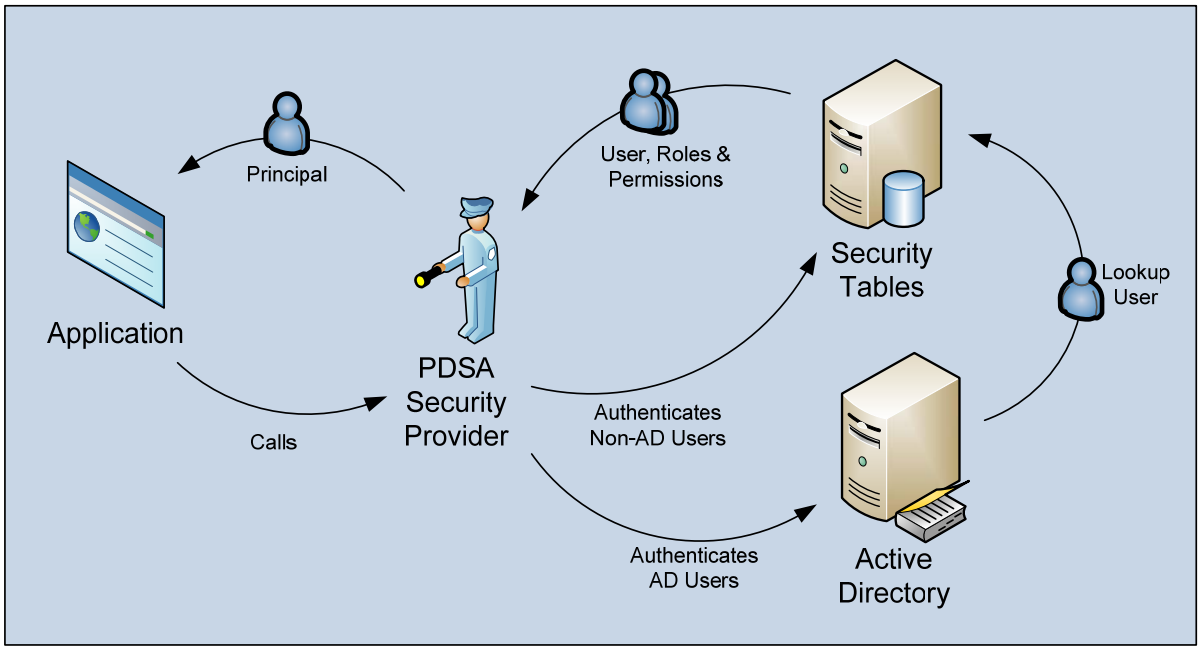


Figure 4: Application Single Sign On Security

# Active Directory with PDSA Roles

Using Active Directory with PDSA Roles provides the ability assign more granular permissions to Active Directory users than would otherwise be possible when using Active Directory alone.

After the provider identifies that the user is a valid Active Directory user, then the provider looks up all Active Directory roles that have been associated with the application, and then determines whether the current user has one or more of the defined roles.

Once the set of roles has been established, then the provider retrieves all permissions associated with the user's roles and creates a new principal object based on the Active Directory user, the set of roles assigned and the permissions.

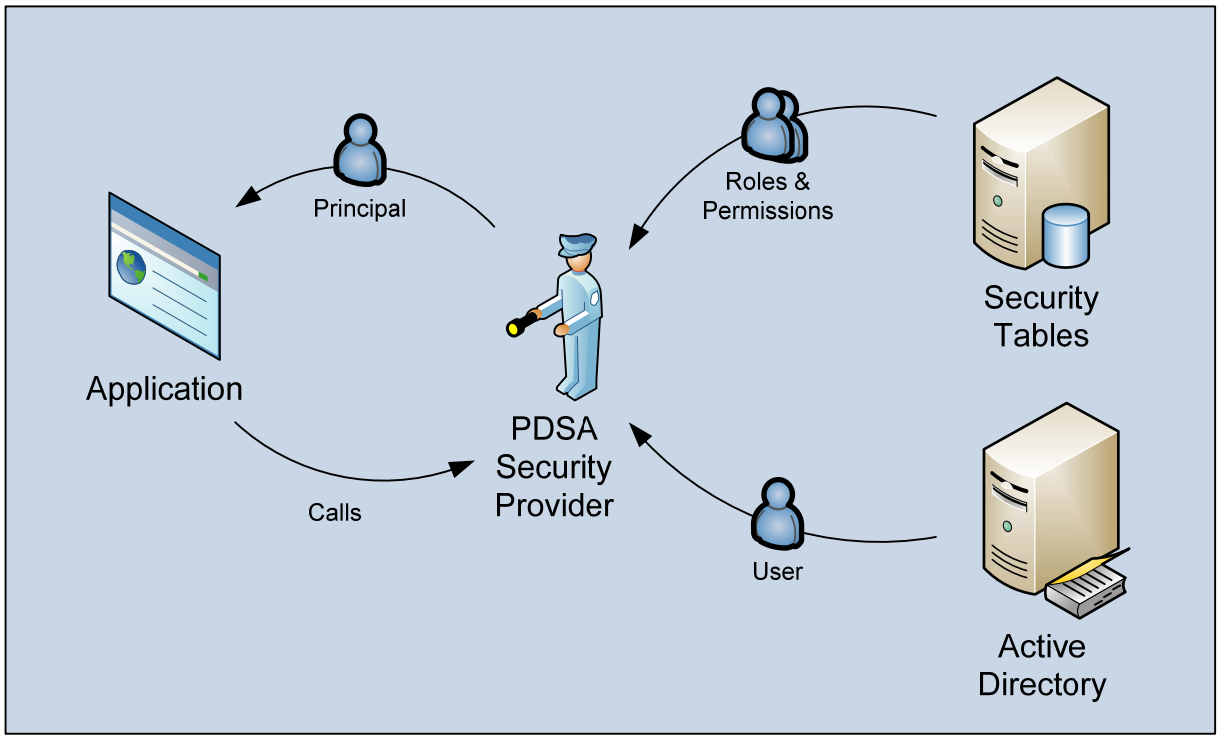


Figure 5: Active Directory with PDSA Roles Security

# User Stories when using PDSA Security

1. As a developer, I can setup a new application so that it comes with a single default entity with the name of the customer for whom I am building the application (“tenant”), a default system administrator account, and a default tenant administrator account so that I can build the application.
2. As a network administrator in organization using Active Directory, I can set up the application to use integrated security for a single organization (“entity” or “tenant”) only, so that all my existing Active Directory users can use the application without having to login again and I do not need to maintain another set of users and passwords outside of Active Directory.
3. As a system administrator, I cannot delete or rename the original “root” system administrator account.
4. As a system administrator, I can create a new tenant (e.g., “entity” or “company”) so that users in that organization will have access to the features of the application.
5. As a system administrator, I can create other system administrator accounts so that more than one person can administer the system.

6. As a system administrator, I can create a new tenant administrator account for a tenant so that a tenant can add users for their organization.
7. As a system administrator, I can create users for a tenant account so that I can help tenants manage their users.
    a. Subject to config setting “on/off”
8. As a system administrator, I can create menu items so that all users can access the features the system.
9. As a system administrator, I can create permissions so that I can restrict access to the features of the system.
10. As a system administrator, I can create roles so that I can assign permissions to groups of users.
11. As a tenant administrator, I can add other tenant administrator accounts so that more than one person in my organization can manage users.
    a. Subject to config: not all tenants may have administrators, not all tenant administrators may create new users.
12. As a tenant administrator, I can create users for my organization so that my users can access the application.
13. As a tenant administrator, I can create new roles for my organization, so that I can customize access to features of the system for my users.
14. As a tenant user, I can access menus, features and data that are applicable only to my organization so that I am confident that my data is secure.

# Application Security Entities

## Application

This is your application. Each application can define any of the following entities that will help control security for that particular application.

## Entity

Think of an Entity as a company, department, division that will use the Application. Each Application will have at least one Entity. An Entity may then choose from the set of users and assign them to that Entity.

## Users

You can define users just for a specific application, or you can have a global set of users that live in one database and all other applications built with the PDSA Framework may reference that user table and each user can be assigned to use an application or not. This allows you to specify which users can use which applications. Furthermore, we have a utility that will allow you to synchronize your Active Directory with our User table.

## Roles

Think of Roles as groups in Active Directory. Each Application will have at least one Role. A Role may be assigned to one or more Entities in the Application. A Role is used to group users together in order to allow them to perform a set of functionality within the application. Roles are typically defined as "Administrator", "User", "Supervisor", etc.

## Permissions

Permission is more granular than a Role and allows you to define very specific functionality for the application. An example of Permission is "Read", "Edit", "Delete" etc. You may setup one or more Permissions for an Application. Each Entity may select from the list of available Permissions to use. An Entity may assign Permission to one or many Roles.

# Things you can do with the PDSA Security System

1. As a developer, I can setup a new application so that it comes with a single default entity with the name of the customer for whom I am building the application (“tenant”), a default system administrator account, and a default tenant administrator account so that I can build the application.
2. As a network administrator in organization using Active Directory, I can set up the application to use integrated security for a single organization (“entity” or “tenant”) only, so that all my existing Active Directory users can use the application without having to login again and I do not need to maintain another set of users and passwords outside of Active Directory.
3. As a system administrator, I can search for tenants by tenant name, control the number of search results returned, and sort the search results, so that I can easily find and manage tenant accounts.

4. As a system administrator, I can add, edit and delete tenants (e.g., “entities” or “companies”) so that an organization will have access to the features of the application.
    a. A tenant account consists of a tenant name and a flag to indicate whether the account is active or disabled.
    b. When the tenant account is locked or disabled, then none of the users related to that account can login to the system.
5. As a system administrator, I can search for menu items by menu name, control the number of search results returned, and sort the search results so that menu items are easy to find and change.
6. As a system administrator, I can add, edit and delete menus and submenus items so that I can customize how users access the features the system.
    a. Menu items are global to all entities in a single application and cannot be managed or modified by an entity administrator.
    b. When a menu item has no submenu items, then it is possible to delete the record. If a menu item has submenu items, I can choose to continue with the deletion of the menu item and all submenus.
    c. A menu item consists of the menu item name, the URL to the page, the menu item sequence, the permission assigned, whether the permission is enabled or disabled. Sequence number controls the display order of a menu item on the menu.
7. As a system administrator, I can search for permissions by permission name and permission category, control the number of search results returned, and sort the search results, so that it is easy to find and manage permissions.
8. As a system administrator, I can add, edit and delete permissions so that I can restrict access to the features of the system.
    a. Permissions are global to an application and cannot be managed by an entity or tenant administrator.
    b. A permission record consists of a permission name, a short description and a permission category.
    c. When editing a permission record, it is possible to add and remove roles associated with the permission.
9. As a system administrator, I can search for a given role by role name, control the number of search results returned, and sort the search results, so that it is easy to find and manage roles.
10. As a system administrator, I can add, edit and delete roles so that I can assign permissions to groups of users.
    a. Roles may be either global to the application or entity-specific.
    b. A role consists of a role name and a short description of the role.

c. When adding or editing a role, it is possible to add and remove permissions associated with the role, and it is possible to add and remove users associated with the role.

d. As a tenant administrator, I can add edit and delete new roles for my organization, so that I can customize access to features of the system for my users.

11. As a system administrator, I can search for permission categories by permission category name, limit the search results, and sort columns to allow easier access to each item.

12. As a system administrator, I can add, edit and delete permission categories.

    a. A permission category consists of the category name and a short description.

    b. Permission categories are global to an application and cannot be managed or modified by an entity.

13. As an administrator, I can search for users by first name, last name, email address, login ID, status and sort results to allow easier access to each item.

14. As an administrator, I can add, edit and delete user accounts so that users can access the system.

    a. User account consists of First Name, Last name, Initials, Login ID, email address, locked flag, status, password, security question and security answer.

    b. System administrator can designate that the user is global or for a specific entity.

    c. When adding or editing a user it is possible to add or remove roles associated with the user

    d. Administrators can re-set a user's password.

    e. Administrators cannot delete or rename the original "root" system administrator account.

    f. System administrators can create other system administrator accounts by creating a global user and assigning the system administrator role.

    g. An entity administrator can add other tenant administrator accounts by creating an entity user and assigning the user the administrator role. This should be subject to configuration identifying whether or not the tenant administrator can create additional admin accounts.

As a tenant user, I can access menus, features and data that are applicable only to my organization so that I can use the features of the system with my own data.

# MVC: Security through Code

In MVC you have access to the User object from within the controller class. You should assign the User object to the Principal property of your ViewModel class.

```
public ActionResult Index()
{
  ProductViewModel viewModel = new ProductViewModel();

  // Assign Principal to View Model
  viewModel.Principal = User;

  viewModel.HandleRequest();

  return View(viewModel);
}
```

The base class for the view models has an IsInRole() method. You can use this to eliminate complete controls from your view as shown in the following code:

```
@if (Model.IsInRole("Admin"))
{
  <div class="form-group">
    @Html.LabelFor(m => m.DetailData.Cost, "Cost")
    @Html.TextBox5For(m => m.DetailData.Cost,
                    HtmlExtensions.Html5InputTypes.number,
                    "Cost",
                    "Enter the Cost",
                    true, true)
  </div>
}
```

To see if a user has permissions, you will also use the Model object from within your CSHTML view.

```
@if (Model.HasPermission("EditCost"))
{
  <div class="form-group">
    @Html.LabelFor(m => m.DetailData.Cost, "Cost")
    @Html.TextBox5For(m => m.DetailData.Cost,
                    HtmlExtensions.Html5InputTypes.number,
                    "Cost",
                    "Enter the Cost",
                    true, true)
  </div>
}
```

# Web Forms: Security through Code

In Web Forms you have access to the User object on a page. We replace the User object with our own implementation of IPrincipal. This means you can still use the built-in IsInRole() method.

```
if(User.IsInRole(“Administrator”))
  txtCost.Enabled = true;
else
  txtCost.Enabled = false;
```

To access the additional permissions and other properties in the PDSA security system you use the Principal object. For example, in ASP.NET you can access the Principal object on a page with the following code:

```
if(base.Principal.HasPermission(“Product Name Field”))
  txtProductName.Enabled = true;
else
  txtProductName.Enabled = false;
```

# Access Settings from Config File

You can always access the PDSASettings class in order to get at the information from the Configuration file of your application. For example to get the maximum amount of attempts that a user may try to login you would use the following line of code.

```
int num = PDSASettings.AllValue.Security.Login.MaxAttempts;
```

# Menu Item Settings for Active Directory with PDSA Roles Security

The following menu items should be set to inactive when Active Directory with PDSA Roles security is used: Maintenance - Users, Maintenance - Entities, My Profile - My Profile and My Profile - Change Password. Each of these menu items refers to data associated with the pdsaUsers table which is not used in this security model.

# Summary

This chapter described the PDSA Security system and all the various scenarios to which you can apply our system.