Описание структуры управляющего программного обеспечения КИП СПП

# Общее описание

Управляющее программное обеспечение для комплексов измерения параметров силовых полупроводниковых приборов (КИП СПП) предназначено для централизованного управления набором измерительных и вспомогательных блоков, из которых состоит комплекс, реализации задач взаимодействия с пользователем и выдачи необходимых данных на внешние интерфейсы.

Данное программное обеспечение реализовано на базе персонального компьютера с ОС Windows с использованием платформы .NET 4.0 и частично .NET 4.6.2. ПО представляет собой взаимосвязанный набор модулей, который может быть запущен на базе одного или нескольких блоков (вариант с выделенным блоком пользовательского интерфейса).

Оглавление

[Общее описание 1](#_Toc477119215)

[Структура программного обеспечения 2](#_Toc477119216)

[Логика работы в рамках комплекса 4](#_Toc477119217)

[Логика работы в рамках сети предприятия 5](#_Toc477119218)

[Централизованное управление профилями измерений 7](#_Toc477119219)

[Синхронизация результатов измерений 8](#_Toc477119220)

[Приложения для доступа к результатам БД и генерации отчетов 9](#_Toc477119221)

[Центральный сервер и БД 11](#_Toc477119222)

[Приложение для управления профилями измерений 12](#_Toc477119223)

[Приложения сервиса 13](#_Toc477119224)

[Настройки модуля SCME.Service 14](#_Toc477119225)

[Настройки модуля SCME.UI 16](#_Toc477119226)

[Настройки модуля SCME.Agent 17](#_Toc477119227)

[Настройки модуля DatabaseServer 17](#_Toc477119228)

[Настройки модуля SqlDatabaseClient 18](#_Toc477119229)

[Архитектура программного обеспечения комплекса 18](#_Toc477119230)

[Библиотека типов SCME.Types 18](#_Toc477119231)

[Библиотека типов SCME.InterfaceImplementation 23](#_Toc477119232)

[Приложение SCME.Service 26](#_Toc477119233)

[Классы для управления оборудованием 28](#_Toc477119234)

[Приложение SCME.UI 29](#_Toc477119235)

[Классы взаимодействия через WCF 30](#_Toc477119236)

[Классы визуального интерфейса 31](#_Toc477119237)

[Классы специализированных элементов управления 32](#_Toc477119238)

[Вспомогательные классы 33](#_Toc477119239)

[Архитектура инфраструктурного программного обеспечения 34](#_Toc477119240)

[Приложение SQLDatabaseClient 35](#_Toc477119241)

[Приложение DatabaseServer 39](#_Toc477119242)

[Архитектура архивного программного обеспечения 40](#_Toc477119243)

[Приложение SCME.NetworkPrinting 40](#_Toc477119244)

[Структура БД результатов измерений 41](#_Toc477119245)

[Дальнейшее усовершенствование системы 43](#_Toc477119246)

[Доработка системы профилей 43](#_Toc477119247)

[Доработка системы ввода данных 44](#_Toc477119248)

[Доработка системы отчетов 44](#_Toc477119249)

[Доработка системы синхронизации 44](#_Toc477119250)

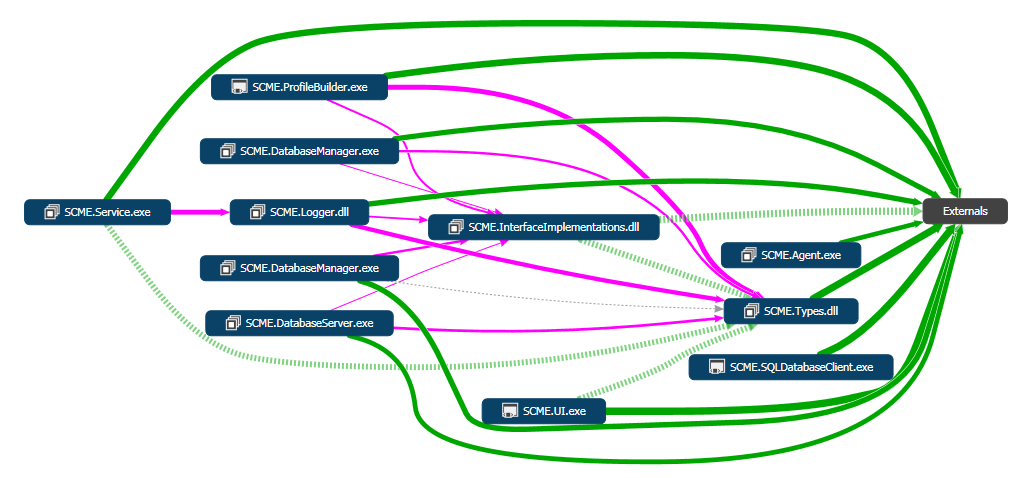
# Структура программного обеспечения

Программное обеспечение реализовано на базе платформы .NET и написано в среде MS Visual Studio с использованием C#. Большинство проектов, входящих в структуру ПО объединены в одно решение SCME.

* **SCME.Types** – библиотека общих типов данных и интерфейсов, используемых совместно различными проектами;
* **SCME.InterfaceImplementations** – библиотека с реализацией классов, обеспечивающих функциональность клиент-серверных взаимодействий (сохранение результатов, загрузка и сохранение профилей) для БД SQlite и MS SQL Server;
* **SCME.Logger** – библиотека, предоставляющая функциональность журнала событий для различных приложений;
* **SCME.Service** – основное управляющее приложение, которое содержит в себе логику инициализации блоков комплекса, проведения измерений и сохранения результатов;
* **SCME.UI** – приложение, обеспечивающее пользовательский интерфейс комплекса, функциональность профилей измерения и просмотра результатов;
* **SCME.DatabaseManager** – приложение инициализации и импорта данных для БД на основе движка SQLite;
* **SCME.SQLDatabaseManager** – приложение инициализации БД на основе MS SQL Server и импорта данных из БД SQLite;
* **SCME.DatabaseServer** – серверная служба, обеспечивающая синхронизацию профилей и результатов на оборудовании с центральным сервером MS SQL Server (и опционально, SQLite);
* **SCME.ProfileBuilder** – приложение для ПК для редактирования профилей измерения, хранимых в центральной БД и сопоставления их различным экземплярам оборудования;
* **SCME.SQLDatabaseClient** – приложение для ПК для получения результатов измерения их центральной БД и составления отчетов;
* **SCME.Agent** – вспомогательное приложение для запуска и мониторинга состояния процессов SCME.UI и SCME.Service;
* **SCCI\_IO** – внешняя библиотека, реализующая интерфейс обмена данными по последовательному интерфейсу с аппаратурой комплекса;

Также присутствуют архивные проекты:

* **SCME.NetworkPrinting** – приложение, развертываемое на сервере внутренней сети предприятия для реализации функциональности печати результатов измерений;
* **SCME.ExcelPrinting** – библиотека, предоставляющая функции генерации отчетов в формате Excel для SCME.NetworkPrinting;



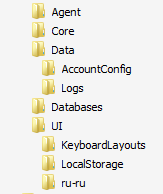
Все перечисленные модули и проекты компилируются для платформы x86 и версии среды исполнения .NET 4.0, кроме **SQLDatabaseClient** (платформа AnyCPU, .NET 4.6.2). Для взаимодействия между удаленными модулями применяются сетевые соединения на основе технологии WCF.

# Логика работы в рамках комплекса

Основным модулем, отвечающим за управление комплексом является SCME.Service. Источником команд для него является клиент, подключающийся к нему по сетевому протоколу и передающий управляющие команды. В случае автономного комплекса с пользовательским интерфейсом роль управляющего клиента реализует SCME.UI, отображающий графический пользовательский интерфейс.

В случае удаленно управляемого комплекса (КИП СПП для АКИМ) приложение SCME.UI не используется и его роль выполняется одним из модулей АКИМ, подключающимся к комплексу удаленно по сетевому интерфейсу.

По умолчанию, запуск SCME.Service и SCME.UI выполняется вспомогательным приложением SCME.Agent, которое следит за их состоянием и перезапускает их в случае непредвиденного сбоя. После запуска, SCME.UI устанавливает сетевое соединение с сервисом и запускает процедуру инициализации. Завершение работы комплекса не предусматривается, достаточно выключить питание комплекса.

 На комплексе программное обеспечение развертывается в следующей структуре:

1. Agent – содержит приложение агента запуска и его конфигурационный файл;
2. Core – содержит приложение SCME.Service;
3. Data -> AccountsConfig – содержит список пользователей комплекса (операторов) и их учетные данные для входа;
4. Data -> Logs – содержит журнал событий в текстовом виде (если данный режим включен);
5. Databases – содержит файлы баз данных журнала событий и результатов измерений;
6. UI – содержит приложение SCME.UI
7. UI -> KeyboardLayouts – содержит варианты раскладок виртуальной клавиатуры;
8. UI -> LocalStorage – содержит файл хранилища параметров конфигурации блоков;

Помимо основного управляющего сетевого соединения, SCME.Service публикует дополнительный интерфейс, предназначенный для запросов к базе данных результатов измерений. Соединение по этому интерфейсу устанавливается постоянным образом приложением SCME.UI и, дополнительно, внешними приложениями при необходимости получения данных. На настоящем этапе данный интерфейс не используется в связи с тем, что отправка данных на центральный сервер осуществляется на основе прямого вызова и синхронизации в момент запуска.

Схема взаимодействия программного обеспечения комплекса представлена на рисунке ниже:

IExternalControl

SCME.Service

Локальная БД

SCME.UI

Профили

Аккаунты

FTDI-адаптер

(MME002)

Интерфейсный адаптер

IDatabaseCommunication

SQLite

RS-232/485

[WCF]

USB

В качестве основы локальной БД применяется технология SQLite, которая обеспечивает следующие преимущества:

* Архитектуру без сервера – движок БД реализован в виде подключаемой библиотеки;
* Поддержку большинства возможностей реляционной модели и SQL;
* Устойчивость к незапланированному выключению питания – полная гарантия сохранения целостности БД;

Последний пункт особенно важен, так как комплекс не предусматривает специальной процедуры выключения, кроме отключения питания. Целостность ОС в данном случае реализуется путем «замораживания» системного раздела диска таким образом, что запуск системы всегда происходит из исходной конфигурации, а любые промежуточные записи кэшируются и не сохраняются в образе тома.

FTDI-адаптер отмечен на схеме светлым цветом, так как является опциональным компонентом, функциональность которого может быть реализована через интерфейсный адаптер в случае если Service и UI исполняются на одном компьютере (совмещенный HMIU). В настоящий момент этот компонент применяется только в комплексе MME-002.

# Логика работы в рамках сети предприятия

Следующие аспекты функционирования комплекса могут быть интегрированы в настоящий момент в сеть предприятия:

1. Централизованное управление профилями измерений – сведения об актуальных профилях измерений запрашиваются из центральной БД (серверного приложения, развернутого в сети) при старте комплекса и кэшируются в локальной БД. В случае недоступности сетевого хранилища используется последняя синхронизированная версия профилей.
2. Синхронизация результатов измерений – полученные результаты измерений сохраняются в центральной БД наряду с их записью в локальную БД. В случае недоступности центральной базы данных результаты будут синхронизированы при перезапуске измерительного комплекса.
3. Приложение для доступа к результатам измерений и построения отчетов – клиент центральной базы данных, устанавливаемый на рабочем ПК и предоставляющий возможности просмотра, ручного ввода и редактирования результатов измерений в центральной БД и построения отчетов.
4. Центральный сервер – приложение, развертываемое на одном из сервером в сети предприятия. Обеспечивает предоставление сервисов из пунктов 1 и 2 («Централизованное управление профилями измерений» и «Синхронизация результатов измерений») для комплексов на предприятии и взаимодействие с центральной БД.
5. Центральная БД – база данных, содержащая в себе информацию о профилях измерений и результатах измерений на всех комплексах в сети предприятия. Развертывается на базе MS SQL Server 2014.
6. Приложение управления профилями измерений – предназначено для редактирования профилей измерений в центральной БД. Обеспечивает возможности создания, удаления и изменения профилей измерений, а также их сопоставления с различными измерительными комплексами.

SCME.Service

Локальная БД

SCME.UI

DATABASE SERVER

*Сервер*

Профили

Центральная БД

Результаты

Синхронизация профилей и результатов

Результаты

SQLDatabase Client

Результаты

ProfileBuilder

Профили

## Централизованное управление профилями измерений

Для синхронизации профилей измерения между различными комплексами и централизованного управления применятся система с центральным сервером, который хранит словарь профилей измерений и его конфигурации для конкретных измерительных комплексов.

При старте измерительный комплекс пробует получить список измерительных профилей с сервера профилей. В случае удачной попытки данный список принимается в работу и сохраняется на комплексе в виде локальной копии. В случае неудачной попытки загрузки с сервера для работы используется локальная копия.

Старт

Установление соединения с DatabaseServer

Успешное соединение

Синхронизация: загрузка списка профилей

Сохранение в виде локальной копии

Использование последней успешной копии

1

0

## Синхронизация результатов измерений

Для централизованного хранения результатов измерений используется центральная БД (та же, в которой хранятся профили измерений). При завершении измерения пробора комплекс пытается отослать информацию на центральный сервер и записывает ее в локальную БД. В случае, если отсылка информации на сервер не удается, то данная запись помечается в локальной БД как незавершенная.

При новом старте комлпекса в момент синхронизации профилей происходит запись в центральную БД всех незавершенных результатов измерений и сброс флагов в записях локальной БД.

Старт

Установление соединения с DatabaseServer

Успешное соединение

Синхронизация:

передача результатов измерений

Сброс флага незавершенности

Измерение

Запись результата на сервер

Успешная запись

Запись копии в локальную БД со сброшенным флагом

Запись копии в локальную БД с флагом незавершенности

1

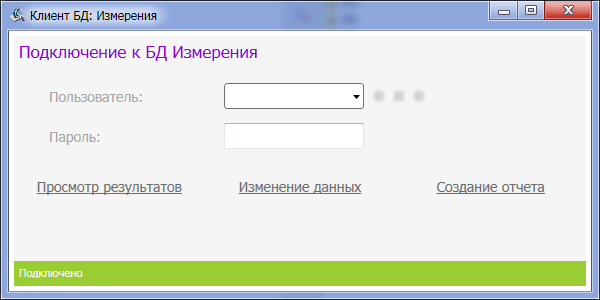
1

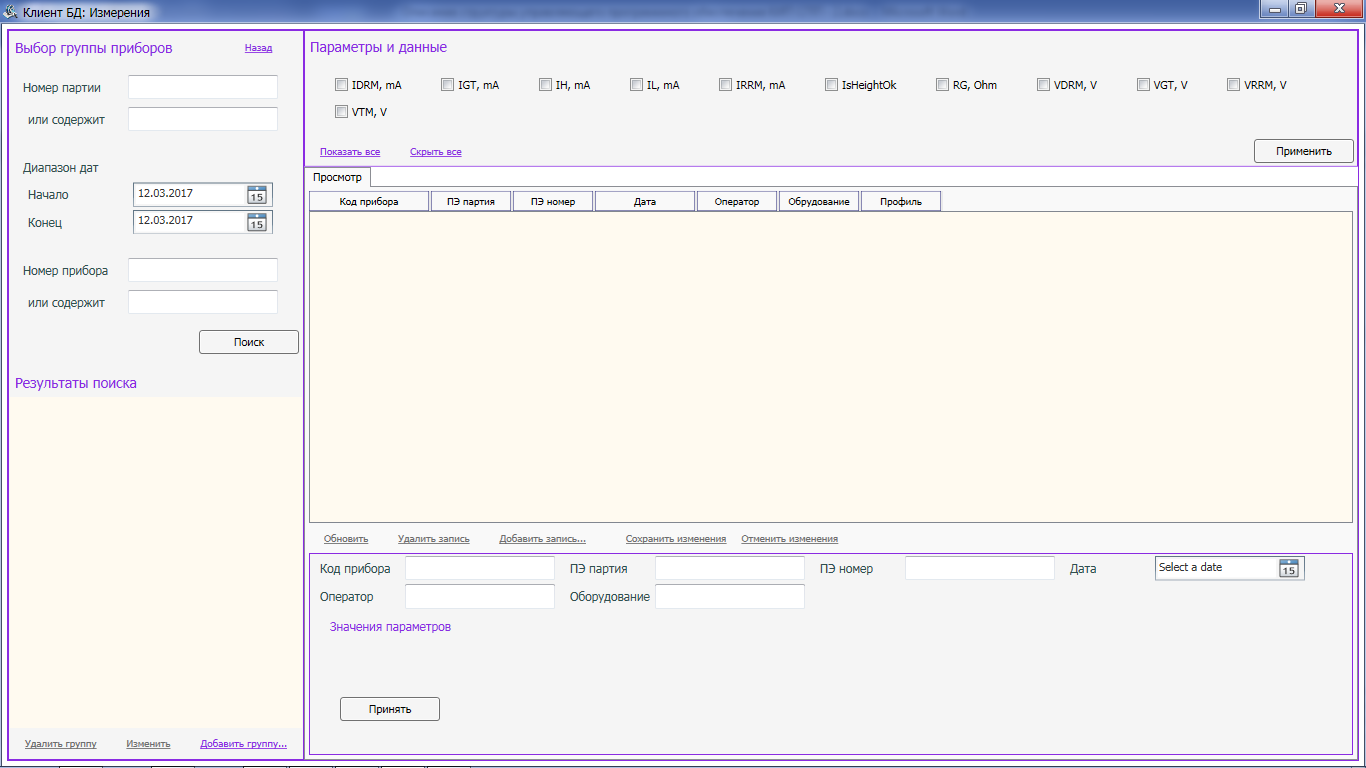
0

## Приложения для доступа к результатам БД и генерации отчетов

Данное приложение устанавливается на рабочем компьютере в сети предприятия и предназначено для осуществления следующих функций:

1. Просмотр результатов измерений, сохраненных в центральной БД:
   1. Поиск групп измеренных приборов по различным критериям;
   2. Выбор отображаемых категорий данных;
2. Ручной ввод и изменение данных:
   1. Добавление/изменение/редактирование групп приборов;
   2. Добавление/изменение/редактирование результатов измерений
   3. Поддержка ввода нескольких результатов измерений для одного параметра;
3. Генерация отчетов:
   1. Создание шаблонов отчетов при помощи FastReport;
   2. Генерация отчетов по данных из БД на основе созданных шаблонов;
4. Прочие функции:
   1. Авторизация пользователей на основе прав доступа
   2. Прямой доступ к БД MS SQL Server.



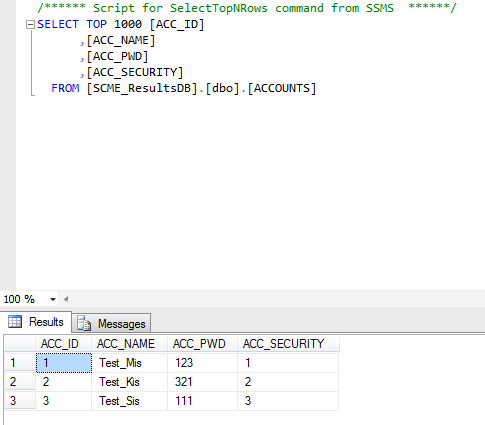


Для задания пользователей и уровней доступа используется таблица ACCOUNTS центральной БД.

Уровни доступа назначаются следующим образом:

1. 1 = права на получение данных;
2. 2 = 1 + права на создание отчетов;
3. 3 = 2 + права на изменение данных;

При увеличении количества различных видов прав доступа разумно рассмотреть переход на схему хранения прав в виде битовой маски, позволяющей упростить хранение различных сочетаний.



В полях задаются следующие значения:

1. ACC\_NAME – имя пользователя;
2. ACC\_SECURITY – уровень доступа, как указано выше;
3. ACC\_PWD – пароль;

## Центральный сервер и БД

Центральная БД реализуется на основе технологии MS SQL Server 2014 и развертывается на одном из серверов в локальной сети предприятия. В БД сохраняется следующие данные:

1. Данные о произведенных результатах измерений;
2. Данные о профилях измерений;
3. Данные о зарегистрированных комплексах;
4. Данные о сопоставлении профилей измерений и комплексов;
5. Данные об уровнях доступа пользователей через клиентское приложение;

Для доступа к БД для измерительных комплексов существует серверная «прослойка» в виде приложения DatabaseServer. Данное приложение может быть развернуто как на одном, так и на разных серверах с ЦБД в сети предприятия. Режимы запуска: сессия интерактивного пользователя и режим работы в качестве системного сервиса.

DatabaseServer

MS SQL Server

MME001

MME00X

…

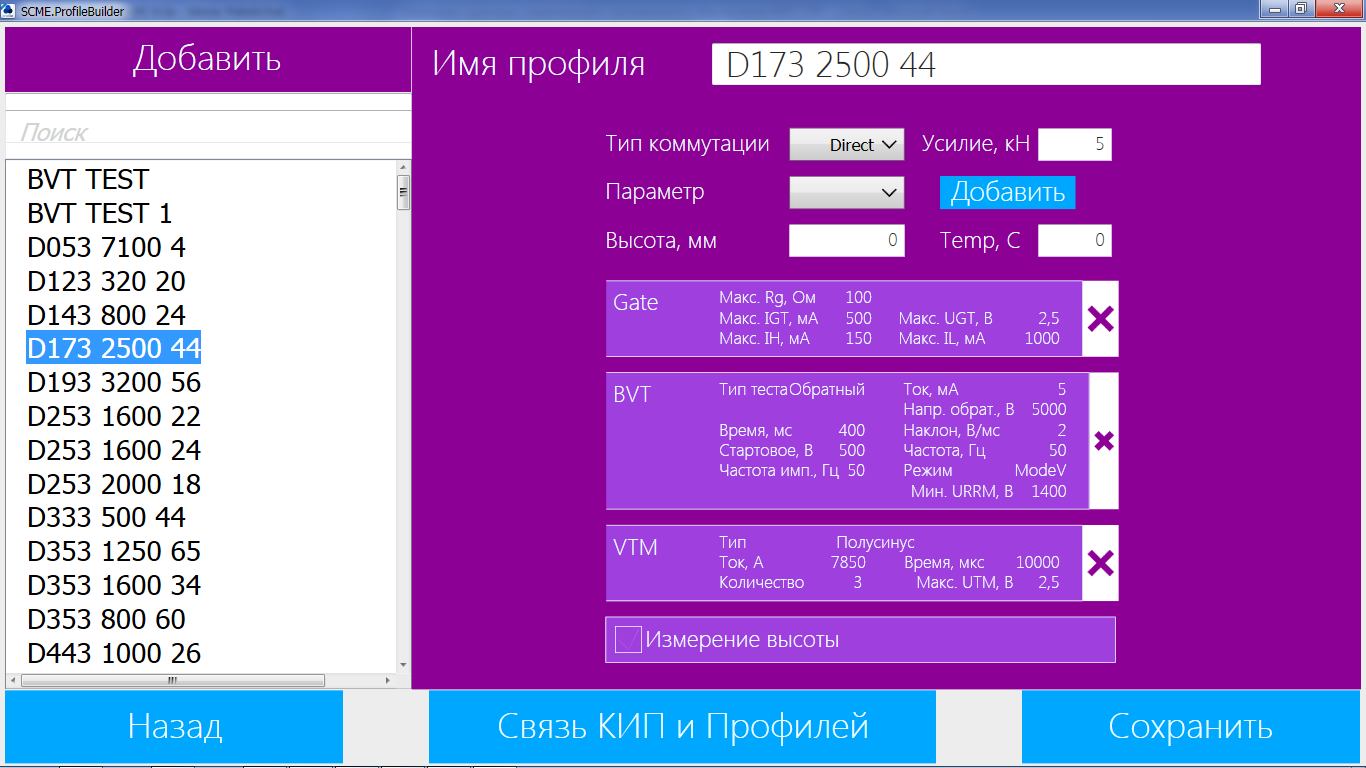
результаты

профили

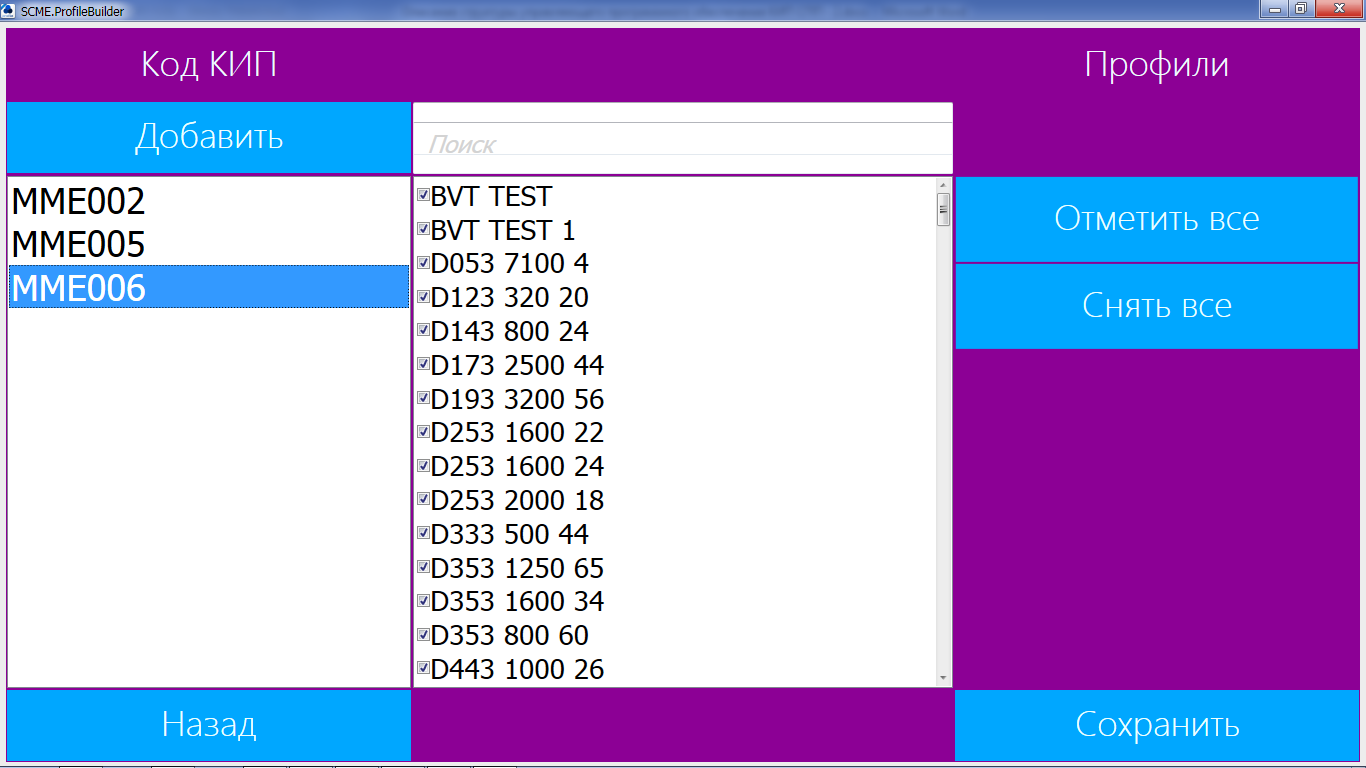
## Приложение для управления профилями измерений

ProfileBuilder предназначен для реализации функциональности создания/изменения/удаления профилей измерения и их сопоставления конкретных комплексам.

Редактирование параметров профиля осуществляется при помощи графического интерфейса, представляющего параметры профиля в структурированном виде, а также предыдущие версии данного профиля. Примечание – при удалении профили не удаляются физически из БД, а помещаются флагом как удаленные; при изменении профиля создается его новая версия, а предыдущие версии помечаются как архивные и доступны только для чтения.

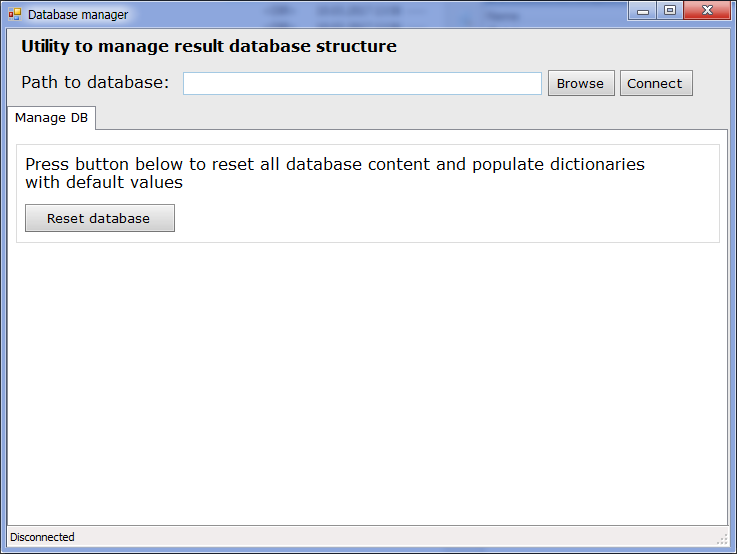


Сопоставление комплексов и профилей осуществляется при помощи отметки активных профилей из числа активных и неудаленных в списке для каждого конкретного комплекса.

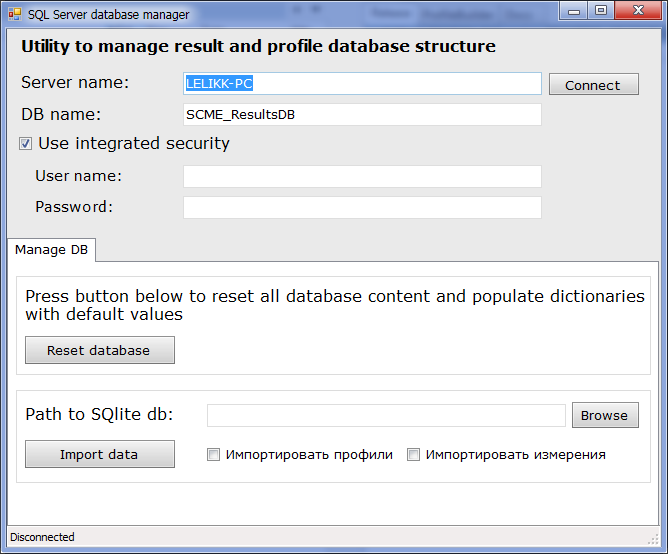


## Приложения сервиса

К сервисным приложениям относятся приложения для инициализации структуры БД и переноса данных.



Приложение DatabaseManager предназначено для инициализации в исходное состояние БД формата SQLite (локальная БД комплекса). После выбора БД и подсоединения к ней кнопка **Reset Database** очищает БД и приводит ее структуру в исходное состояние.



Приложение SQLDatabaseManager предназначено для очищения БД MS SQL Server и импорта в нее данных из БД формата SQLite. После выбора БД и подсоединения к ней кнопка **Reset Database** очищает БД и приводит ее структуру в исходное состояние.

Импорт профилей и результатов измерения можно осуществить из БД SQlite путем нажатия кнопки **Import** **Data**, предварительно выберя исходную БД.

# Настройки модуля SCME.Service

Как описано выше, данное приложение обеспечивает управление всеми блоками измерительного комплекса. Коммуникация между блоками осуществляется посредством шины CAN, обеспечивающей гальванически изолированную отказоустойчивую среду обмена данными. Доступ управляющего компьютера к этой шине производится через промежуточный интерфейсный контроллер, подсоединенный к компьютеру по последовательному интерфейсу. Протокол обмена данными с интерфейсным контроллером специализированный, базирующийся на протоколе MODBUS, и обеспечивается библиотекой SCCI\_IO.

Для внешней конфигурации (SCME.Service.exe.config) доступны следующие настройки приложения:

* GateNode – адрес блока измерения характеристик управления (Gate) на шине CAN;
* SLNode – адрес блока измерения статических потерь (SL) на шине CAN;
* BVTNode – адрес блока измерения блокирующих характеристик (BVT) на шине CAN;
* dVdtNode – адрес блока измерения критической скорости нарастания напряжения (dVdt) на шина CAN;
* CommutationNode – адрес блока коммутации (Commutation) на шине CAN;
* AdapterPort – адрес COM-порта для соединения с интерфейсным контроллером;
* GatewayNode – адрес интерфейсного контроллера (Gateway) на шине CAN;
* IsCommutationType6 – флаг, задающий 2-х или 6-ти позиционный коммутационный блок используется;
* CommutationExNode – адрес дополнительного блока коммутации (CommutationEx) на шине CAN;
* CommutationExEmulation – флаг эмуляции блока CommutationEx;
* IsCommutationExType6 – флаг, задающий 2-х или 6-ти позиционный дополнительный коммутационный блок используется;
* ClampingSystemEmulation – флаг эмуляции пресса;
* ClampingSystemNode – адрес пресса (ClampingSystem) на шине CAN;
* IsClampingSystemConnected – флаг, задающий использование пресса в данном комплексе;
* DBOptionsResults – опции подключения к БД результатов;
* DBOptionsLogs – опции подключения к БД журнала событий;
* IncludeDetailsInLog – задает детализацию ведения журнала событий;
* ForceLogFlush – записывать в журнал события не используя кэш;
* GateReadGraph – читать осциллограммы с модуля Gate;
* SLReadGraph – читать осциллограммы с модуля SL;
* BVTReadGraph – читать осциллограммы с модуля BVT;
* DisableResultDB – выключить запись в БД результатов;
* DisableLogDB – выключить запись в БД событий;
* MMECode – идентификатор комлпекса;
* CommutationEmulation – эмуляция блока Commutation;
* GatewayEmulation – эмуляция блока Gateway;
* dVdtEmulation – эмуляция блока dVdt;
* AdapterEmulation – эмуляция подключения к интерфейсному адаптеру;
* LogsDatabasePath – путь к БД журнала событий;
* ResultsDatabasePath – путь к БД журнала результатов;
* BVTEmulation – эмуляция блока BVT;
* GateEmulation – эмуляция блока Gate;
* SLEmulation – эмуляция блока SL;
* LogsTracePathTemplate – путь к файлу текстового журнала событий;
* DBServiceExternalEndpoint – точка подключения внешнего интерфейса запроса данных;
* ReadOnlyProfiles – профили на комплексе находятся в режиме синхронизации в сети;

# Настройки модуля SCME.UI

Данный модуль осуществляет реализацию пользовательского интерфейса и управление приложением SCME.Service по протоколу WCF.

Для внешней конфигурации (SCME.UI.exe.config) доступны следующие настройки приложения:

* GateIsVisible – блок Gate включен в работу;
* SLIsVisible – блок SL включен в работу;
* BvtIsVisible – блок BVT включен в работу;
* dVdtIsVisible – блок dVdt включен в работу;
* CommIsVisible – блок Commutation включен в работу;
* ExitButtonIsVisible – отображается кнопка закрытия приложения UI;
* IsAnimationEnabled – включена анимация элементов управления;
* TechPassword – пароль для входа в режим наладчика;
* Localization – текущая локализация приложения;
* PlotUserVTM – отрисовывать график результатов VTM в режиме измерений оператором;
* PlotUserGate – отрисовывать график результатов Gate в режиме измерений оператором;
* PlotUserBVT – отрисовывать график результатов BVT в режиме измерений оператором;
* IsTechPasswordEnabled – запрос пароля при входе в режим наладчика;
* KeyboardsPath – путь к файлу с раскладками клавиатур;
* BVTVoltageFrequency – частота напряжения для блока BVT;
* SinglePositionModuleMode – режим работы с однопозиционными модулями;
* FTDIPresent – присутствует плата расширения на базе чипа FTDI;
* CommExIsVisible – блок CommutationEx включен в работу;
* IgnoreSC – игнорировать цепь безопасности;
* RunExplorer – запускать Explorer при закрытии приложения;
* ProfilesPath – путь к файлу профилей;
* AccountsPath – путь к файлу с учетными записями операторов;
* FTDIIsInUse – плата расширения FTDI используется в работе (обработка светодиодов и кнопок START-STOP на КИП СПП-002);
* ClampIsVisible – блок ClampingSystem включен в работу;
* NormalWindow – отображать в нормальном режиме вместо полноэкранного;
* UseVTMPostProcessing – использовать постобработку результатов VTM;
* StoragePath – путь к файлу локального хранилища параметров;
* UseLocalProfiles – использовать данные профилей измерений из локального файла, а не с сервера профилей;
* MMECode – идентификатор комплекса;

# Настройки модуля SCME.Agent

Данный модуль осуществляет запуск и перезапуск в случае аварии компонентов Service и UI.

Настройки модуля:

* ServiceAppPath – путь к приложению Service;
* UIAppPath – путь к приложению UI;
* IsUserInterfaceEnabled – включено ли слежение за приложением UI;
* ProxyAppPath – путь к дополнительному модулю (опционально);
* IsProxyEnabled – включено ли слежение за дополнительным приложением;
* ProxyInitDelayMs – задержка после запуска дополнительного приложения;

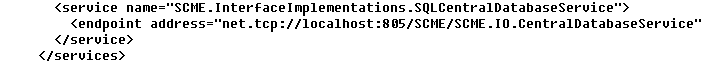
# Настройки модуля DatabaseServer

Данное приложение предоставляет доступ к центральной БД дял измерительных комплексов.

Настройки модуля:

* DBIntegratedSecurity – тип аутентификации на MS SQL SERVER;
* DbPath – адрес сервера;
* DBName – имя БД;
* DBUser – пользователь (в случае DBIntegratedSecurity = False);
* DBPassword – пароль (в случае DBIntegratedSecurity = False);
* LogPathTemplate – путь к файлам лога;

Для задания адреса вещания сервиса используется следующая строка:



Адрес задается в параметре address, между элементами **net.tcp://** и портом **:805**.

# Настройки модуля SqlDatabaseClient

Клиент центральной базы данных, устанавливаемый на рабочем ПК и предоставляющий возможности просмотра, ручного ввода и редактирования результатов измерений в центральной БД и построения отчетов.

Настройки модуля:

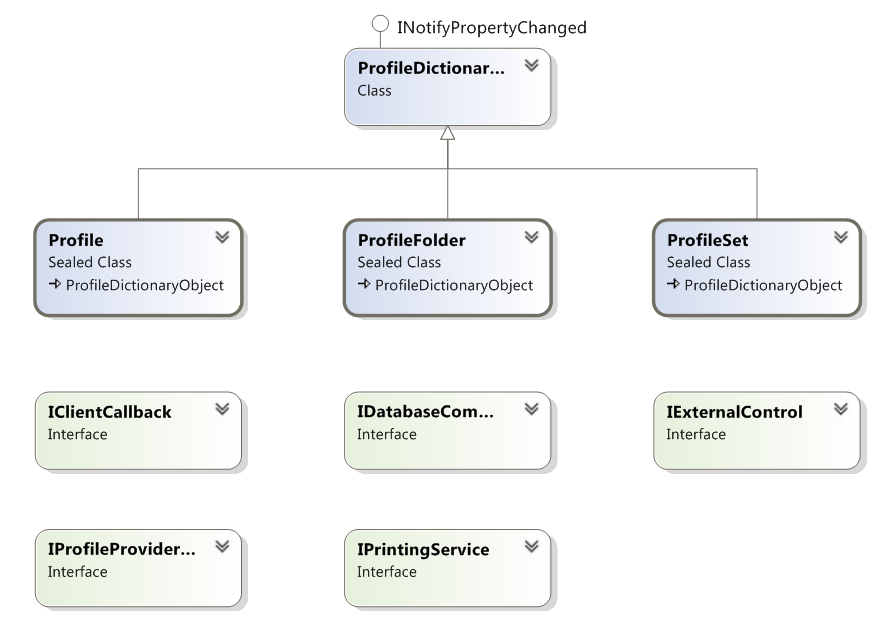
* DBIntegratedSecurity – тип аутентификации на MS SQL SERVER;
* DbPath – адрес сервера;
* DBName – имя БД;
* DBUser – пользователь (в случае DBIntegratedSecurity = False);
* DBPassword – пароль (в случае DBIntegratedSecurity = False);
* ReportTemplateFolder – путь к файлам шаблонов отчетов;

# Архитектура программного обеспечения комплекса

## Библиотека типов SCME.Types

Помимо большого количества классов, описывающих технические детали обмена информацией между приложением интерфейса (UI) и сервисом (Service), в данной библиотеке присутствуют следующие важные типы данных:

* IExternalControl – задает интерфейс управления приложением Service;
* IDatabaseCommunication – задает интерфейс запроса данных из БД измерений;
* IClientCallback – задает интерфейс оповещения управляющего приложения об изменении состояния Service;
* IPrintingService – интерфейс запроса удаленной печати;
* IProfileProvider – интерфейс запроса списка профилей измерений с сервера профилей;
* ProfileDictionaryObject – абстрактный класс, предоставляет общую функциональность для элементов словаря профилей измерений;
* Profile – профиль измерений, содержащий информацию о проводимых измерениях, условиях проведения и нормативов на полученные значения.
* ProfileSet – элемент словаря профилей, объединяющий историю изменения параметров профиля;
* ProfileFolder – элемент словаря профилей, объединяющий профили по произвольным категориям;



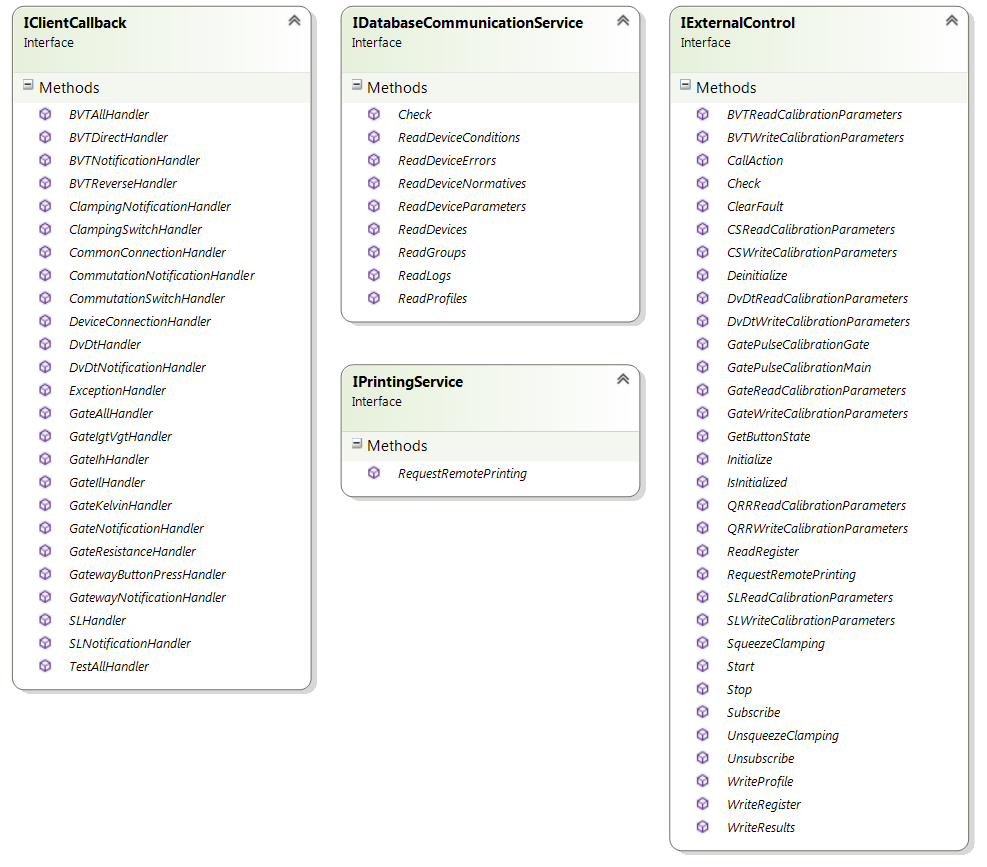
Интерфейс **IExternalControl** содержит список методов, условно группируемых по своему отношению к различным измерительным блокам. Это относится к методам, начинающимся с префиксов SL, BVT, Gate, CS, QRR и пр. Существуют также методы, управляющие общей работой измерительного комплекса: *Initialize*, *Start*, *Stop*. Результаты измерений записываются при помощи методов *WriteProfile*, *WriteResults*.

Интерфейс **IDatabaseCommunication** предоставляет возможность запроса данных о результатах измерений с помощью методов:

* *ReadGroup* – список партий за некоторый период;
* *ReadDevices* – список приборов в заданной партии;
* *ReadDeviceParameters* – результаты измерения прибора;
* *RadDevuceNormatives* – нормативы на результаты измерения;
* *ReadDeviceConditions* – условия измерения приборов;
* *ReadDeviceErrors* – ошибки измерения прибора;

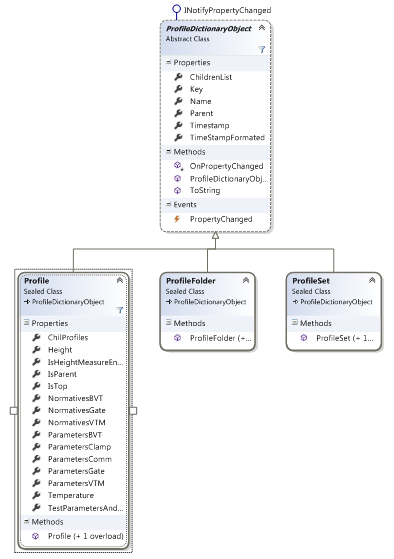
Интерфейс **IClientCallback** используется для функций обратного вызова в паре с интерфейсом IExternalControl. С его помощью Service уведомляет ведущее приложения (UI) об изменениях состояния. Методы также можно сгруппировать по именам, отнеся их к различным измерительным блокам.

Интерфейс **IPrintingService** содержит единственный метод *RequestRemotePrinting*, который передает всю необходимую информацию: название партии, идентификатор комплекса, настройки печати.



Структура семейства классов Profile ориентирована на иерархическую структуру хранения. Базовый класс предоставляет свойства для хранения имени элемента, его GUID, времени создания (необходимо для версионирования профилей с помощью ProfileSet), ссылки на родителя дерева и коллекции дочерних элементов.

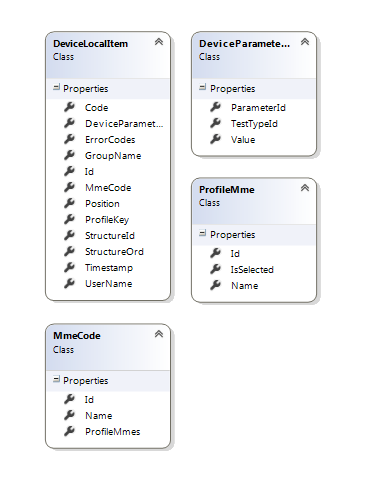
Класс **Profile** содержит поля для хранения *Normatives{}*, *Parameters{}* для различных блоков.



Классы DeviceLocalItem, DeviceParameterItem, MmeCode и ProfileMme предназначены для обеспечения функциональности сохранения профилей и сопоставления их с конкретными комплексами.

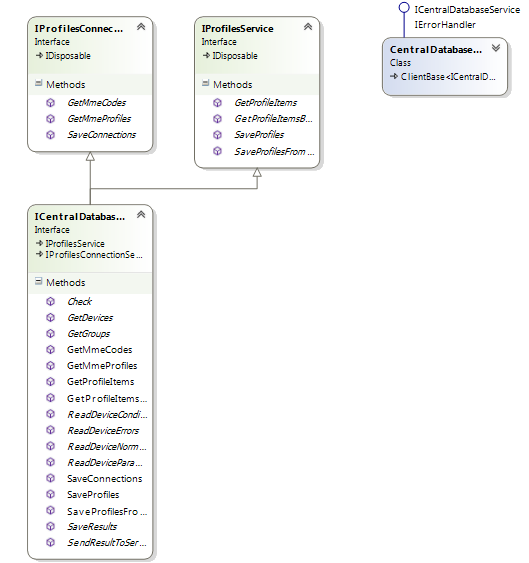
Наиболее важные поля:

1. MmeCode -> ProfileMmes – содержит коллекцию профилей, сопоставленных конкретному комплексу;
2. DeviceParameterItem -> ParameterId – содержит идентификатор параметра их БД;
3. DeviceParameterItem -> ParameterId – содержит идентификатор типа измерения из БД (Gate, SL etc.);
4. DeviceParameterItem -> Value – значение параметра;

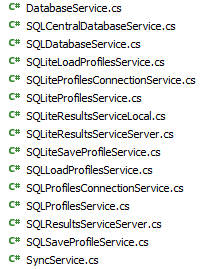


Для работы приложения DatabaseServer, предоставляющего функции получению профилей и их редактирования публикуются следующие интерфейсы:

1. IProfilesConnectionService – предоставляет возможности получения кодов оборудования и связей:
   1. GetMmeCodes – получение зарегистрированных комплексов;
   2. GetMmeProfiles – получение профилей, сопоставленных комплексу;
   3. SaveConnections – сохранение сопоставления профилей и комплексов;
2. IProfilesService – позволяет изменять профили в центральной БД:
   1. GetProfileItems – получение списка профилей;
   2. GetProfileItemsByMme – получение списка профилей дял конкретного комплекса;
   3. SaveProfiles – сохранение списка профилей;
3. ICentralDatabaseService – реализует вышеописанные сервисы и дополнительные методы:
   1. SaveResults – запись результатов в центральную БД;
   2. SendResultToServer – синхронизация локальной и центральной БД результатов;



## Библиотека типов SCME.InterfaceImplementation

Данная библиотека содержит классы, реализующие функциональность доступа к профилям в локальной и центральной БД и записи результатов. Классы условно разделяются на 2 группы: Sqlite\* - классы доступа к БД Sqlite, SQL\* - классы доступа к БД SQL Server.

Классы доступа к БД Sqlite:

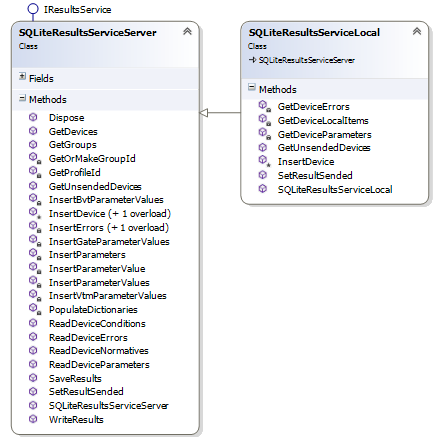
1. SQLiteLoadProfileService – загрузка профилей;
2. SQLiteSaveProfile – сохранение профилей;
3. SQLiteProfileService – общие операции с профилями;
4. SQLiteProfileConnectionService – сопоставление профилей и комплексов;
5. SQLiteResultServiceLocal – сохранение результатов в локальную БД;
6. SQLiteResultServiceServer – сохранение результатов в серверную БД формата Sqlite (не используется в настоящий момент);
7. DatabaseService – сервисные операции с БД;

Классы доступа к БД SQL Server:

1. SQLLoadProfileService – загрузка профилей;
2. SQLSaveProfile – сохранение профилей;
3. SQLProfileService – общие операции с профилями;
4. SQLProfileConnectionService – сопоставление профилей и комплексов;
5. SQLiteResultServiceServer – сохранение результатов в серверную БД;
6. SQLDatabaseService – сервисные операции с БД;

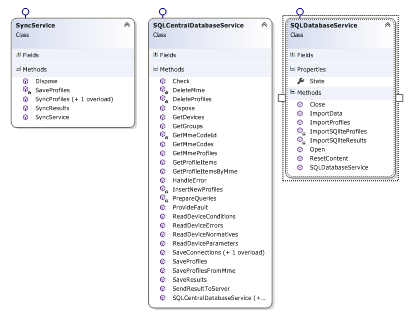
Прочие классы:

1. SQLCentralDatabaseService – объединение функций перечисленных классов для приложения DatabaseServer;
2. SyncService – реализация синхронизации между локальной и центральной БД;



Класс SQLiteResultServiceLocal является потомком класса SQLiteResultServiceServer, реализующим дополнительные функции, такие как:

1. GetUnsendedDevices – получение устройств, для которых не сброшен флаг незавершенности;
2. SetResultSended – сброс флага незавершенности;
3. GetDeviceLocalItems – получение информации из локальной БД для отсылки на сервер;



Методы классы SyncService обеспечивают реализацию следующих функций:

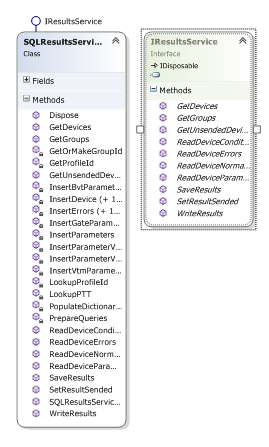
1. SyncResults – синхронизация незавершенных результатов из локальной БД с сервером;
2. SyncProfiles – получение актуальных профилей измерения из центральной БД и запись их в локальную для последующего использования;

Класс SQLCentralDatabaseService агрегирует методы, связанные с загрузкой, сохранением и сопоставлением путем реализации всех упомянутых интерфейсов:



Реализуемый ICentralDatabaserService является наследником интерфейсов IProfileService и IProfileConnectionService, реализуемых классами \*LoadProfile и \*SaveProfile. Класс SQLProfileService агрегирует данную функциональность и используется внутри SQLCentralDatabaserService.

Класс SQLResultServiceServer реализует интерфейс IResultService в контексте работы с БД SQL Server.

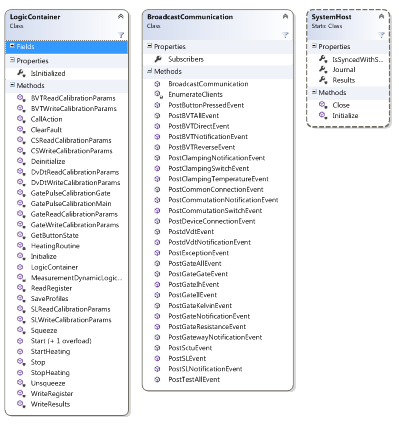


Основным используемым методом в данном интерфейсе является WriteResults, который осуществляет запись результатов в БД. Метод SaveResults предназначен для сохранения результатов из локальной БД в центральную БД. Остальные методы не используются в настоящее время.

## Приложение SCME.Service

Основная функциональность данного приложения содержится в трех группах классов:

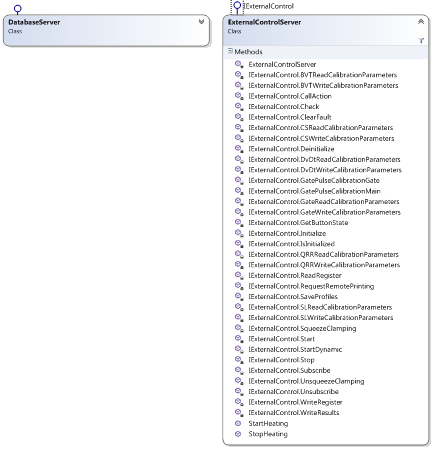
1. Классы основного функционала: LogicContainer, BroadcaseCommunication, SystemHost
2. Классы взаимодействия с управляющим приложением: DatabaseServer, ExternalControlServer
3. Классы взаимодействия с оборудованием: IOAdapter, IOGateway, IOCommutation, IOGate, IOStLs, IOBvt, IOClamping



Статический класс **SystemHost** отвечает за инициализацию журнала событий, базы данных результатов и запуск приложения. После запуска приложения Service, внутреннее управление передается классу LogicContainer.

Класс **LogicContainer** содержит реализацию функциональности IExternalControl, передаваемую через ExternalControlServer (см. ниже). Последовательность процесса измерения и логика реализованы в паре методов Start/Stop.

Класс **BoardcastCommunication** предназначен для трансляции сообщений об изменении состояния Service через интерфейс IClientCallback. Данное сообщение рассылается всем клиентам, подписавшимся на обновления (интерфейс предусматривает возможность получения информации мониторинговыми сервисами).

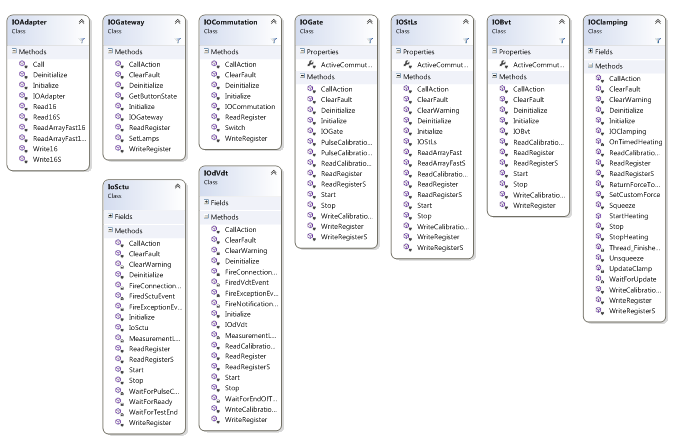


Класс **ExternalControlServer** непосредственно реализует интерфейс IExternalControlServer, выполняя технические действия по обработке запросов, необходимые для инфраструктуры WCF (обработка и упаковка исключений). Логика работы **DatabaseServer** аналогична – функциональность реализуется классом LogicContainer, а этот класс обеспечивает выполнение требований сетевой инфраструктуры, в настоящее время не используется.

### Классы для управления оборудованием

Каждому блоку измерительного комплекса соответствует специальный класс, реализующий необходимые функции через запросы от LogicContainer.

1. IOAdapter – реализует функциональность подключения к интерфейсному адаптеру и обмен данными с блоками комплекса через шину;
2. IOGateway – доступ к интерфейсному адаптеру как самостоятельному устройству и чтение его портов ввода-вывода;
3. IOCommutation – представляет функциональность блока коммутации;
4. IOGate – представляет функциональность блока измерения управления;
5. IOStLs – представляет функциональность блока измерения статических потерь;
6. IOBvt – представляет функциональность блока классификатора;
7. IOClamping – представляет функциональность пресса;
8. IODvDT – функциональность блока критической скорости нарастания напряжения;
9. IOSCTU – функциональность блока ударного тока



## Приложение SCME.UI

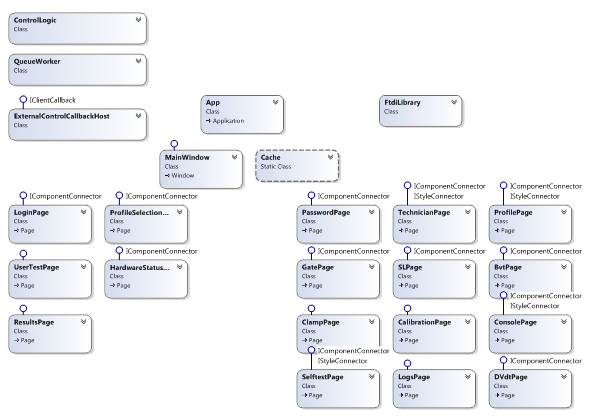
Данное приложение предоставляет графический пользовательский интерфейс для управления комплексом со стороны оператора.

Интерфейс приложения реализован при помощи технологии WPF для обеспечения удобного использования на устройствах с сенсорным экраном. Коммуникация с приложением SCME.Service осуществляется по двум каналам: IExternalControl – для обеспечения функций управления, IDatabaseCommunication – для обеспечения функций получения данных о результатах измерений.

Структура проекта SCME.UI состоит из следующих групп классов:

1. Классы взаимодействия с SCME.Service через механизмы WCF: ControlLogic, ExternalControlCallbackHost, QueueWorker;
2. Классы визуального интерфейса пользователя;
3. Классы специализированных графических элементов управления;
4. Вспомогательные классы;

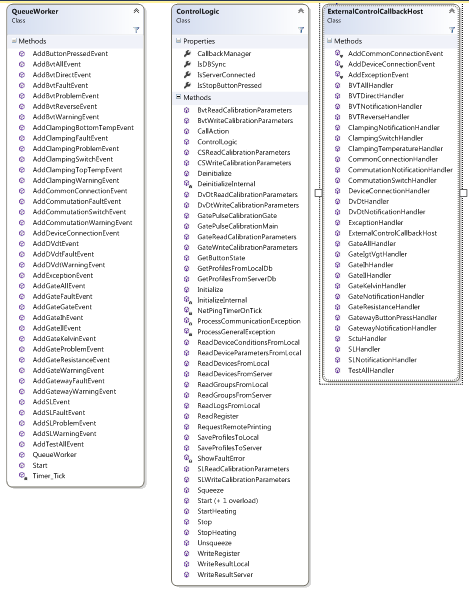
В соответствии с настройками в конфигурационном файле приложение формирует необходимые страницы пользовательского интерфейса, включая их конфигурирование под одно- или двухпозиционный режим измерений.



### Классы взаимодействия через WCF

Функции установления соединения с приложением SCME.Service реализуются классом **ControlLogic**. Данный класс выполняет установление двух соединений с сервисным приложением и проведение процедуры инициализации комплекса. Для получения обновлений состояния сервисного приложения используется интерфейс обратного вызова IClientCallback, который реализуется классом **ExternalControlCallbackHost**. Для трансляции этих вызовов в пользовательский интерфейс применяется класс **QueueWorker**. Данный механизм необходимо для того, чтобы обновления пользовательского интерфейса выполнялись в основном потоке приложения, в то время как обработка сообщений WCF происходит в служебных потоках.

Список методов, реализуемых классом **ControlLogic** в целом совпадает с методами интерфейс IExternalControl. Метод *NetPingTimerOnTick* предназначен для регулярного обновления состояния коммуникационного канала и детектирования обрыва связи. Функции установления связи и запуска процесса инициализации оборудования реализуются в методах *Initialize* и *InitializeInternal*, которые визуализируют состояние этого процесса через обратные вызовы и экран HardwareStatusPage (см. ниже). Функции выключения блоков реализуются методами *Deinitialize* и *DeinitializeInternal*. Метод *ProcessCommunicationException* предназначен для обработки ситуации потери связи с сервисным приложением и проведения процедуры повторной установки соединения с учетом того, что в случае сбоя сервисное приложение будет перезапущено агентом.



Класс **QueueWorker** осуществляет добавление методов обратного вызова, инициированных классом ExternalControlCallbackHost во внутреннюю очередь, которая опустошается в процедуре обработки внутреннего таймера *Timer\_Tick*.

Класс **ExternalControlCallbackHost** реализует интерфейс IClientCallback и транслирует вызовы с минимальной внутренней обработкой в QueueWorker.

### Классы визуального интерфейса

Визуальный интерфейс приложения организован при помощи различных функциональных вкладок (наследников класса Page), отображаемых во фрейме главного окна **MainWindow**. Для ускорения процесса старта приложения вкладки не инстанцируются при запуске, а создаются динамически по мере необходимости. Этим процессом (инстанцирование и хранение созданных объектов) управляет статический класс **Cache**.

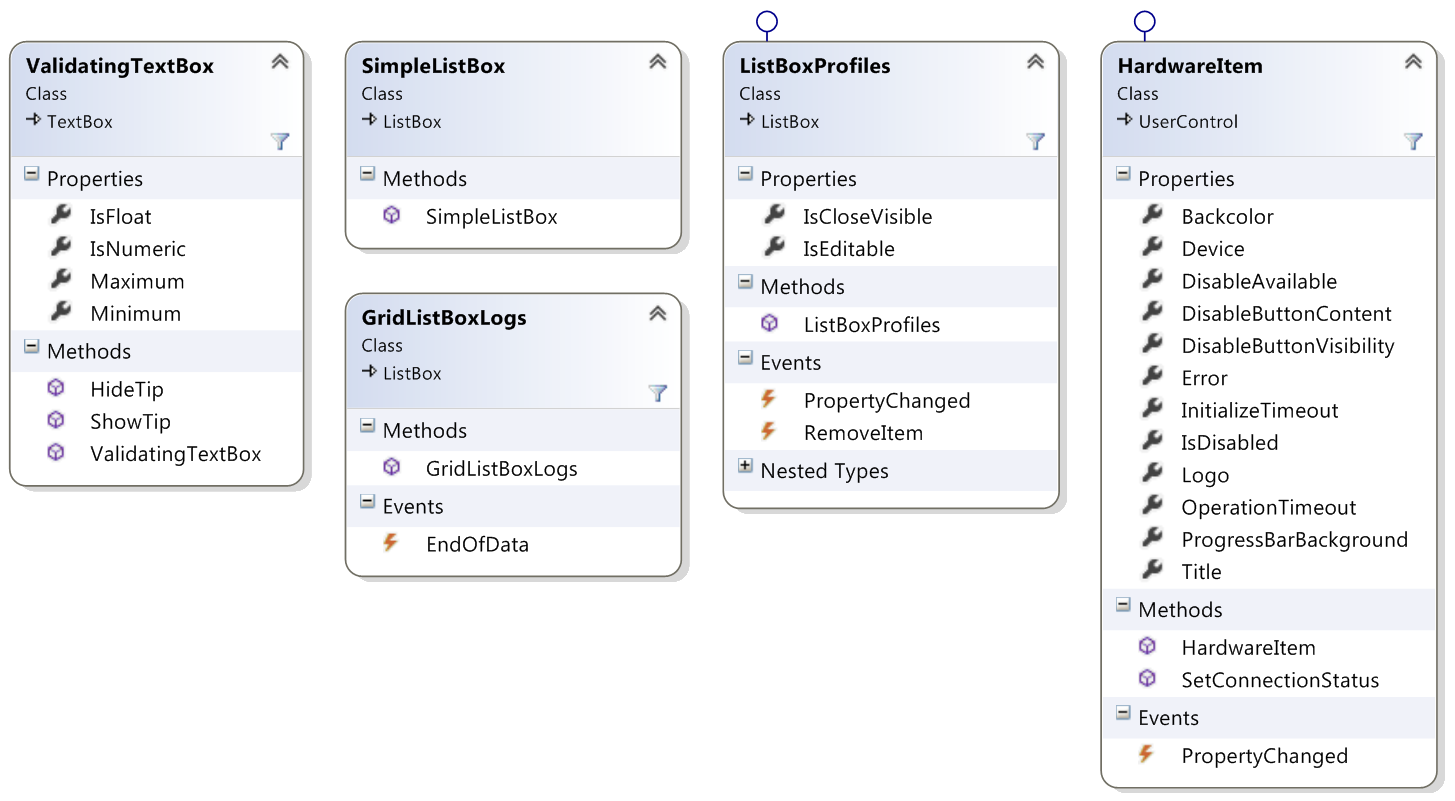
Набор вкладок можно разделить на две группы: вкладки интерфейса оператора и интерфейса наладчика:

1. Интерфейс оператора и общие
   1. **LoginPage** – страница выбора учетной записи оператора и его авторизации;
   2. **ProfileSelectionPage** – страница выбора профиля измерений;
   3. **UserTestPage** – страница проведения поточных измерений приборов;
   4. **ResultsPage** – страница запроса удаленной печати результатов;
   5. **HardwareStatusPage** – страница, отображающая процесс инициализации комплекса;
2. Интерфейс наладчика
   1. **PasswordPage** – страница ввода пароля для входа в режим наладчика;
   2. **TechnicianPage** – страница меню режима наладчика;
   3. **ProfilePage** – страница редактирования локального списка профилей измерений;
   4. **GatePage** – страница измерения характеристик управления в расширенном режиме;
   5. **SLPаge** – страница измерения статических потерь в расширенном режиме;
   6. **BVTPage** – страница измерения блокирующих характеристик в расширенном режиме;
   7. **dVdtPage** – страница измерения характеристик критической скорости нарастания напряжения;
   8. **ClampPage** – страница управления зажимным устройством (прессом);
   9. **CalibratinoPage** – страница доступа к параметрам калибровки блоков;
   10. **ConsolePage** – страница, обеспечивающая функционал прямого доступа к таблицам регистров блоков;
   11. **SeftTestPage** – страница процедуры самотестирования блоков;
   12. **LogsPage** – страница доступа к записям журнала событий;

### Классы специализированных элементов управления

Специализированные элементы управления созданы для специфических задач отображения данных:

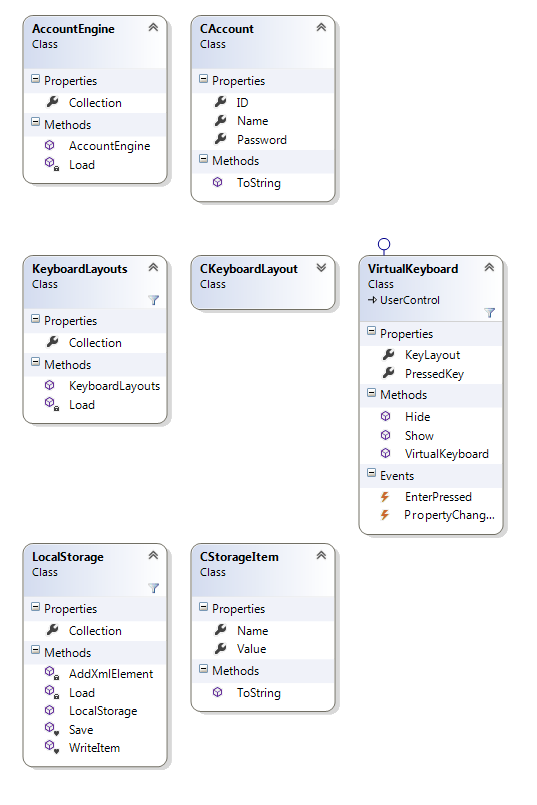
1. **ValidatingTextBox** – предоставляет функциональность полу ввода данных с контролем вводимых значений через свойства *Maximum* и *Minimum*. Формат данных задается свойствами *IsNumeric* и *IsFloat*.
2. **SimpleListBox** – задает простой стилизованный списочный элемент управления, поддерживающий связывание данных через отображение списка строк;
3. **ListBoxProfiles** – список профилей, поддерживающий режим удаления элементов и оповещающий об этом событии через *RemoveItem*;
4. **GridListBoxLogs** – список записей из журнала событий, поддерживающий автоматическую подгрузку содержимого по событию *EndOfData*;
5. **HardwareItem** – графический элемент, отображающий состояние инициализации блока комплекса. Содержит свойства, позволяющие отображать кнопку, отключающую данный блок в текущей сессии работы – *DisableAvailable*, *DisableButtonContent*, *DisableButtonVisibility*. Позволяет управлять внешним видом путем задания свойств *Logo*, *Title*, *Backcolor*, *ProgressBarBackground*. Режим инициализации задается через свойства *OperationTimeout* и *InitializeTimeout*.



### Вспомогательные классы

Вспомогательные классы обеспечивают техническую реализацию различных функций приложения:

1. **AccountEngine** – обеспечивает загрузку списка аккаунтов пользователей, зарегистрированных для работы на данном измерительном комплексе;
2. **CAccount** – представляет собой пользовательский аккаунт для класса AccountEngine с указание имени (*Name*) и пароля (*Password*);
3. **KeyboardLayouts** – обеспечивает загрузку списка раскладок клавиатур;
4. **CKeyboardLayout** – представляет собой локализованную раскладку клавиатуры;
5. **VirtualKeyboard** – виртуальная клавиатура для сенсорного монитора;
6. **LocalStorage** – обеспечивает загрузку списка сохраняемых параметров настройки блоков комплекса;
7. **CStorageItem** – представляет собой сохраняемый параметр;



# Архитектура инфраструктурного программного обеспечения

Внешнее программное обеспечение представлено приложениями SQLDatabaseClient и DatabaseServer.

## Приложение SQLDatabaseClient

Данное приложение предоставляет пользовательский интерфейс для доступа к центральной БД результатов. Пользовательский интерфейс выполнен на основе технологии WPF, доступ к БД осуществляется посредством технологии ORM (Object-relation management) EntityFramework 6.1. Прослойка ORM отвечает за формирование запросов и взаимодействие с SQL Server.

БД SQL Server

Табличные данные

ORM EntityFramework

Модель данных

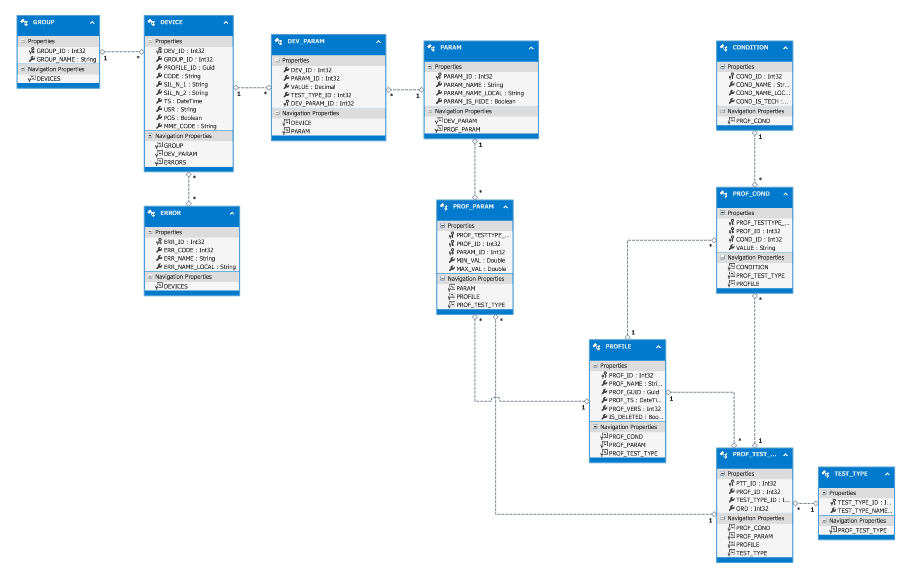
Коллекции объектов

«Контроллер»

Представление данных

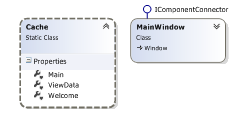
WPF Databinding

Модель данных EntityFramework генерируется на основе метаданных из БД и представляет из себя структуру, представленную на рисунке далее. В данной структуре воспроизводятся связи, аналогичные связям в структуре БД, но на уровне объектов. Ссылки между таблицами преобразуются в навигационные связи, в том числе автоматически разрешаются связи «многие-ко-многим» представленные промежуточной таблицей.

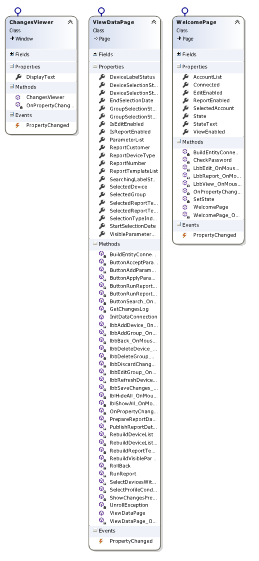


Пользовательский интерфейс представлен страницами:

1. **WelcomePage** – страница аутентификации пользователей, обеспечивающая выбор пользователя и ввод пароля;
2. **ViewDataPage** – страница представления данных, редактирования и генерации отчетов;
3. **ChangesViewer** – окно отображения протокола вносимых в БД изменений;

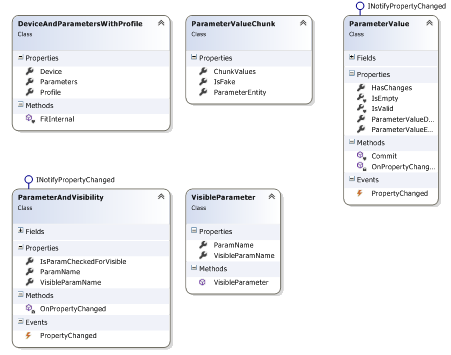


Указанные страницы используют окно **MainWindow** как поверхность отображения и навигации и сохраняются в виде кэшированных объектов в статическом классе **Cache**.



К вспомогательным классам относятся классы, обеспечивающие функционирование контроллера отображения данных:

1. DeviceAndParametersWithProfile – объект, представляющий собой запись из БД об измеренном приборе с сообеветствующими параметрами и прявязанным профилей измерения;
2. ParameterValueChunk – цепочка параметров, ассоциированных с измеренным прибором;
3. ParameterValue – измеренное значение конкретного параметра;
4. VisibleParameter – параметр, отмеченный пользователем для загрузки из БД;
5. ParameterAndVisibility – параметр, присутствующий в БД и доступный для загрузки;



В процессе генерации отчетов в генератор FastReport передаются исходные данные, для которых применяются следующие вспомогательные классы:



1. RestrictionValues – ограничения на значения измеренного параметра согласно профилю измерения;
2. Restionction – максимальное и минимальное значения;
3. ConditionValues – условия измерения;
4. ParametersReportItemBase – базовый класс, на основе которого генерируются передаваемые объекты на оснвое технологии Reflection.

Пространство имен Reflection.Emit предоставляет набор классов для динамической генерации кода на промежуточном языке (MSIL). Для динамической компиляции используется следующая последовательность действий:

1. Создается динамическая сборка (экземпляр класса AssemblyBuilder) с помощью метода AppDomain.DefineDynamicAssembly;

2. Создается модуль (экземпляр класса ModuleBuilder) с помощью метода AssemblyBuilder.DefineDynamicModule.

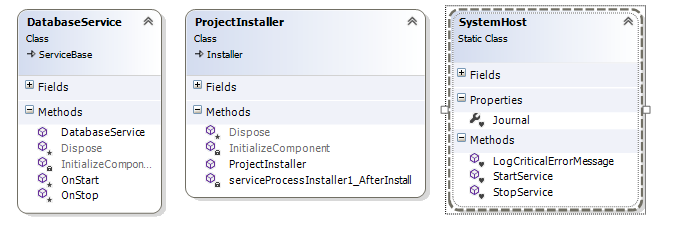
3. Создаются элементы модуля (TypeBuilder, EnumBuilder,..) с помощью методов класса ModuleBuilder.

4. Создаются нужные экземпляры классов (получить соответствующий тип можно с помощью TypeBuilder.CreateType) и вызываются их методы посредством Reflection (если они не были унаследованы от известных вызывающему коду базовых классов и не реализуют известные интерфейсы).

## Приложение DatabaseServer

Данное приложение может запускаться как интерактивное приложение или системный сервис. Основной функционал реализон классом SQLCentralDatabaseService из сборки InterfaceImplementation.

В данном приложении непосредственно реализуются возможности установки/удаления системного сервиса и публикации интерфейса WCF.



В данном проекте интерфейс WCF реализуется при помощи методов класса SystemHost. Эти методы вызываются как при старте интерактивного приложения, так и сервиса.

Сервис системы реализуется при помощи класса DatabaseService, инсталлятором сервиса является класс ProjectInstaller.

Для установки в качестве сервиса приложение должно быть запущено из-под обычной сессии с параметром командной строки: --install , для удаления --uninstall.

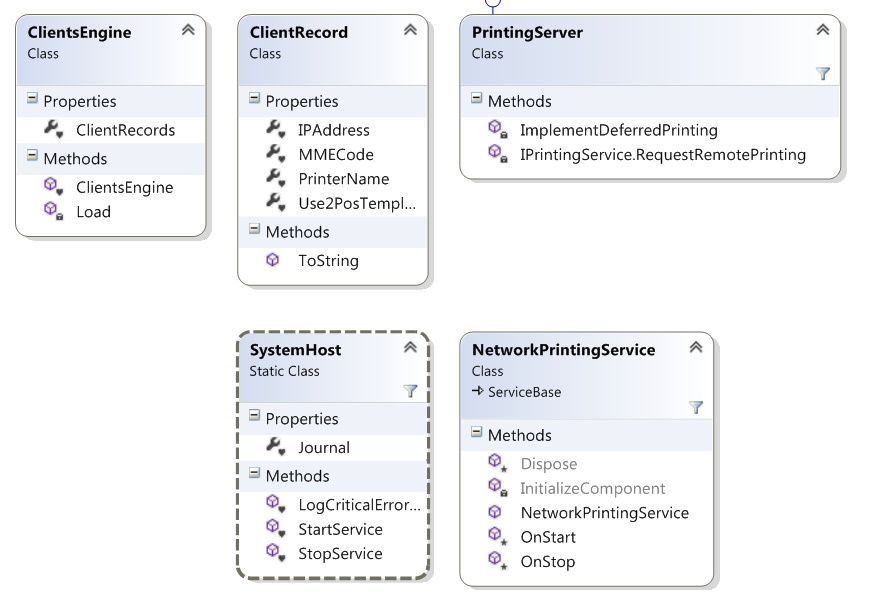
# Архитектура архивного программного обеспечения

## Приложение SCME.NetworkPrinting

Данное приложение может быть запущено как в интерактивной сессии пользователя, так и в качестве службы Windows. Отчеты в формате MS Excel формируются при помощи библиотеки **ExcelPrinting**.

При запуске приложение открывает на прослушку интерфейс WCF IPrintingService, который реализуется классом **PrintingServer**. Пара классов **ClientEngine** и **ClientRecord** представляют собой коллекцию зарегистрированных измерительных комплексов в сети предприятия, которая загружается их конфигурационного файла.

Класс **NetworkPrintingService** предназначен для реализации функции запуска приложения в режиме службы.



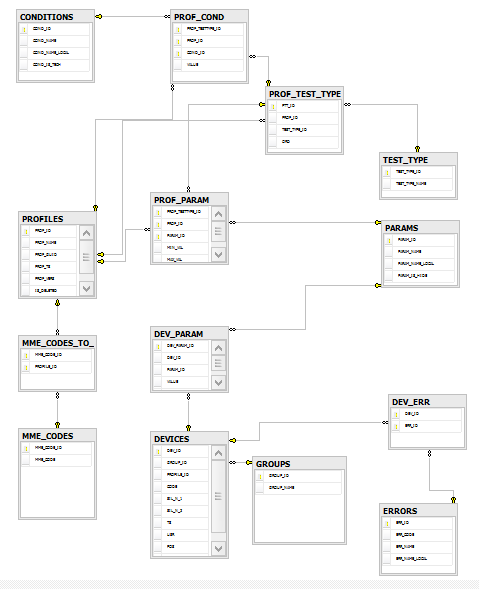
# Структура БД результатов измерений

База данных результатов измерений содержит в себе данные о результатах тестирования приборов и модулей на измерительном комплексе, включая данные о условиях проведения измерений и нормативах на измеряемые параметры.

Так как для хранения данных выбрана реляционная схема, то для достижения 3-ей нормальной формы, структура данных представлена в виде 10 таблиц, из них 4 предназначены для декомпозиции связей типа «многие-ко-многим».

Список таблиц:

1. **GROUPS** – содержит данные о партиях модулей;
   1. GROUP\_NAME – идентификатор партии модулей или приборов;
2. **DEVICES** – содержит данные об измеряемом модуле
   1. CODE – идентификатор прибора или модуля;
   2. SIL\_N\_1 – данные о партии ППЭ;
   3. SIL\_N\_2 – данные о номере ППЭ;
   4. TS – время измерения и дата;
   5. USR – имя аккаунта оператора;
   6. POS – номер позиции прибора (для приборов с двумя позициями в данной таблице содержатся две записи);
   7. MME\_CODE – код измерительного комлпекса;
   8. PROFILE\_ID – код профиля измерения;
3. **PARAMS** – параметры измеряемые на приборе
   1. PARAM\_NAME– название параметра;
   2. PARAM\_NAME\_LOCAL – локализованное название;
   3. PARAM\_IS\_HIDE – параметр не отображается в отчет;
4. **PROFILES** – данные о профилях измерений
   1. PROF\_NAME – отображаемое имя (не уникально, так как профили могут модифицироваться);
   2. PROF\_GUID – уникальный идентификатор профиля (отражает одну из версий профиля измерений с заданным именем);
   3. PROF\_TS - время измерения профиля и дата;
   4. PROF\_VER –версия профиля;
   5. IS\_DELETED – признак того, что профиль удален;
5. **CONDITIONS** – задает условия проведения измерений
   1. COND\_NAME– имя условия;
   2. COND\_NAME\_LOCAL – локализованное имя условия;
   3. COND\_IS\_TECH – флаг того, что условие является техническим параметром измерительного комплекса;
6. **ERRORS** – ошибки измерений
   1. ERR\_CODE – код ошибки;
   2. ERR\_NAME – интернациональное название;
   3. ERR\_NAME\_LOCAL – локализованное название;
7. **PROF**\_**COND** – значения заданного условия измерений в указанном профиле;
   1. PROF\_TESTTYPE\_ID – идентификатор измерения, к которому принадлежит значение;
   2. VALUE – значения условия;
8. **PROF\_PARAM** – нормативы на заданный параметр в указанном профиле;
   1. PROF\_TESTTYPE\_ID – идентификатор измерения, к которому принадлежит значение;
   2. MIN\_VAL – минимальное значение параметра;
   3. MAX\_VAL – максимальное значение параметра;
9. **DEV**\_**PARAM** – значение заданного параметра в указанном приборе;
   1. VALUE – значение параметра;
10. **DEV**\_**ERR** – присутствие заданного кода ошибки для указанного прибора;
11. **PROF\_TEST\_TYPE** – измерения, присутствующие в указанном профиле;
    1. ORD – порядок измерения;
12. **TEST\_TYPE** – типы тестов;
    1. TEST\_TYPE\_NAME – название типа измерения
    2. TEST\_TYPE\_ID – код теста;



# Дальнейшее усовершенствование системы

## Доработка системы профилей

В качестве дальнейшей доработки системы профилей необходимо ввести новую абстракцию профилей, как условий проведения измерений приборов вне зависимости от того, проводится ли измерение при помощи MME, либо при помощи ручного измерения.

Создаваемые профили измерения должны иметь возможность привязки к результатам измерения, вводимым при помощи средств ручного редактирования (SQLDatabaseClient) и верификации вводимых данных.

## Доработка системы ввода данных

При использовании методов измерения отличных от MME дальнейшим развитием системы ручного ввода данных является система ввода на основе импорта данных из внешнего файла (например, Excel) с одновременной валидацией данных на основе профиля измерения ручного ввода.

Это позволит увеличить производительность и эффективность данной операции при уменьшении количества ошибок и повышении структурированности процесса.

## Доработка системы отчетов

Движение данных в базе в процессе проведения измерений происходит в режиме PUSH, то есть проведенные измерения заносятся в БД по факту выполнения. Генерация отчетов должна происходить по принципу PULL, то есть извлечения необходимых данных из БД на основе одного или нескольких измерений. Для этого необходимо произвести созданием виртуального каталога идеальных приборов, для которых определены базовые характеристики, интересующие при производстве и включаемые в отчет.

Процесс извлечения данных будет основываться на поиске данных среди проведенных измерений для заполнения отчета. В этому процессе также могут быть выявлены недостающие и дублированные данные.

## Доработка системы синхронизации

Для повышения надежности системы синхронизации результатов из разных БД необходимо ввести двустадийную синхронизацию, в которой данные из комплексов записываются во временную БД, а перенесение в центральную БД осуществляется единовременно в ночное время с формированием отчета по перенесению данных и выявлению несогласованностей. Это позволит гарантировать целостность основной БД результатов в любой момент времени вне зависимости от технических модернизаций.