SecMan.Data

The SecMan.Data project is the data layer for the SecMan.

It is a C# .NET 8.0 DLL built with Visual Studio 2022 and exposes a set of interface objects. The primary object, SecManDb, should be instantiated by the SecMan business layer, it contains all the methods to configure the security data.

The data is stored in a database managed by Microsoft Entity Framework. For the final release this will be an encrypted SQLCipher database, however for initial development it will be SQLite.

Contents

[1 Development Setup 2](#_Toc176273574)

[1.1 Folders 2](#_Toc176273575)

[1.2 Setup Process 2](#_Toc176273576)

[1.3 Database Initialisation 2](#_Toc176273577)

[2 Database 3](#_Toc176273578)

[3 Interface objects 3](#_Toc176273579)

[3.1 Interface Object - SecManDb 3](#_Toc176273580)

[3.1.1 Constructor 4](#_Toc176273581)

[3.1.2 Public Methods – SuperUser 4](#_Toc176273582)

[3.1.3 Public Methods – User 4](#_Toc176273583)

[3.1.4 Public Methods – Role 4](#_Toc176273584)

[3.1.5 Public Methods - Device Definition Methods 4](#_Toc176273585)

[3.1.6 Public Methods – Device 4](#_Toc176273586)

[3.1.7 Public Methods – Zone 5](#_Toc176273587)

[3.1.8 Public Methods - System Features 5](#_Toc176273588)

[3.1.9 Public Methods – Values 5](#_Toc176273589)

[3.2 Interface Object – SuperUser 5](#_Toc176273590)

[3.2.1 Public Methods and Properties 5](#_Toc176273591)

[3.3 Interface Object - SysFeat 5](#_Toc176273592)

[3.3.1 Public Methods and Properties 5](#_Toc176273593)

[3.4 Interface Object - User 6](#_Toc176273594)

[3.4.1 Public Methods 6](#_Toc176273595)

[3.5 Interface Object - Role 6](#_Toc176273596)

[3.5.1 Public Methods and Properties 6](#_Toc176273597)

[3.6 Interface Object - DevDef 6](#_Toc176273598)

[3.6.1 Public Methods and Properties 6](#_Toc176273599)

[3.7 Interface Object – Dev 7](#_Toc176273600)

[3.7.1 Public Methods and Properties 7](#_Toc176273601)

[3.8 Interface Object - Zone 8](#_Toc176273602)

[3.8.1 Public Methods and Properties 8](#_Toc176273603)

# Development

## Folders

There is one folder containing the skeleton solution and the documentation…

* SecMan

This has three sub folders…

* SecMan.Data – This is the SecMan data layer project and is included in the SecMan solution.
* SecMan.Data.Test – This is a test program for SecMan.Data and is included in the SecMan solution.
* SecMan.Db – This contains the database SecMan.db and the json files used to initialise it.

## Debug and test SecMan.Data

1. Open the SecMan solution using Visual Studio 2022.
2. Run SecMan.Data.Test in debug.

## Database Initialisation

A newly installed empty SecMan.db will be initialised when it is first loaded using the json files in the SecManDb folder.

* Device Definitions – Policies, permissions, and signature definitions for each device type…
* Secman.json
* EPM.json
* Reviewer.json
* ERPM.json
* System Features – Defines the system features and their properties…
* PasswordComplexity.json
* PasswordManagement.json
* UserAccountManagement.json
* Email.json
* SysLog.json
* Default Configuration – Defines an initial configuration Users, Roles, Devices and Zones…
* DefaultConfiguration

The Device Definitions are fixed and cannot be edited, however the SecMan.Data interface will modify…

* System Feature property values (e.g. Email server port)
* Default Configuration (e.g. User, Roles, Permissions)

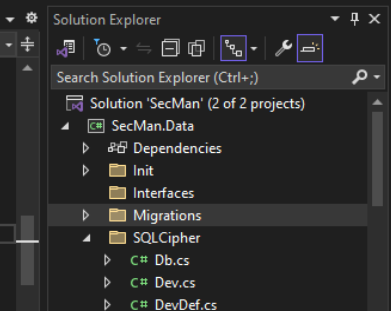
## Database Modifications

The structures are defined by the classes in the folder SQLCipher. If it is necessary to modify these classes, for example to add a new property, then the database must be recreated…

1. Delete the database file…

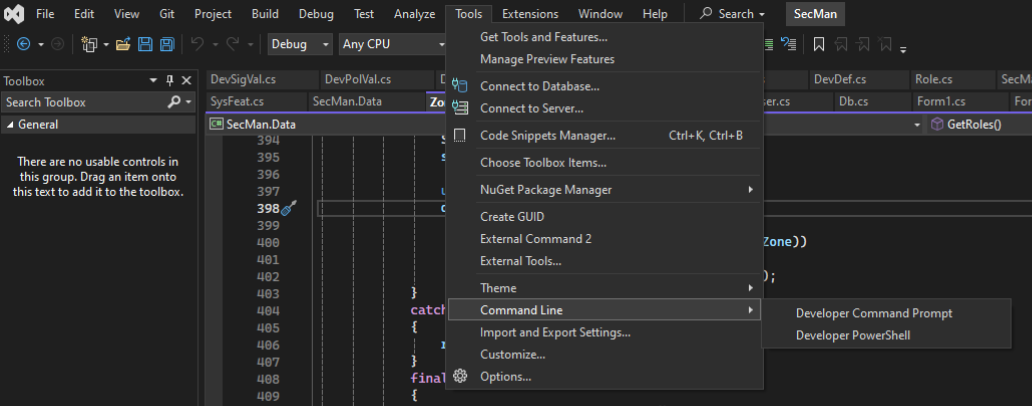
…\SecMan\SecMan.Db\SecMan.db

1. Open the SecMan solution in Visual Studio 2022
2. Delete the Migrations folder from the SecMan.Data project Delete the database file…



…\SecMan\SecMan.Db\SecMan.db

1. From Visual Studio open a Developer Command Prompt…



1. From command prompt execute

> cd SecMan.Data

> dotnet ef migrations add InitialCreate

> dotnet ef database update

1. This will recreate the Migrations in SecMan.Data and recreate the SecMan.db which will be reinitialised the next time it is loaded.

# Database

The database is called SecManDb.db

Note: For initial development it is an SQLite database, however in the final release it will be an encrypted SQLCipher database.

# Interface objects

The primary interface object, SecManDb, is instantiated in the SecMan business layer and used for all configurations. This object and all secondary interface objects exist in the namespace…

* namespace SecMan.Data

## Interface Object - SecManDb

This is the primary object used to access all data. It is instantiated in the SecMan business layer…

SecManDb secManDb = new();

### Constructor

On a clean installation the tables in the database are empty. The constructor tests for this and uses the json files in the SecManDb folder to initialise the database.

### Public Methods – SuperUser

* public SuperUser? GetSuperUser()

Get the Super User

### Public Methods – User

* List<User> GetUsers()

Get all users.

* User? GetUser(ulong userId)
* User? GetUser(string userName)
* User? GetUser(string domain, string userName)

Get a specific user.

* User? AddUser(string userName, string password)
* bool DelUser(ulong userId)

Create and delete users.

### Public Methods – Role

* List<Role> GetRoles()
* Role? AddRole(string name)
* bool DelRole(ulong roleId)

List, add and delete roles.

### Public Methods - Device Definition Methods

* List<DevDef> GetDevDefs(string langCode)

List all device definitions in the selected language.

### Public Methods – Device

* List<Dev> GetDevs()

List all devices.

* Dev? GetDev(ulong devId)

Get a specific device.

* Dev? AddDev(string devDefName, string devName)
* ReturnCode DelDev(ulong devId)

Add and delete devices

### Public Methods – Zone

* List<Zone> GetZones()
* Zone? AddZone(string name)
* bool DelZone(ulong zoneId)

List, add and delete zones.

### Public Methods - System Features

* List<SysFeat> GetSysFeats(string langCode)

List all system features and their properties.

### Public Methods – Values

A set of methods used to set values for

* System Features
* Device Policies
* Device Signatures
* User Permissions.

Each value is addressed by Id…

* bool SetSysFeatPropVal(ulong sysFeatPropId, string val)

This method is used to set the System Feature property values returned from SecManDb: GetSysFeats.

* ReturnCode SetDevPolVal(ulong devPolValId, string val)

This method is to set the values for the Device Policies returned from Zone: GetDevDefPolVals.

* ReturnCode SetSigVal(ulong sigValId, bool sign, bool auth, bool note)

This method is to set the values for the Device Signatures returned from Zone: GetDevDefSigVals.

* ReturnCode SetPermVal(ulong permValId, bool val)

This method is to set the values for the User Permissions returned from Zone: GetDevDefRolePerms.

## Interface Object – SuperUser

### Public Methods and Properties

* Get and Set the Super User’s username and password.

## Interface Object - SysFeat

### Public Methods and Properties

* Name
* Common
* If true, this feature may be edited in EPM-S setting menu. This requires SecMan CAN\_EDIT\_COMMON\_SYSTEM\_FEATURES permission,
* If False, it can only be edited within SecMan and requires CAN\_EDIT\_SYSTEM\_FEATURES
* List the System Features properties.

### 

## Interface Object - User

### Public Methods

* Get and Set

Read write all user account properties (e.g. Email, Password, etc...)

* List<Role> Roles
* bool AddRole(ulong roleId)
* bool RemRole(ulong roleId)

List, allocate and deallocate the user’s roles.

## Interface Object - Role

### Public Methods and Properties

* Get and Set.

Read write Role t properties

* List<User>? Users
* bool AddUser(ulong userId)
* bool RemUser(ulong userId)

List, allocate and deallocate users to the role.

* List<Zone>? Zones
* bool AddZone(ulong zoneId)
* RemZone(ulong zoneId)

List, allocate and deallocate the role’s zone.

### 

## Interface Object - DevDef

This object defines a device type. It is not configurable and is set by the json initialization file when the database is loaded. If a later version of a device (e.g. EPM) is released with new permissions or policies then there will be a later version of its json initialization file installed, and the device definition in the database will be automatically updated.

### Public Methods and Properties

* string Name

Name of the Device Type (e.g. SecMan, EPM etc…)

* bool App

Flag to indicate the device type is an application. App devices are singletons and their logon is managed by the EPM Suite SSO (Single Sign On)

* ulong Vers

The version of the device definition. If a later version is available in the json initialization files it will automatically be updated in the database when the database is loaded.

* List<DevPolDef>? DevPolDefs

List all the device’s policies.

* List<DevPermDef>? DevPermDef

List all the device’s user permissions.

### 

## Interface Object – Dev

### Public Methods and Properties

* string? Name

Get and set the device’s name.

* DevDef DevDef

Get the device’s type definition.

* public string? Vers

The version of the instrument firmware

* bool Legacy

Flag to indicate this is either a legacy device or it is configured to allow legacy users.

* ulong SysPolVer
* ulong DevPolVer
* ulong DevPermVer
* ulong UserVer
* ulong RoleVer

Deployed versions

* ConnRate
* int ConnState
* DateTime LastConnDate

Device connection status

* Zone? Zone

Allocated zone

* AddZone(ulong zoneId)
* RemZone()

Allocate and deallocate the device to a zone.

## Interface Object - Zone

### Public Methods and Properties

* string? Name

Get and set the zone name.

* List<Dev> Devs
* AddDev(ulong devId)
* bool RemDev(ulong devId)

Allocate and deallocate devices to the zone.

* List<Role> Roles
* bool AddRole(ulong roleId)
* RemRole(ulong roleId)

Allocate and deallocate a role to the zone.

* List<DevDef> GetDevDefs(string lang)

List the unique Device Definitions allocated to the zone

* List<DevPolVal> GetDevDefPolVals(ulong devDefId)
* List<DevSigVal> GetDevDefSigVals(ulong devDefId)

Get the Device Policy and Device Signature configuration for a Device Type Definition allocated to the zone.

* List<DevPermVal> GetDevDefRolePerms(ulong devDefId, ulong roleId)

Get the User Permissions configuration for a Role on a Device Type Definition allocated to the Zone.