2015

Платформа ОПТИМУМ

Версия 3.2

РУКОВОДСТВО РАЗРАБОТЧИКА

Оглавление

Термины и сокращения 3
Общее описание системы 4
Назначение платформы 4
Компоненты платформы 4
БД MS SQL Server 4
Сервис лицензирования 4
Сервис синхронизации 4
Центр управления платформой 4
Сервис обмена данными 4
Мобильные библиотеки 5
Плагины 5
Принципы работы платформы 6
Общее описание работы платформы 6
Синхронизируемые таблицы 7
Переменные платформы 8
Группы синхронизации 8
Обмен данными 8
Лицензирование 9
Системные требования 9
Требования к серверной части системы 9
Требования к мобильной части системы 9
Требования к учетной записи MS SQL 9
Установка и первичная настройка компонентов платформы 10
Установка компонентов платформы 10
Запуск Центра управления 14
Интерфейс Центра управления 15
Таблицы 16
Переменные 21
Группы синхронизации 22
Соединения 24
Объекты обмена 26
Расписания обмена 28
Группы обмена 29
Моб. часть 31

Лицензии 32
Устройства 33
Журнал синхронизации 36
Аутентификация 37
Журнал обмена 38
Настройка Сервиса обмена 40
Настройка Сервиса лицензирования 41
Настройка Сервиса синхронизации 43
Методика разработки приложений 44
Быстрый старт 44
Первоначальные требования 44
Создание и настройка проекта 45
Обмен 54
Создание проекта мобильного приложения Android 62
Соединение и синхронизация 67
Руководство разработчика серверной части 68
Определение синхронизируемых таблиц 68
Выбор стратегии сегментации данных 68
Использование специальных функций для работы с данными 69
Использование переменных платформы 70
Прямая вставка данных в синхронизируемые таблицы 71
Генерация данных для разработчика мобильной части 72
Использование переменных платформы 72
Руководство разработчика мобильной части 74
Файл-описание БД и библиотеки платформы 74
Разработка под Android 74
Работа с БД 74
Основные настройки 74
Синхронизация 75
Переменные платформы 75
Примеры 75
Рекомендации 77
Рекомендации по реализации периодической автоматической синхронизации 77
Рекомендации по отправке логов и другой технической информации в службу техподдержки 78
Ролевая модель данных 81
Рекомендации по построению высоконагруженной, отказоустойчивой, масштабируемой системы 82

Термины и сокращения

Платформа ОПТИМУМ – именованный набор модулей, предназначенный для разработки мобильных приложений, использующих архитектуру «Клиент-Сервер».

Проект платформы – комплекс настроек платформы, предназначенный для решения определённой бизнес-задачи.

КИС, BackEnd – корпоративная информационная система.

БД – база данных.

ГС – группа синхронизации.

МУ – мобильное устройство.

МЧ – мобильная часть.

ОО – объект обмена.

Сегментация – фильтрация данных синхронизируемых таблиц для разных мобильных устройств на этапе синхронизации.

СЛ – Сервис лицензирования.

СО – Сервис обмена.

СС – Сервис синхронизации.

СТ – синхронизируемая таблица.

ПА – плагин аутентификации.

ПО – плагин обмена или программное обеспечение, в зависимости от контекста.

ПП – переменная платформы.

ЦУ – Центр управления платформой.

Общее описание системы

Назначение платформы

Платформа ОПТИМУМ предназначена для разработки мобильных приложений, использующих архитектуру «Клиент-Сервер». Платформа поддерживает создание клиентов на операционных системах Android, Windows Phone, Windows 8.1 и iOS. Наиболее выгодно использовать платформу в приложениях, которые обмениваются большим объёмом данных и должны поддерживать режим «offline».

Компоненты платформы

В состав платформы входит следующее ПО:

- Сервис лицензирования;
- Сервис синхронизации;
- Центр управления платформой;
- Сервис обмена данными;
- Плагины для работы с источниками данных;
- Плагины аутентификации;
- Мобильные библиотеки;
- Документация;
- Примеры;
- БД MS SQL Server создается в проекте платформы.

БД MS SQL Server

БД выполняет две основные функции:

- хранит данные, необходимые для функционирования платформы – настройки, пользовательские алгоритмы обработки данных и т.д.;
- выступает в роли кэширующей БД, т.е. в ней содержатся бизнес-данные на пути из BackEnd на мобильные устройства или с мобильных устройств в BackEnd.

Сервис лицензирования

Контролирует наличие ключей лицензирования и предоставляет информацию о лицензиях остальным модулям платформы.

Сервис синхронизации

Обеспечивает синхронизацию данных между кэширующей БД и мобильными устройствами.

Центр управления платформой

Позволяет управлять экземплярами платформы, в том числе создавать новые БД платформы, создавать и редактировать все сущности платформы – таблицы и группы синхронизации, переменные платформы и т.д. Кроме того, в центре управления платформой расположен интерфейс управления Сервисом обмена данными. Интерфейс управления СО позволяет определять соединения с источниками данных, объекты обмена данными, группы обмена и настраивать расписание обмена. Осуществляет управление лицензиями и их учет.

Сервис обмена данными

Обеспечивает интеграцию BackEnd и кэширующей базы платформы. В соответствии с настройками, пересылает данные из таблиц источников данных в синхронизируемые таблицы платформы и обратно.

Мобильные библиотеки

Набор программных библиотек, предназначенных для использования разработчиком мобильного приложения. Отвечает за создание и поддержание актуальной структуры БД мобильного приложения, определение изменившихся данных и синхронизацию данных.

Плагины

Плагин – это подключаемый программный модуль, расширяющий возможности основного программного продукта. Правила написания плагина описываются производителем программного обеспечения. Реализация необходимого плагина может быть выполнена силами пользователя платформы.

В платформе используются два вида плагинов:

1) Плагин аутентификации (ПА);
2) Плагин обмена (ПО).

Для разработки плагинов платформы предоставляется специальная библиотека - Optimum.Interfaces.dll. В этой библиотеке описаны интерфейсы, которые должны реализовывать плагины, работающие с платформой.

Плагины аутентификации напрямую используются Сервисом синхронизации для организации того или иного метода аутентификации пользователей мобильного приложения. Центр управления также взаимодействует с плагинами, но не напрямую, а через Сервис синхронизации, являясь инструментом пользователя для ввода и передачи в СС настроек плагинов, а также для запуска проверки системы аутентификации, настроенной с помощью того или иного плагина.

Плагины обмена напрямую используются Сервисом обмена для организации соединений платформы с различными типами BackEnd. Центр управления предоставляет пользователю интерфейс для ввода и передачи в СО настроек плагинов, а также для запуска проверки соединения, настроенного с помощью того или иного плагина.

СС или СО при запуске просматривают директорию, в которой они установлены, и загружают все файлы dll, которые в ней есть. Далее они проверяют, содержится ли в этих файлах плагин (ПА или ПО соответственно). Если содержится – то они загружают этот плагин, и он становится доступным для использования в проекте.

Плагин аутентификации

ПА служит для проверки логина и пароля пользователя. В рамках одного проекта может использоваться только один ПА. Когда мобильный разработчик в своем коде (на МУ) вызывает функцию платформы authenticate (login, password), логин и пароль, которые он туда передает параметрами, передаются в Сервис синхронизации. Сервис синхронизации определяет, какой плагин используется на проекте и с какими настройками. После этого СС передает плагину логин и пароль, полученные с устройства, и плагин отвечает ему, правильные они или нет.

Если в ходе проекта выявляется потребность в собственном методе аутентификации (например, список логинов и соответствующих паролей должен храниться в текстовом файле), то разрабатывается отдельный плагин. Например, этот плагин будет открывать текстовый файл, искать в нем указанный логин, сверять указанный пароль с паролем из файла и возвращать результат. Для подключения этого разработанного плагина необходимо будет только скопировать его в директорию СС и перезапустить службу СС.

Плагин обмена

Плагины обмена используются Сервисом обмена и служат для чтения данных из какого-либо источника или для записи данных в какой-либо источник (приемник). В рамках одного проекта может применяться неограниченное множество разных плагинов обмена. На основе плагина обмена создаются «Соединения» проекта. На основе одного плагина может быть создано несколько разных соединений (с разными настройками).

Например, с использованием плагина MS SQL может быть создано два соединения – к двум разным БД на разных серверах.

Кроме того, для каждого проекта автоматически создается специальное соединение – «Platform» на основе специального, платформенного плагина.

Задача плагина обмена – быть настраиваемым и предоставить операции чтения и/или записи данных.

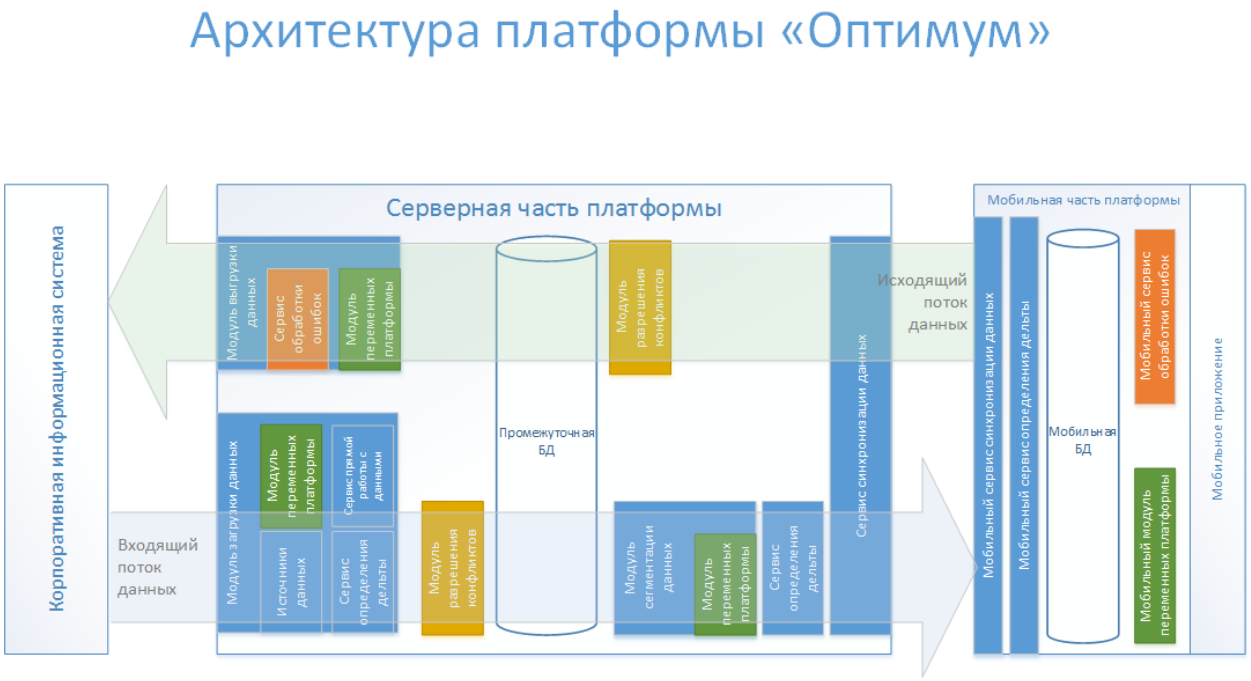
На текущий момент разработано три плагина обмена (кроме платформы) – для работы с СУБД MS SQL, MySql и Oracle. Они все обеспечивают и чтение, и запись данных, однако плагин может быть (но не обязательно) сделан только для чтения или только для записи данных.

Предполагается, что откуда бы плагин ни читал данные, он может (должен) их в конечном итоге привести в вид таблицы – со столбцами и строками.

При записи данных предполагается, что плагин получает данные в виде таблицы и записывает их в соответствии со своей логикой.

Плагины обмена могут разрабатываться в двух случаях – для осуществления обмена данными как таковым, и для обработки каких-либо сложных случаев выгрузки или загрузки данных.

Принципы работы платформы



Общее описание работы платформы

Платформа передает данные от источника данных до мобильного устройства и обратно, используя кэширующую БД. В ходе разработки проекта разработчик серверной части должен описать модель передаваемых данных, наладить интеграцию данных с кеширующей БД и произвести прочие

настройки, специфичные для платформы – описать сегментацию данных, принципы разрешения конфликтов, создать переменные платформы и т.д. Разработчик мобильной части, используя библиотеки платформы под соответствующую нативную среду разработки, создает мобильную БД и вызывает функции платформы для установки параметров и обмена данными.

Синхронизируемые таблицы

Синхронизируемая таблица – это основная сущность платформы, хранящая в себе данные. СТ должна быть создана через Центр управления вначале проекта. В СТ передаются данные из КИС и отправляются на мобильное устройство. В обратном направлении данные передаются из мобильной части в СТ, а затем в КИС.

СТ может восприниматься как обычная таблица реляционной БД, имеющая некоторые дополнительные свойства, такие как приоритеты синхронизации, тип синхронизации, методы разрешения конфликтов.

Поля синхронизируемой таблицы

Каждая синхронизируемая таблица состоит из набора полей. Для каждого поля устанавливается имя, тип данных, признак первичного ключа и признак допустимости пустых значений.

Приоритеты синхронизации

Приоритет при передаче с сервера на МЧ – порядок, в котором данная таблица будет передаваться с сервера на мобильное устройство.

Приоритет при передаче с МЧ на сервер – порядок, в котором данная таблица будет передаваться с мобильного устройства на сервер.

Приоритет учитывается в порядке возрастания, то есть приоритет 1 – наивысший.

Типы синхронизации

- Не синхронизируется – данные из СТ не уходят на мобильное устройство, а используются только в сегментации.
- По дате изменения – на устройство отправляются данные, время изменения которых больше времени последней синхронизации устройства. Сегментация не выполняется.
- По состоянию – на устройство отправляются все данные, которые на нем отсутствуют. Также на устройство отправляются данные, которые уже присутствуют на устройстве, но были изменены в платформе. Отсутствующие или измененные данные определяются автоматически, сегментация не выполняется.
- Сегментация простая – для определения данных, которые должны быть отправлены на мобильное устройство, используется запрос сегментации в простом виде (одна SQL выборка, состоящая из одного результата).
- Сегментация сложная – для определения данных, которые должны быть отправлены на мобильное устройство, используется запрос сегментации в сложном виде, заполняющий временную таблицу (используются все возможности TSQL- вызовы процедур, функций, создание промежуточных таблиц).

В случае если для таблицы выбран тип синхронизации с сегментацией, то для этой таблицы должен быть написан запрос сегментации. Запрос обязан сформировать данные, которые должны быть на конкретном мобильном устройстве. Далее к этим данным будут автоматически применены операции определения дельты, после чего они будут отправлены на мобильное устройство. В запросе сегментации для доступа к данным базы платформы и данным мобильного устройства должны быть использованы специальные функции, автоматически сгенерированные для каждой синхронизируемой таблицы при ее создании. Эти функции возвращают текущие наборы данных для таблицы на сервере и на мобильном устройстве.

В случае если для СТ установлен тип синхронизации, подразумевающий сегментацию, но запрос сегментации пуст, тип синхронизации для этой СТ принимается за «По состоянию».

Методы разрешения конфликтов

Для синхронизируемой таблицы устанавливается тип разрешения конфликтов. Конфликтом считается ситуация, когда какая-либо запись изменяется в нескольких местах (например, в КИС и на мобильном устройстве). Конфликт выявляется на этапе обработки данных при синхронизации в любом направлении, когда становится ясным, что та или иная запись была изменена и в КИС, и в МЧ **до начала синхронизации**. В этот момент включается механизм разрешения конфликтов.

Предусмотрены варианты разрешения конфликтов:

- Приоритет КИС. За действительную принимается запись, пришедшая из КИС.
- Приоритет МЧ. За действительную принимается запись, пришедшая с мобильного устройства.
- По времени: кто раньше. Действительной считается запись, измененная раньше, независимо от источника.
- По времени: кто позже. Действительной считается запись, измененная позже, независимо от источника.

Переменные платформы

Переменные платформы – это набор созданных при разработке проекта переменных определенного типа. На каждом мобильном устройстве любой ПП могут быть установлены индивидуальные значения. В дальнейшем эти значения попадают на сервер и могут быть использованы в некоторых серверных алгоритмах, например, в запросах сегментации, в процессе обмена данными с КИС, в процессе обработки ошибок и т.д. Поскольку значения переменных платформы индивидуальны для каждого мобильного устройства, использоваться на сервере они могут только в контексте синхронизации конкретного устройства.

Группы синхронизации

Синхронизируемые таблицы объединяются в группы синхронизации. На мобильном устройстве при запуске синхронизации указывается ГС, для которой необходимо провести синхронизацию. В рамках запущенной сессии синхронизации будут синхронизированы только таблицы, входящие в указанную группу. По умолчанию все вновь созданные таблицы добавляются в группу синхронизации «default».

Синхронизируемая таблица может входить одновременно в несколько групп синхронизации.

В процессе синхронизации таблицы, входящие в указанную группу синхронизации, синхронизируются по порядку, указанному в мобильном и серверном приоритетах синхронизации. При этом сначала все таблицы передаются с мобильного устройства на сервер, а потом – с сервера на мобильное устройство.

Обмен данными

Сервис обмена данными обеспечивает передачу данных из КИС в платформу и из платформы в КИС. Параметры обмена данными настраиваются через Центр управления в рамках конкретного проекта. Для реализации алгоритмов источников и приемников данных используется концепция плагинов с поставляемыми в комплекте штатными плагинами для наиболее распространенных информационных систем. Интерфейс плагинов является открытым, что позволяет реализовать собственными силами плагин для доступа к любой КИС или оригинальный/сложный алгоритм выгрузки данных.

Соединения

Соединение определяет тип информационной системы (используемый плагин). В дальнейшем соединение используется объектами обмена в качестве источника или приемника данных.

Объект обмена

Объект обмена определяет, откуда берутся данные (соединение-источник) и куда записываются (соединение-приемник). Поскольку соединение само по себе предоставляет только связь с КИС, в объекте обмена для соединения-источника задается запрос на получение данных и для соединения-приемника запрос на запись (обработку) данных. Форма запроса зависит от сущности соединения. Например, для БД это может быть sql-запрос, а для платформы – просто имя синхронизируемой таблицы. После того, как настроены запросы получения и обработки данных, определяется маппинг полей. Маппинг полей указывает соответствие полей в источнике и приемнике данных.

Группы обмена

Объекты обмена объединяются в группы обмена. При инициации обмена происходит выполнение всех ОО из группы обмена. Инициация обмена может произойти по расписанию или по наступлению определенного события (например, синхронизации мобильного устройства).

Расписания

Расписания позволяют спланировать запуск процедуры обмена на регулярной основе.

Лицензирование

В поставку платформы входят две полнофункциональные лицензии для мобильных устройств.

Системные требования

Требования к серверной части системы

Серверная часть платформы требует для работы ОС Windows 7 или выше, а также – MS SQL Сервер версии 2008 или выше. Также должна быть установлена платформа .NET Framework версии 4.0 или старше.

Требования к мобильной части системы

Мобильные библиотеки платформы работают на:

- ОС Android версии 2.3.3 или выше;
- Windows Phone версии 8.1 и выше;
- Windows 8.1 и выше;
- iOS версии 5.0 или выше.

Требования к учетной записи MS SQL

Учетная запись MS SQL, от которой работают ЦУ, СО, СС и СЛ, должна обладать следующими правами:

Для работы ЦУ требуются права уровня сервера:

- ALTER ANY DATABASE;
- ALTER ANY LOGIN;
- CONNECT SQL.

Для работы СС и СО требуются права:

- На уровне сервера:
    - ADMINISTER BULK OPERATIONS;

- На уровне БД:
    - db_datareader;
    - db_datawriter.

Для работы СЛ требуются права уровня базы данных:

- db_datareader;
- db_datawriter.

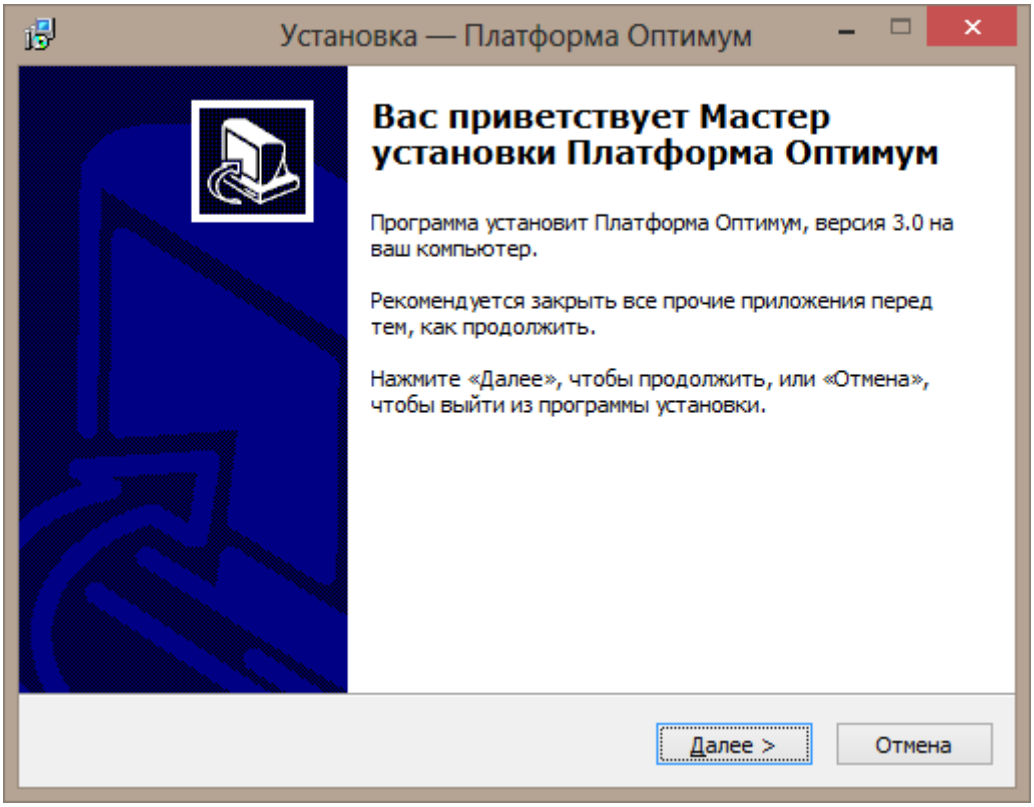
Установка и первичная настройка компонентов платформы

Установка всех компонентов платформы производится с помощью единого инсталлятора. Все компоненты платформы могут быть установлены на одной или на разных рабочих станциях. СУБД MS SQL Server, которая необходима для работы платформы, не входит в состав платформы и должна быть установлена отдельно.

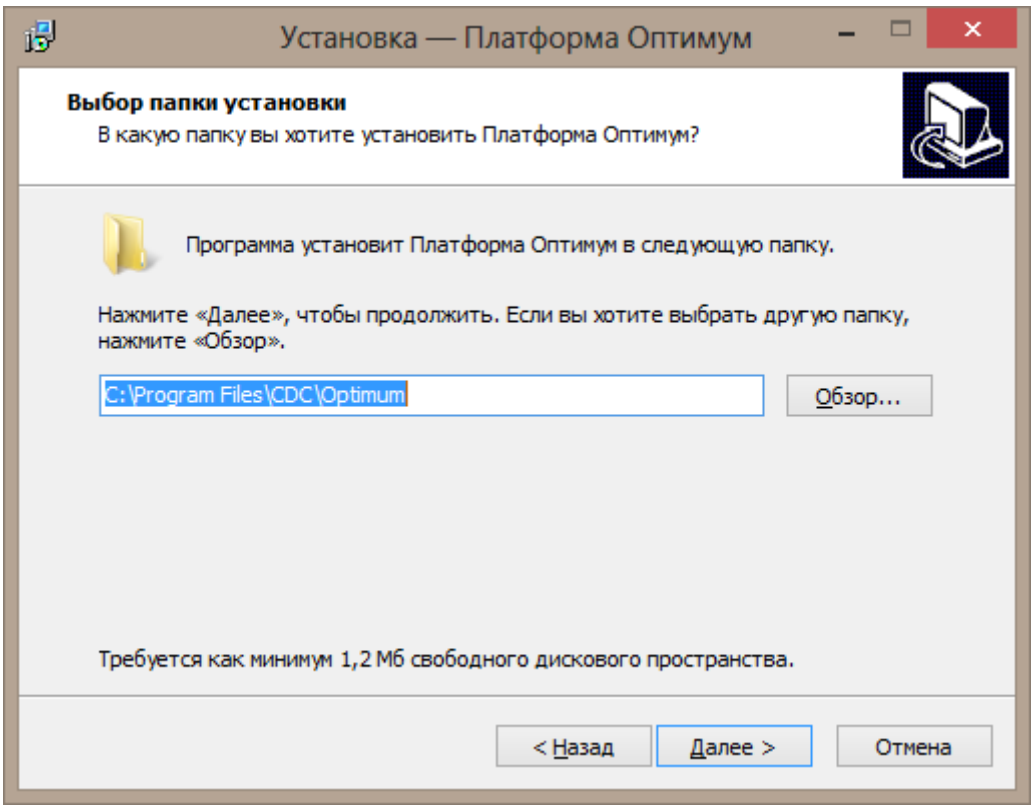
Установка компонентов платформы

Для того чтобы установить компоненты платформы, выполните следующие действия:

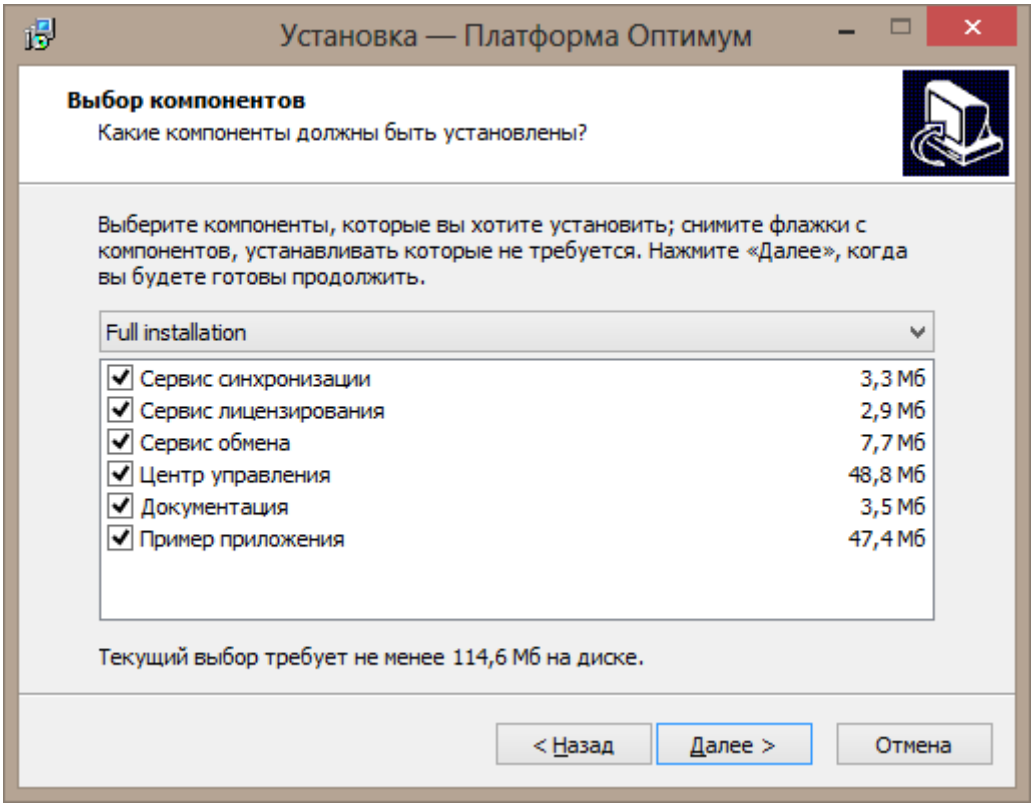
1. Запустите дистрибутив setup_optimum_platform_ru.exe и в открывшемся окне нажмите **Далее**.



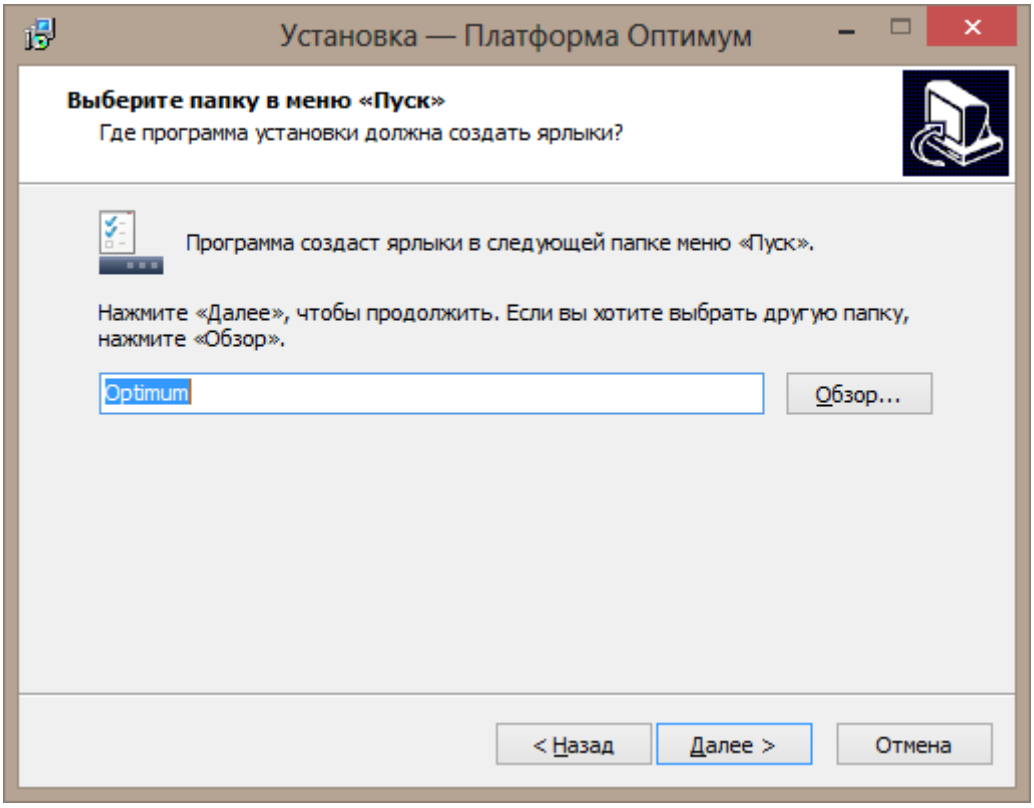
2. Выберите директорию для установки и нажмите **Далее**.



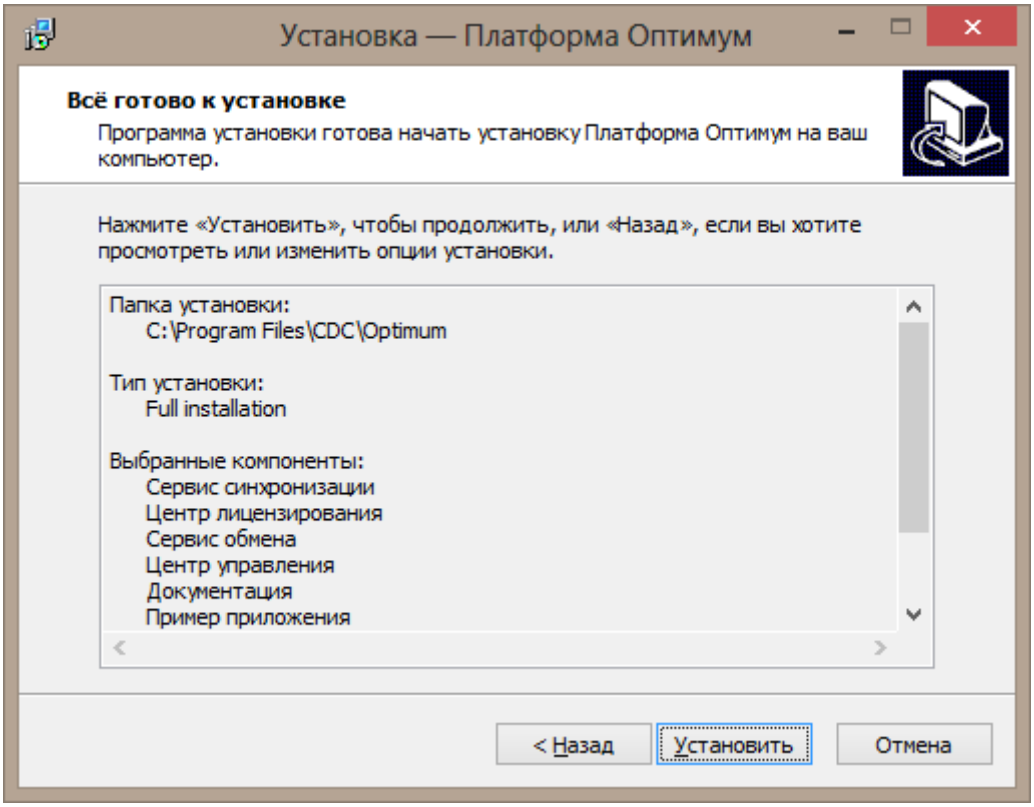
3. Выберите тип установки – Full installation и нажмите **Далее**.



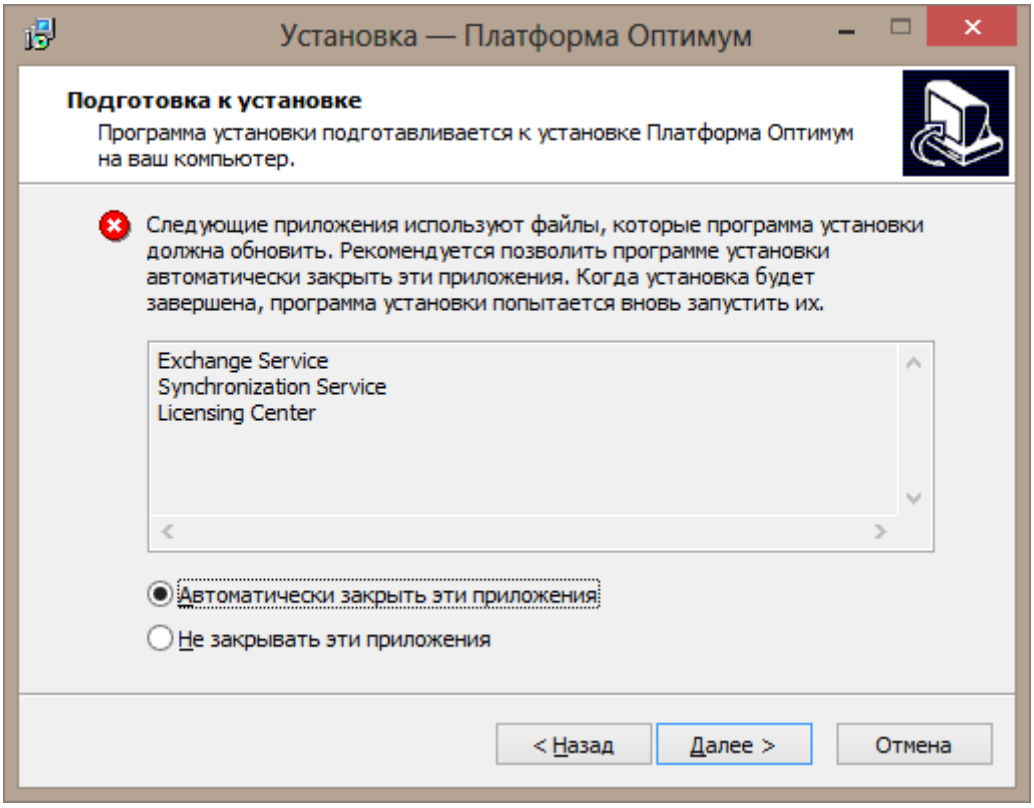
4. Выберите папку в меню **Пуск** и нажмите **Далее**.



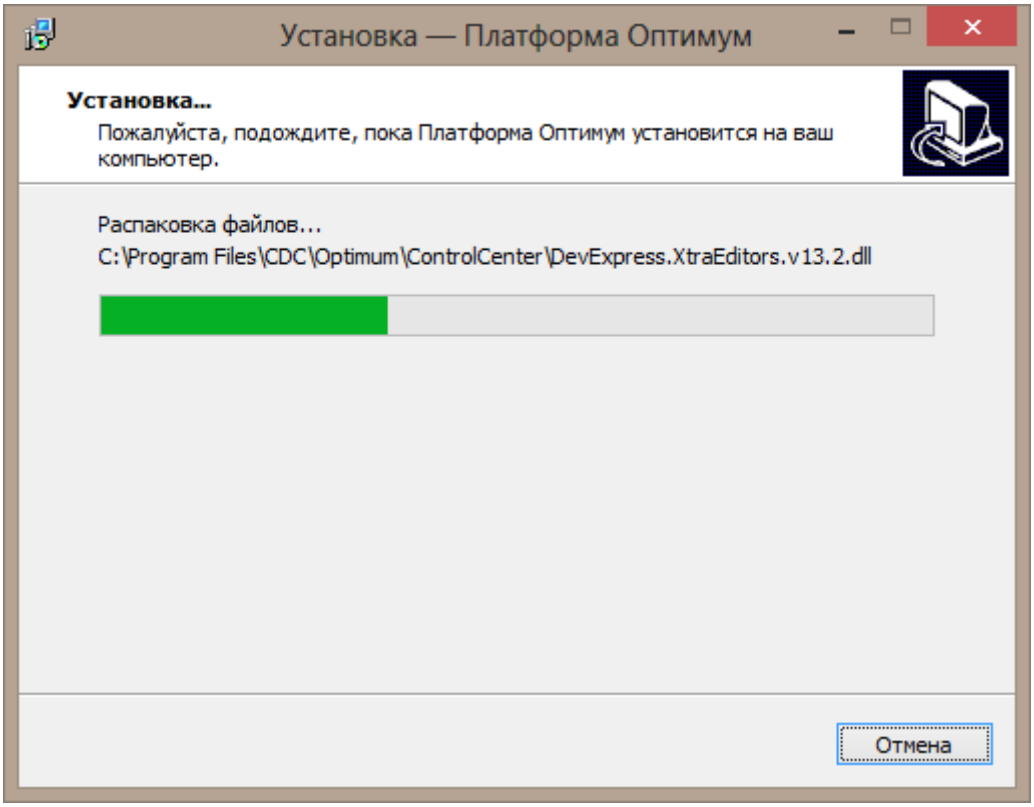
5. Проверьте параметры установки и нажмите **Установить**.



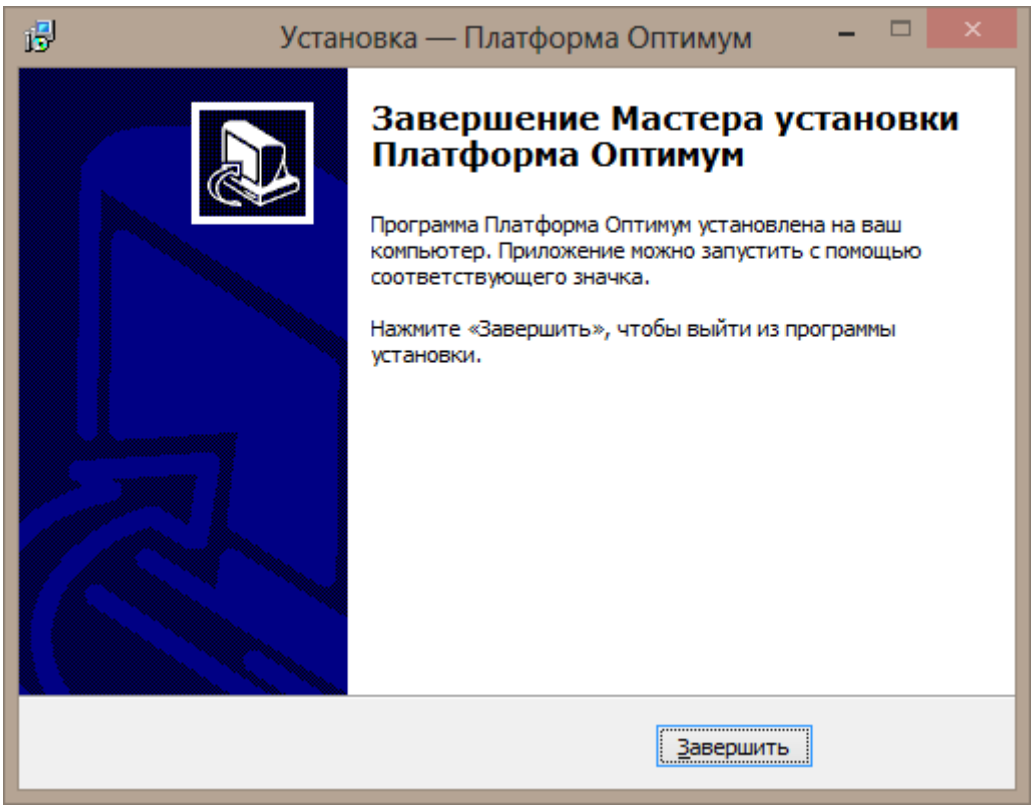
6. Если те или иные служебные файлы, которые инсталлятор задействует в процессе установки, заняты какими-либо программами, эти программы будет предложено закрыть.



7. Будет запущена установка компонентов платформы.



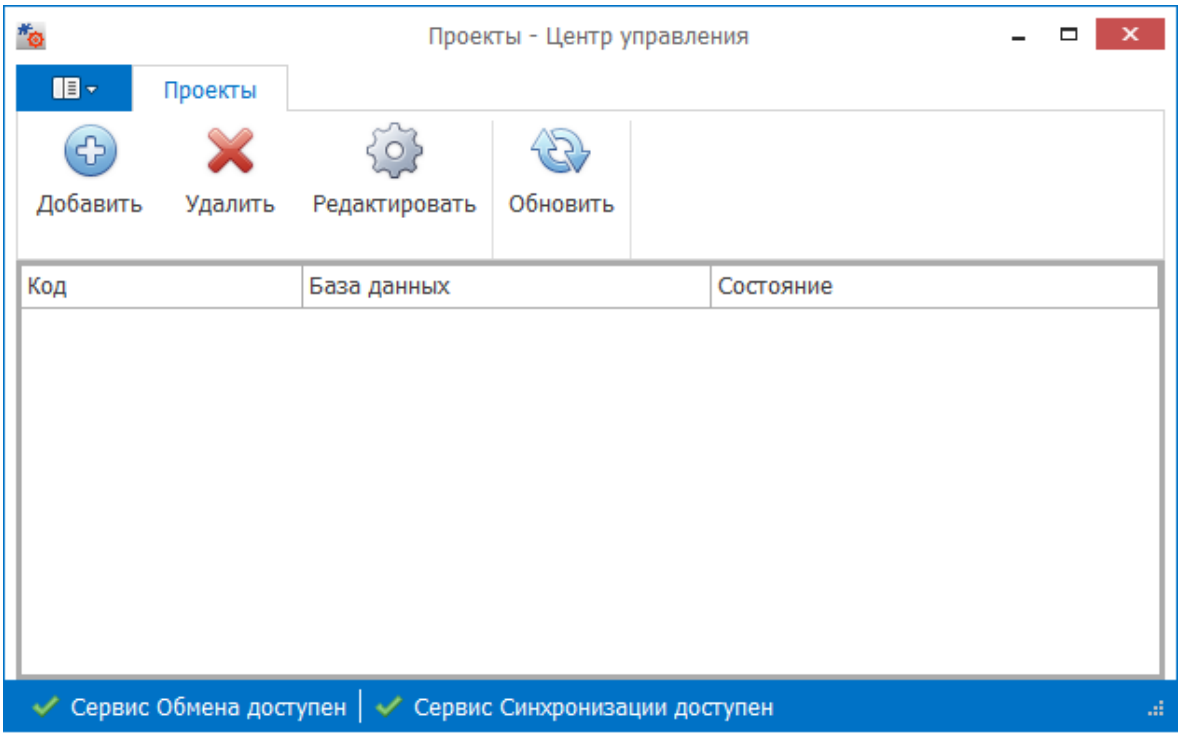
8. По окончании установки нажмите **Завершить**.



Процесс установки будет завершен.

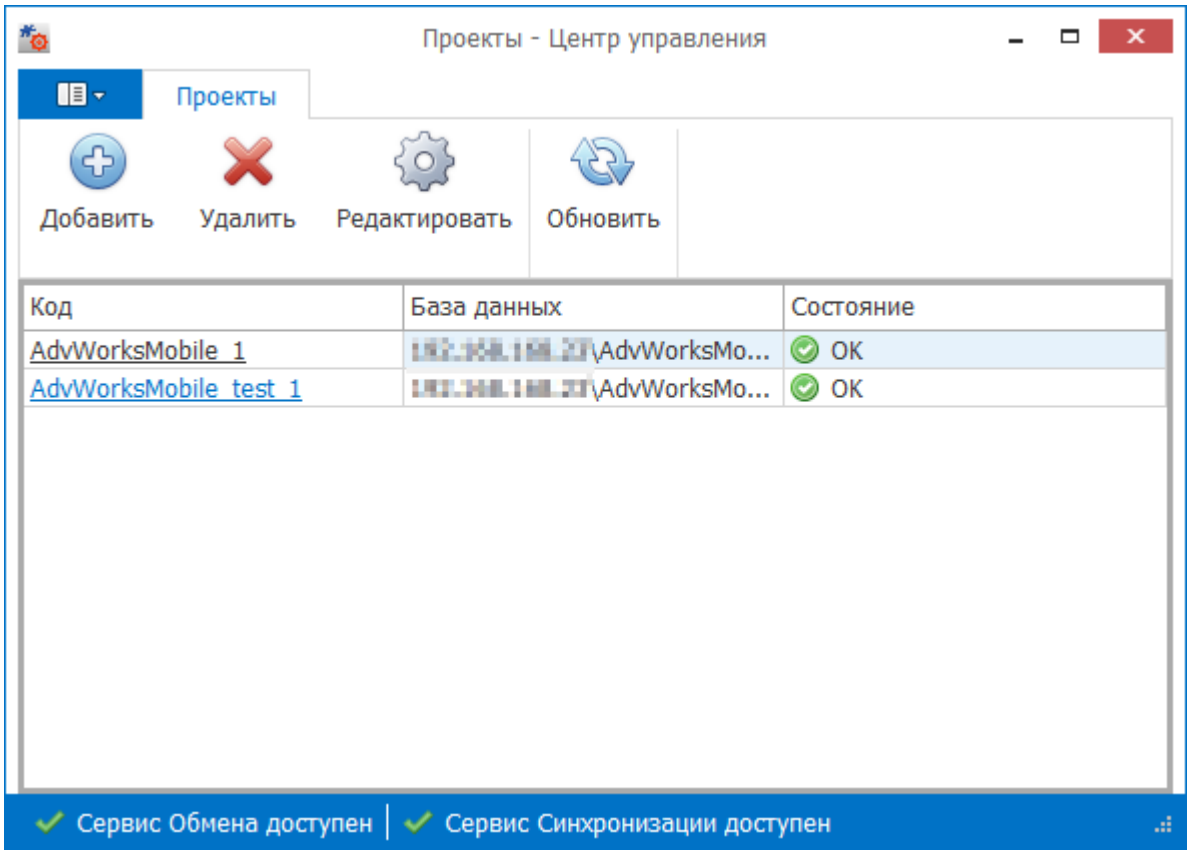
Запуск Центра управления

Для запуска Центра управления платформой воспользуйтесь ярлыком на рабочем столе или запустите файл ControlCenter.exe в директории установки компонента платформы ControlCenter. Откроется главное окно Центра управления.



Интерфейс Центра управления

Стартовое окно Центра управления содержит вкладку **Проекты**. На этой вкладке отображается перечень созданных проектов.



Пользователю доступны следующие операции:

- Добавить – создание нового проекта;
- Удалить – удаление проекта;
- Редактировать – редактирование настроек соединения проекта;
- Обновление данных.

Из стартового окна нажатием на область  пользователь может открыть меню приложения, в котором содержатся следующие пункты:

- Сведения – Сведения о программном продукте и контактная информация.
- Параметры – параметры настройки компонентов платформы.
- Закрыть – выход из приложения.

После создания проекта на вкладке **Проекты** для перехода в режим работы с проектом необходимо нажать на его название.

Внутри проекта пользователь выполняет его настройку, перемещаясь по вкладкам:

- **Таблицы** – на вкладке осуществляется создание и удаление таблиц БД платформы.
- **Переменные** – на вкладке создаются и удаляются переменные платформы.

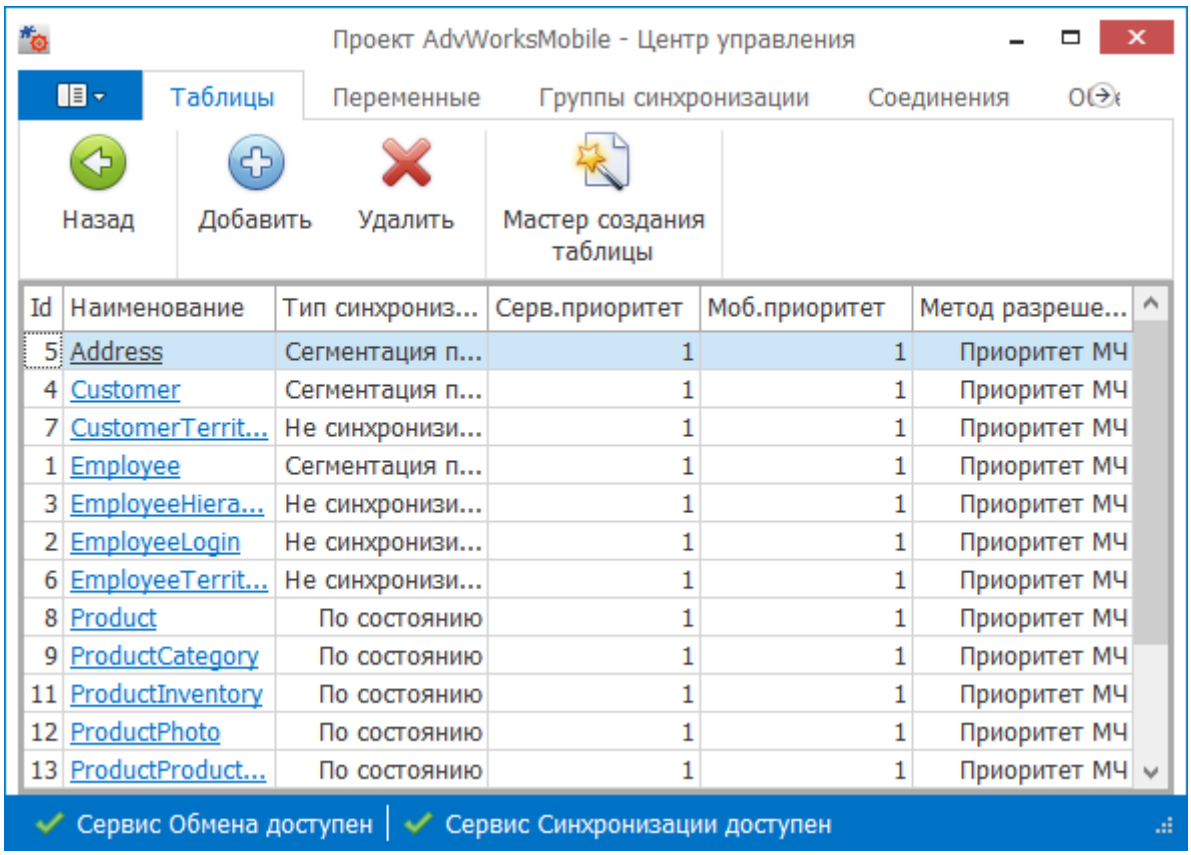
- **Группы синхронизации** – на вкладке формируются и удаляются группы синхронизации.
- **Соединения** – на вкладке настраиваются соединения с источником и приёмником данных. Также осуществляется удаление созданных соединений.
- **Объекты обмена** – на вкладке настраиваются и удаляются объекты обмена.
- **Расписания обмена** – на вкладке осуществляется формирование и удаление расписаний запуска процедур обмена данными.
- **Группы обмена** – на вкладке выполняется объединение объектов обмена в группы обмена. Также осуществляется удаление групп обмена.
- **Моб. часть** – вкладка служит для получения сформированного конфигурационного файла мобильной части.
- **Лицензии** – на вкладке осуществляется управление лицензиями.
- **Устройства** – на вкладке отображается перечень всех мобильных устройств, проводивших синхронизацию с платформой, к текущему моменту.
- **Журнал синхронизации** – журнал содержит записи о событиях Сервиса синхронизации.
- **Аутентификация** – на вкладке осуществляется настройка и проверка аутентификации в проекте.
- **Журнал обмена** – журнал содержит записи о событиях Сервиса обмена.

На каждой из вкладок есть кнопка **Назад**, по нажатию на которую происходит возврат в стартовое окно на вкладку **Проекты**.

Таблицы

На вкладке **Таблицы** выводится список синхронизируемых таблиц, созданных в рамках проекта. Для каждой таблицы отображается:

- Id – порядковый номер, который присваивается таблице при создании;
- Наименование – имя таблицы;
- Тип синхронизации;
- Серв. приоритет;
- Моб. приоритет;
- Метод разрешения конфликтов;

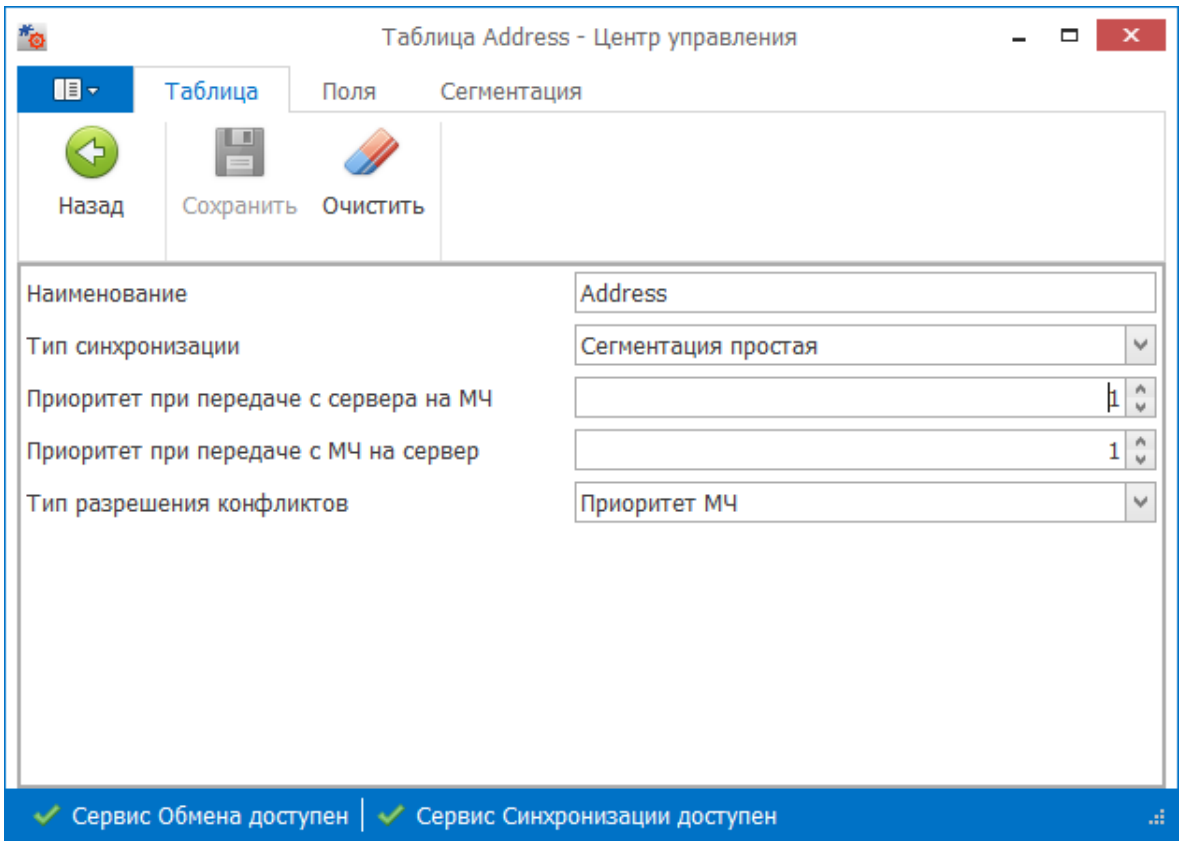


На панели инструментов размещаются следующие кнопки:

- **Добавить** – добавить таблицу.
- **Удалить** – удалить таблицу.
- **Мастер создания таблицы** – открыть инструмент для создания таблицы.

Для того чтобы перейти к просмотру или редактированию параметров той или иной таблицы, нажмите на её название. Откроется окно редактирования таблицы, содержащее следующие вкладки:

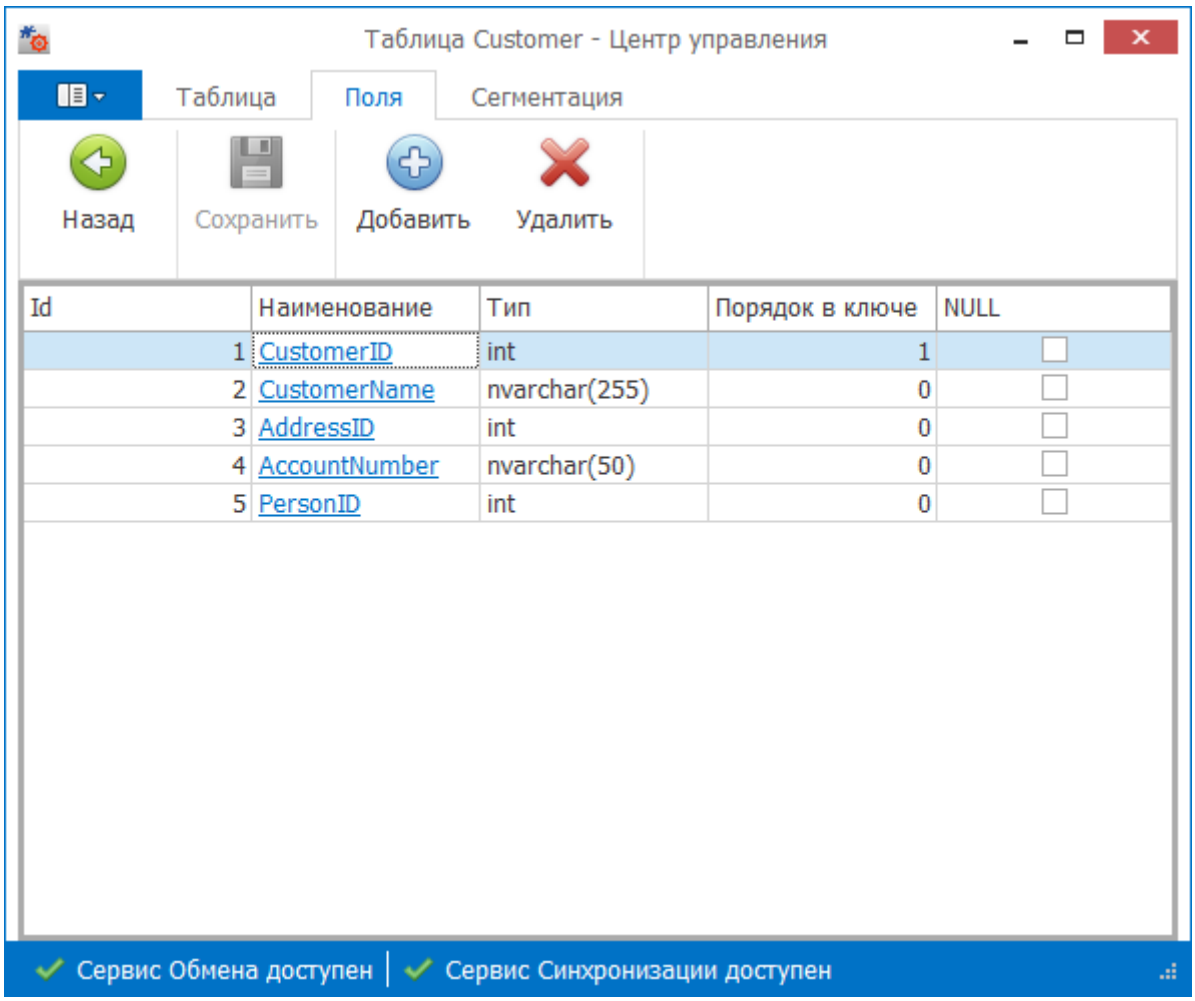
- **Таблица**;
- **Поля**;
- **Сегментация**.



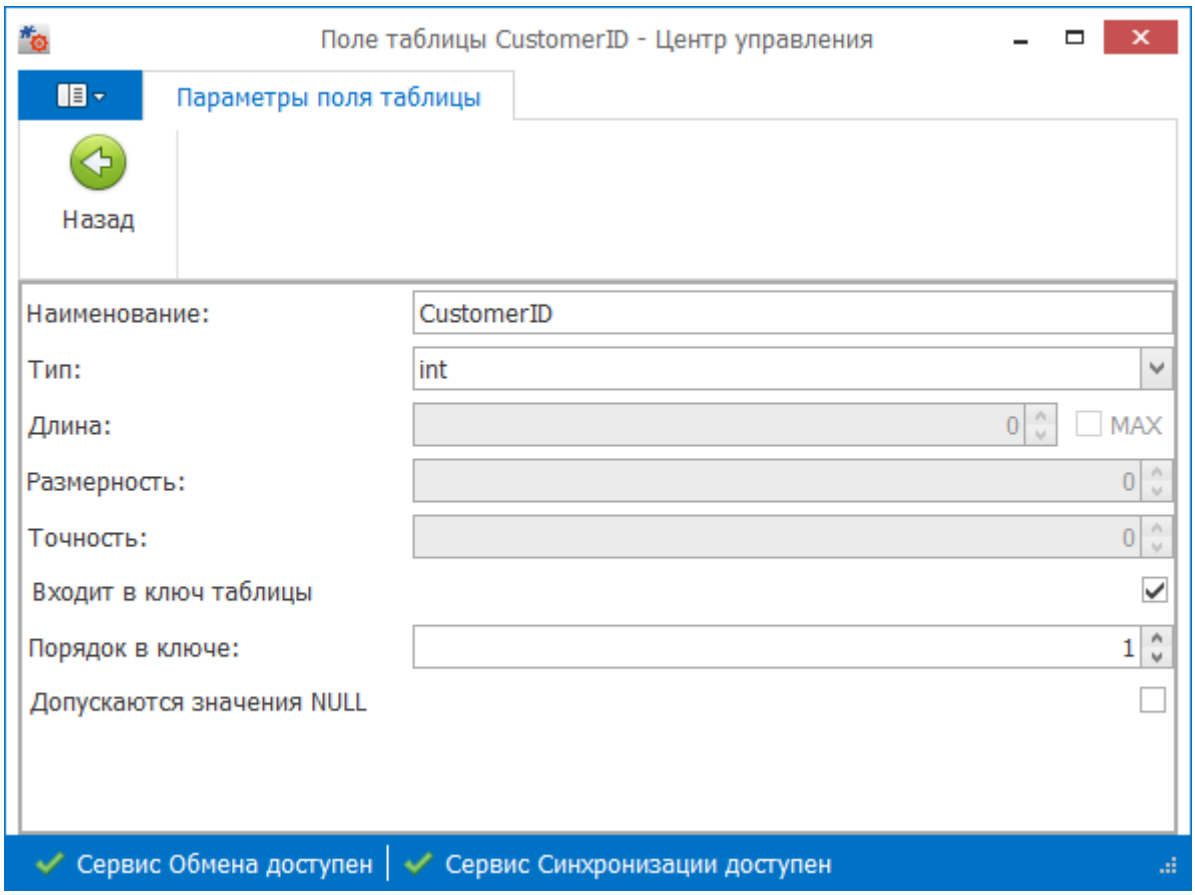
- **Таблица** – на этой вкладке редактируются параметры таблицы:
  - Наименование;
  - Тип синхронизации:
    - Сегментация простая;
    - Не синхронизируется;
    - По дате изменения;
    - По состоянию;
    - Сегментация простая;
    - Сегментация сложная.
  - Приоритет при передаче с сервера на МЧ;
  - Приоритет при передаче с МЧ на сервер;
  - Тип разрешения конфликтов:
    - Приоритет КИС;
    - Приоритет МЧ;
    - По времени: кто раньше;
    - По времени: кто позже.

Для очистки содержимого СТ предусмотрена кнопка **Очистить**. По нажатию на эту кнопку происходит удаление данных в синхронизируемой таблице и в связанных служебных таблицах.

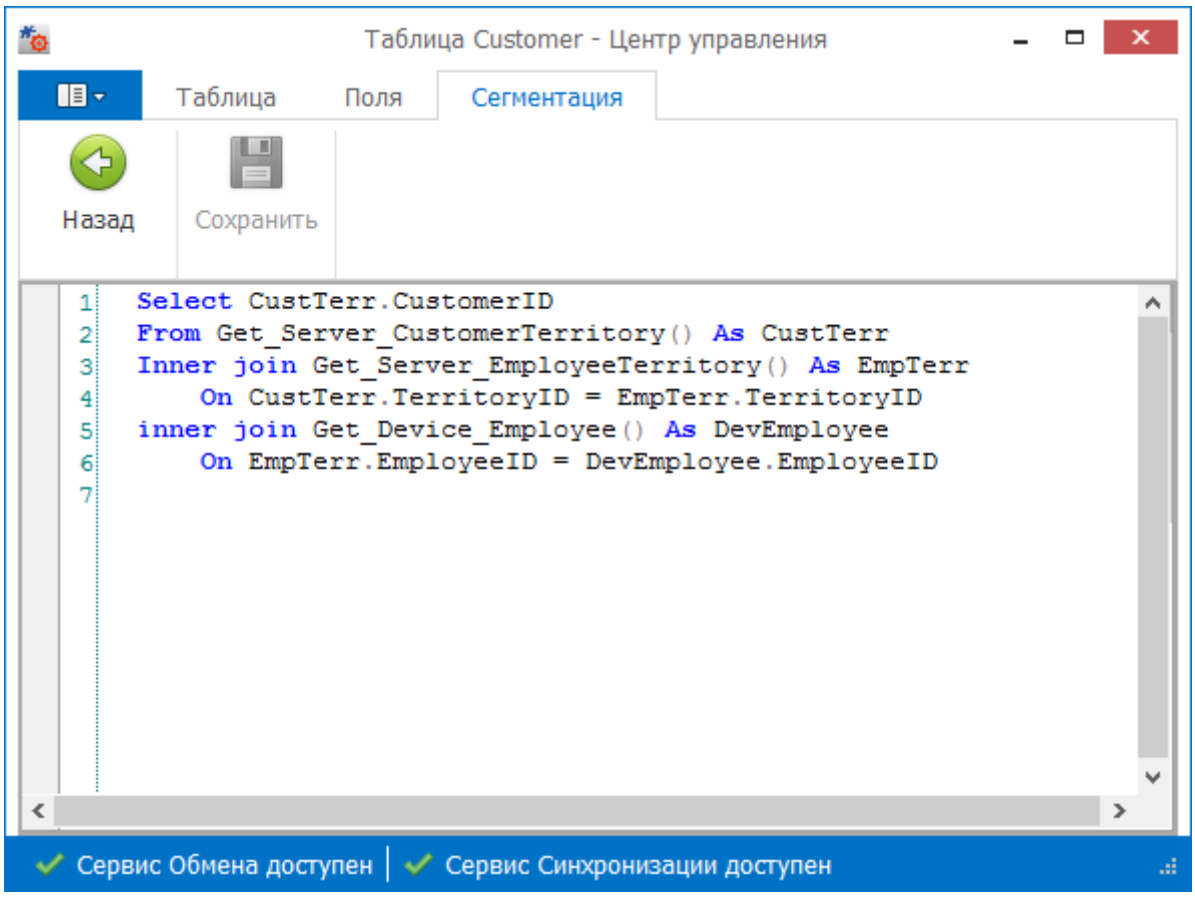
- **Поля** – на этой вкладке добавляются и редактируются поля таблицы. Для каждого поля выводится:
  - Id – порядковый номер поля;
  - Наименование – имя поля;
  - Тип – тип поля;
  - Порядок в ключе;
  - NULL.



Для редактирования параметра поля таблицы нажмите на его название. Откроется окно с параметрами.



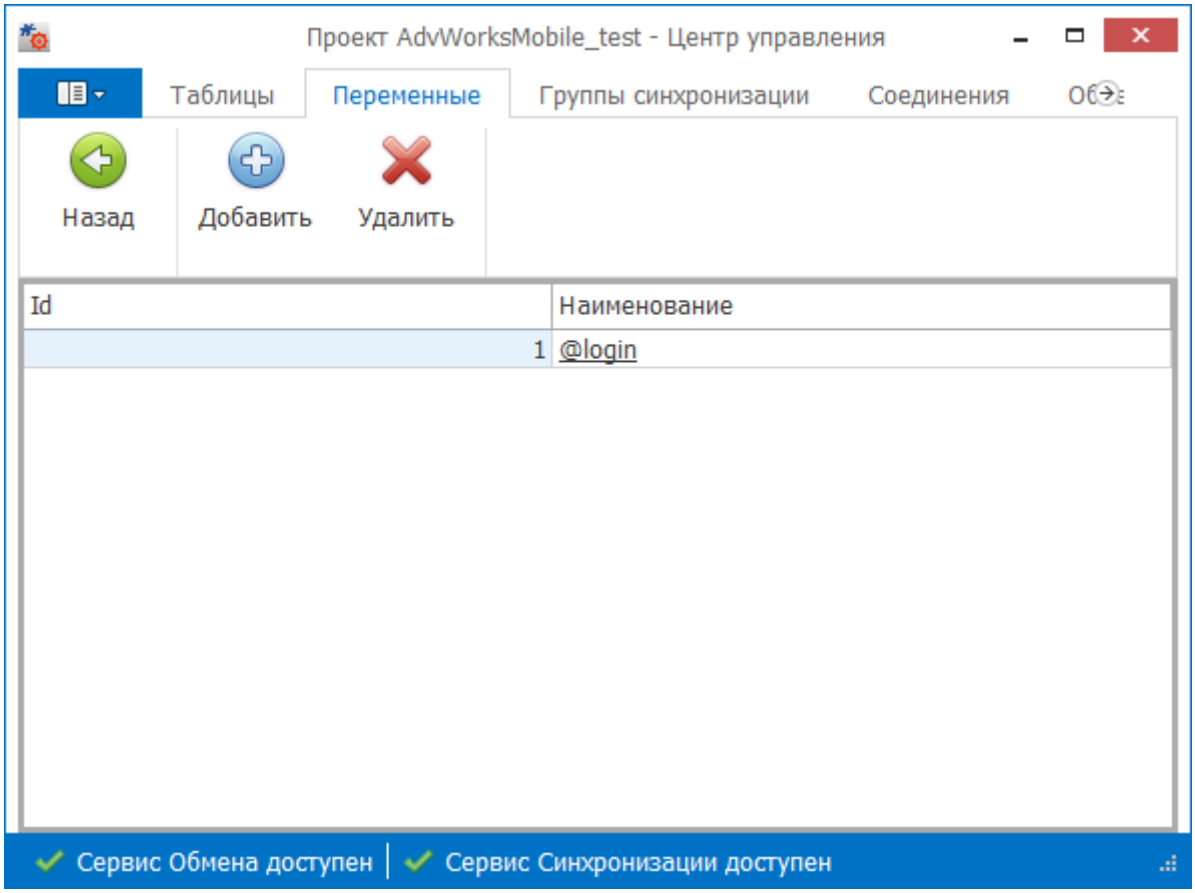
- **Сегментация** – вкладка содержит поле для ввода запроса сегментации.



Переменные

На этой вкладке выводится список переменных. Для каждой переменной выводится:

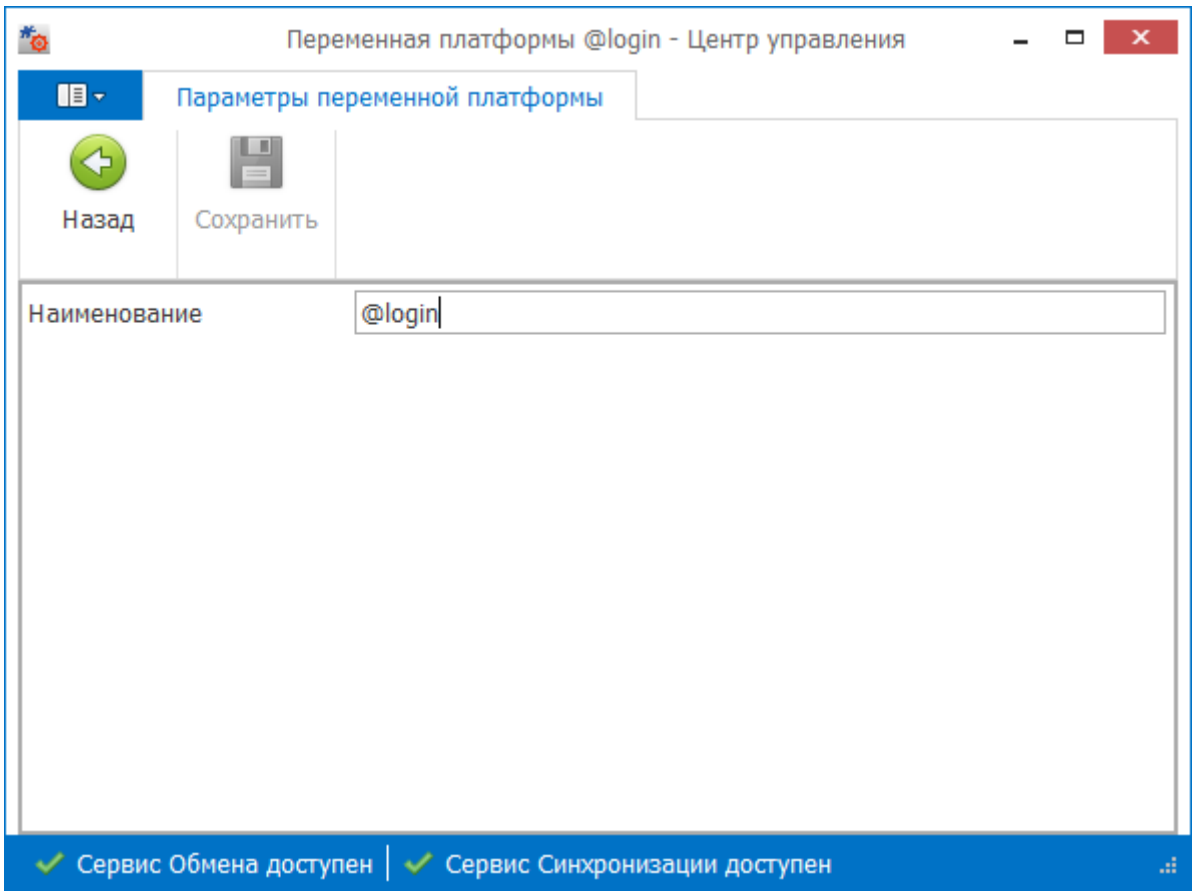
- Id – порядковый номер, который присваивается при создании переменной;
- Наименование – названия переменной.



На панели инструментов расположены следующие кнопки:

- **Добавить** – создать переменную;
- **Удалить** – удалить переменную.

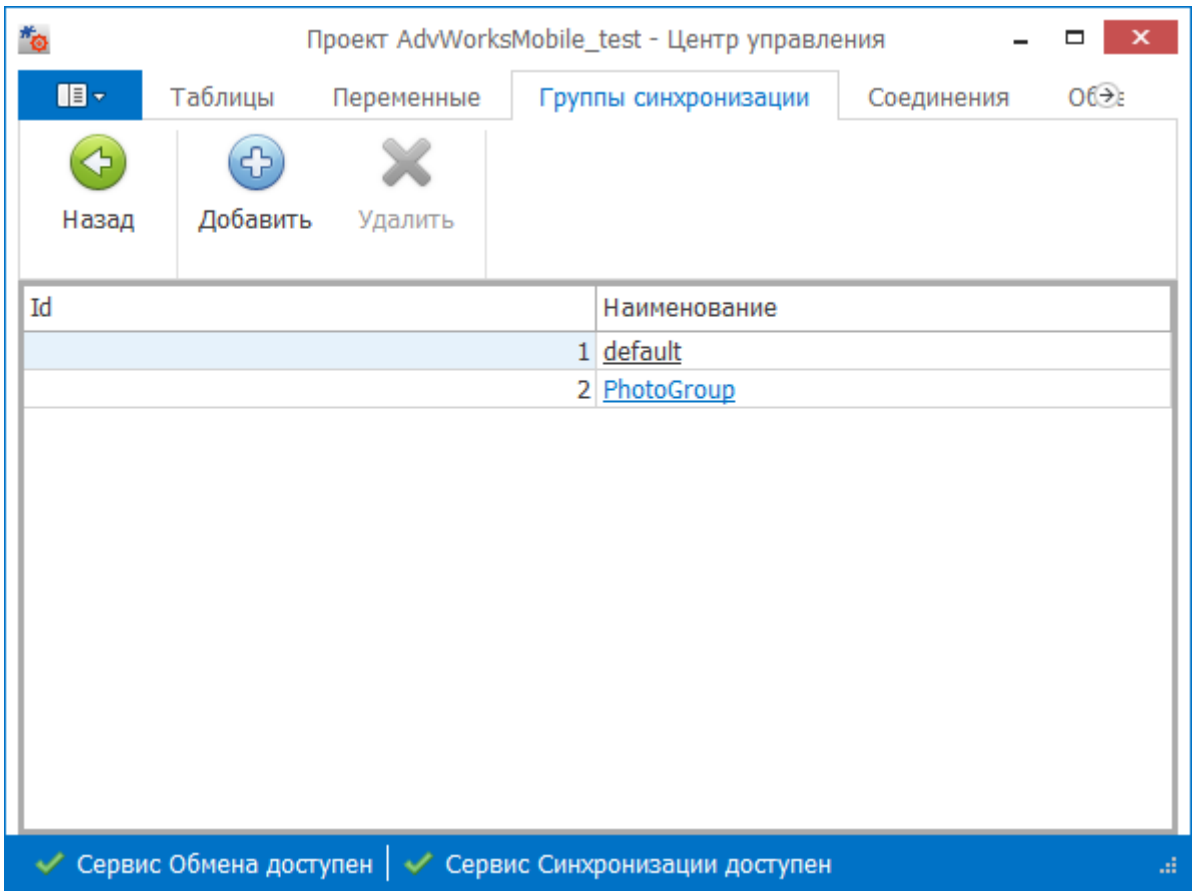
Для того чтобы изменить имя переменной, нажмите на её название. Откроется окно редактирования параметров переменной.



Группы синхронизации

На этой вкладке формируются и редактируются группы синхронизации. Для каждой группы синхронизации выводятся следующие данные:

- Id – порядковый номер группы;
- Наименование – имя группы синхронизации.

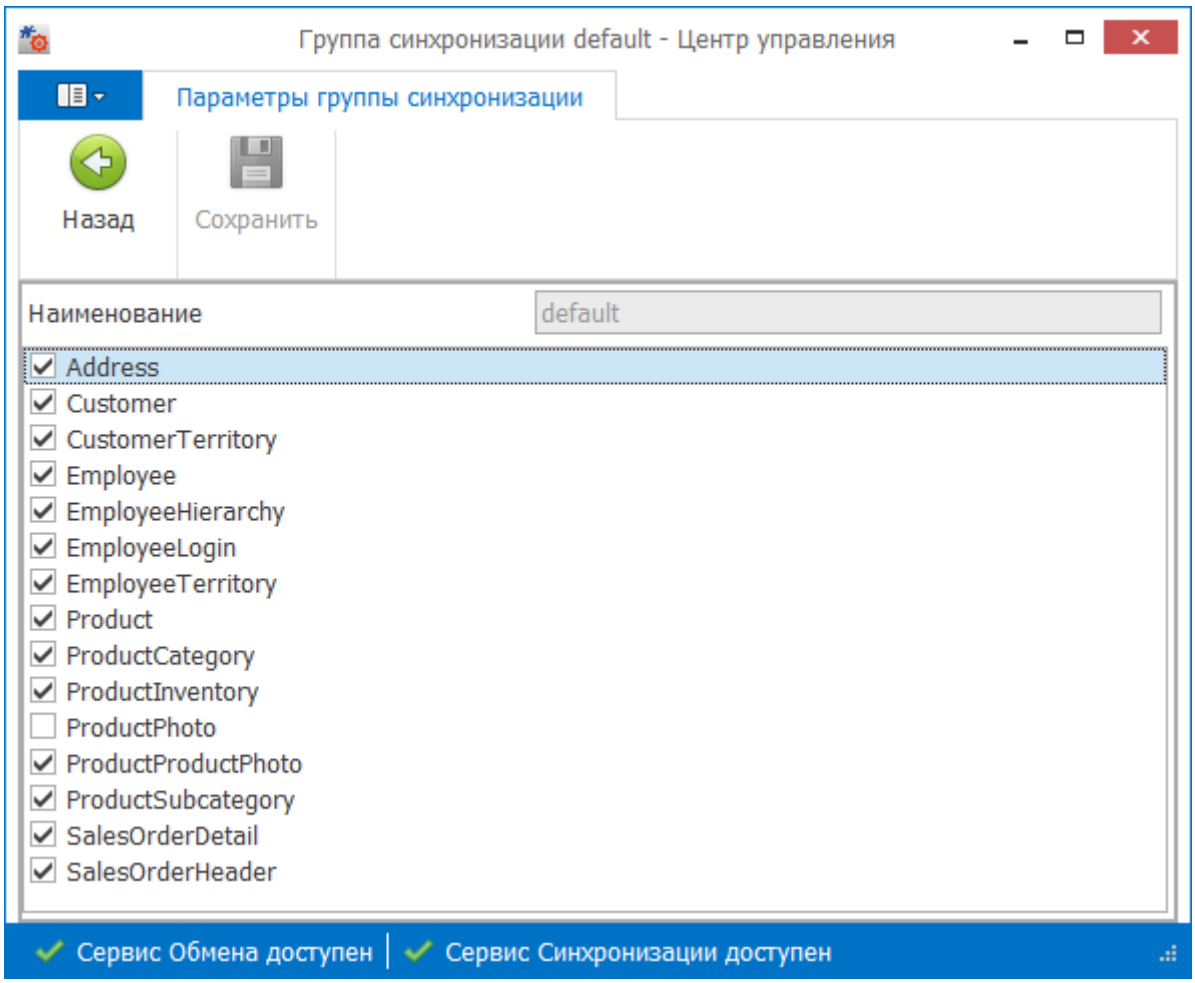


На панели инструментов доступны следующие кнопки:

- **Добавить** – сформировать новую группу синхронизации;
- **Удалить** – удалить группу синхронизации.

Примечание: группа default не может быть удалена.

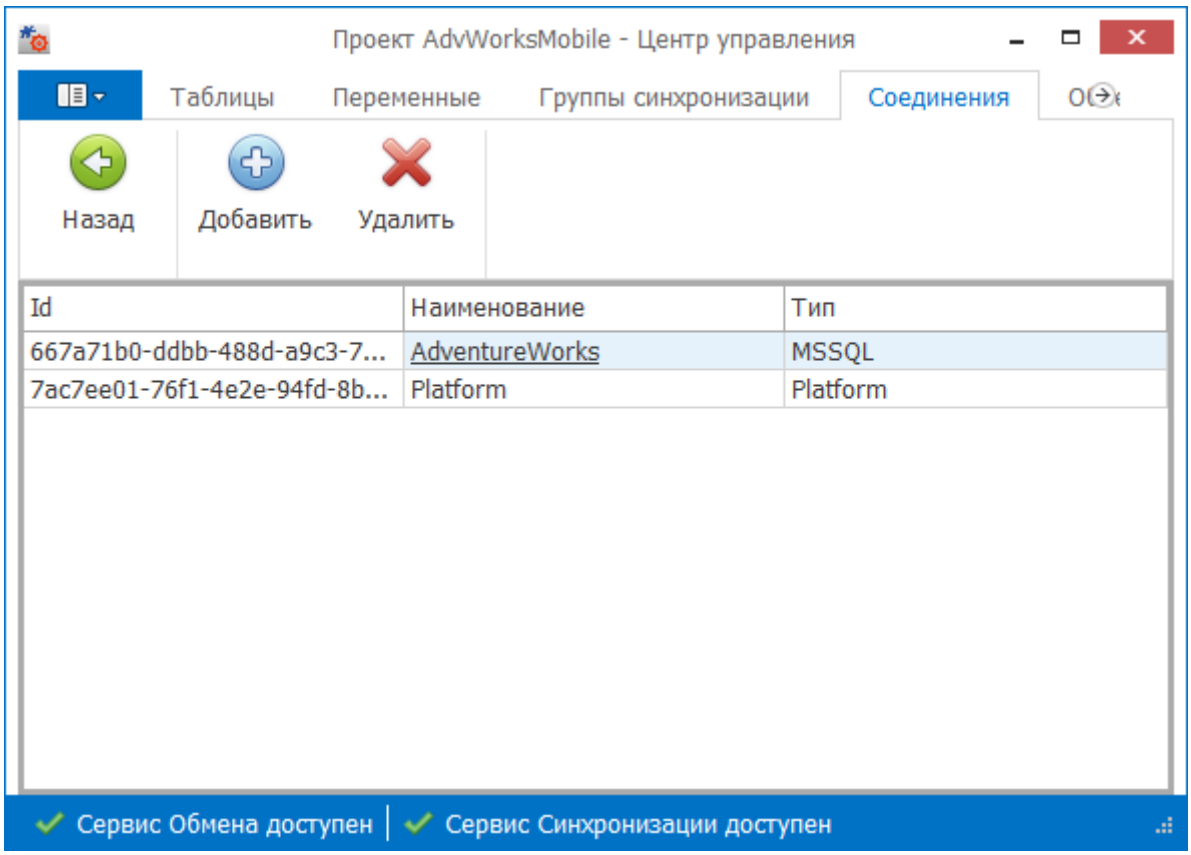
Для того чтобы отредактировать группу, нажмите на её название. Откроется окно редактирования параметров группы синхронизации. Флагами отмечены те синхронизируемые таблицы, которые входят в группу.



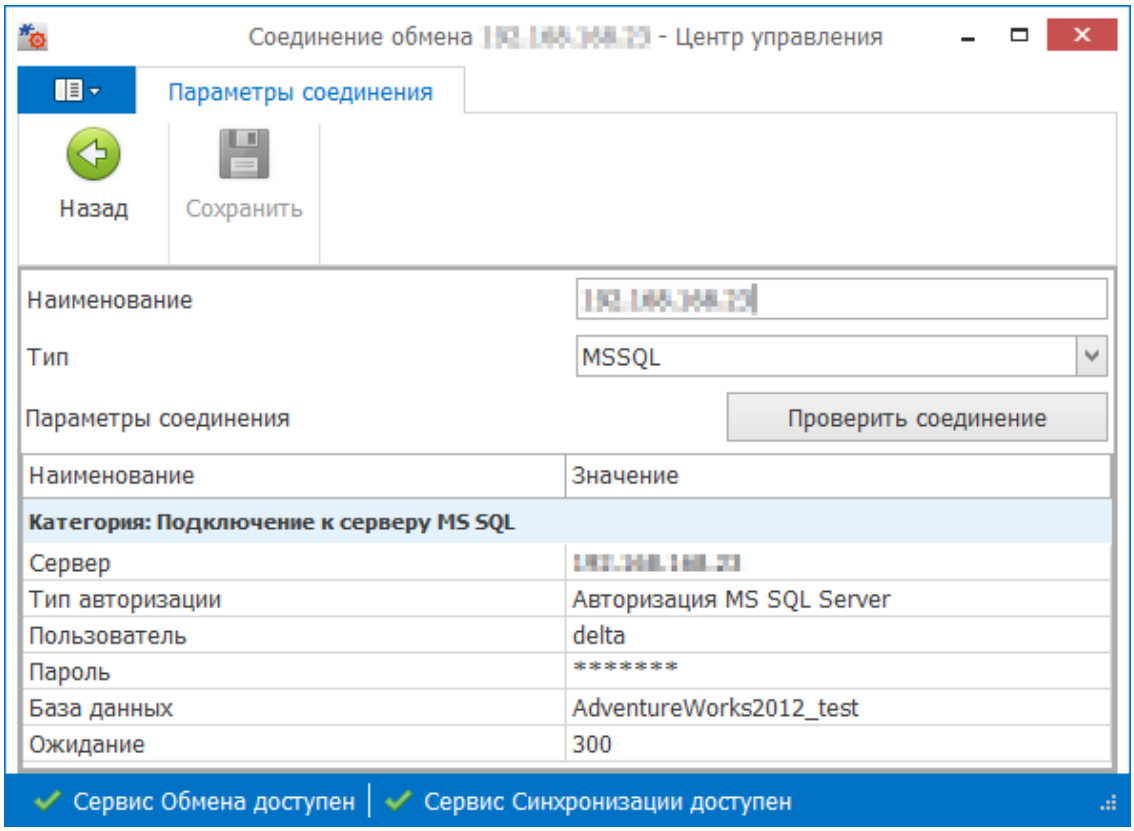
Соединения

На этой вкладке настраиваются соединения с источником и приёмником данных. Для каждого соединения выводятся следующие данные:

- Id – идентификатор соединения, присваиваемый ему при создании;
- Наименование – имя соединения;
- Тип – тип соединения.



Параметры соединения с источником данных редактируются по нажатию на наименование соединения. При этом открывается вкладка **Параметры соединения**.

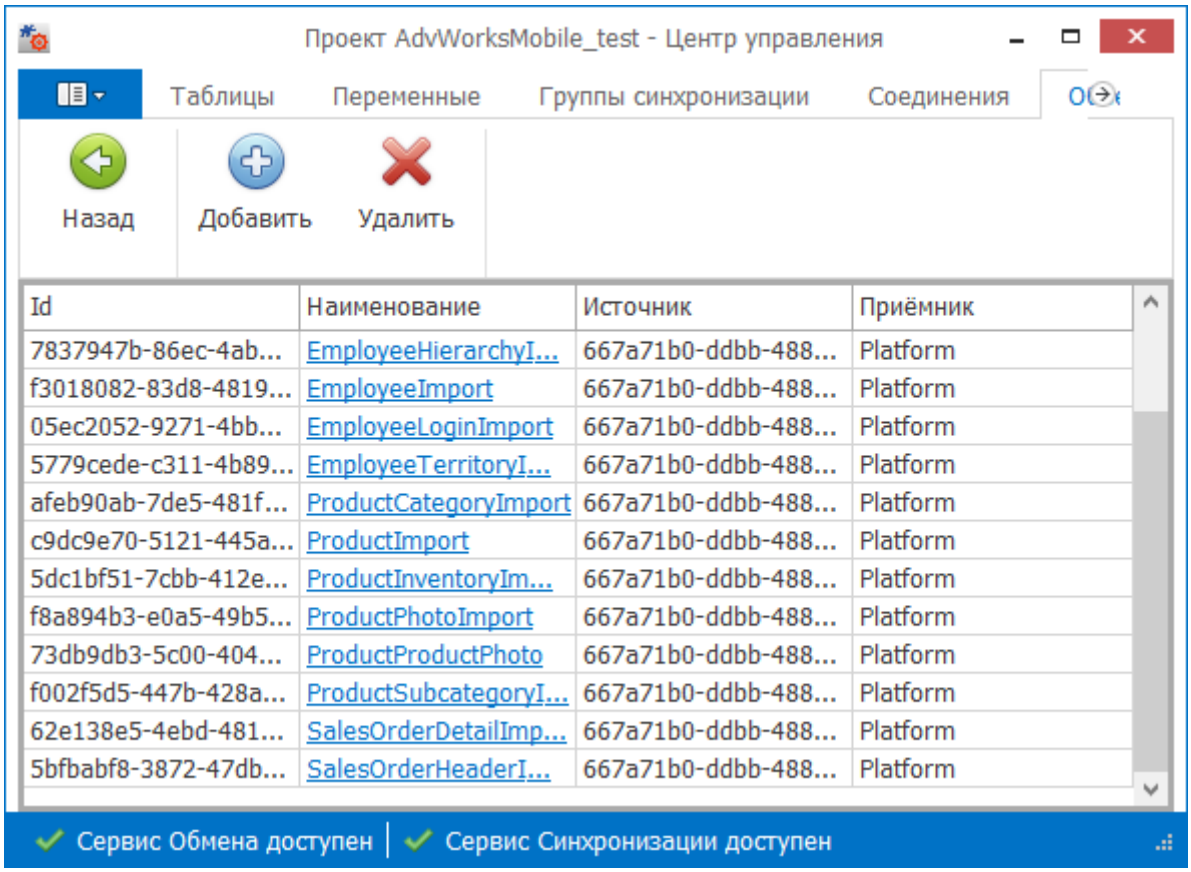


Объекты обмена

На этой вкладке настраиваются объекты обмена, которые необходимы для организации передачи данных из таблиц источника в таблицы приёмника данных. Для каждой таблицы формируется объект обмена для импорта данных и объект для экспорта данных.

По каждому из объектов обмена выводится следующая информация:

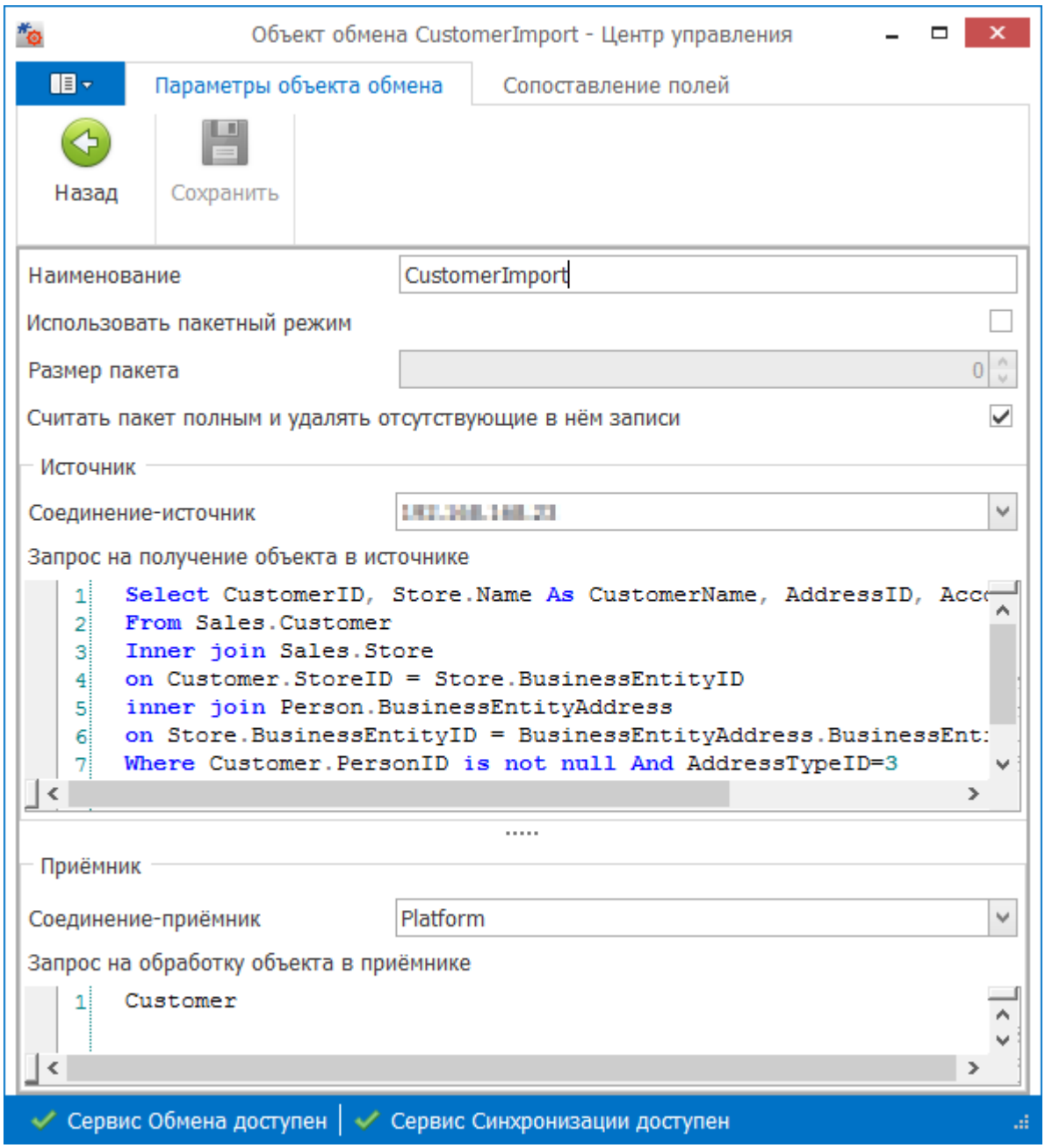
- Id – идентификатор объекта, формирующийся автоматически;
- Наименование – имя объекта;
- Источник – соединение-источник данных;
- Приёмник – соединение-приёмник данных.



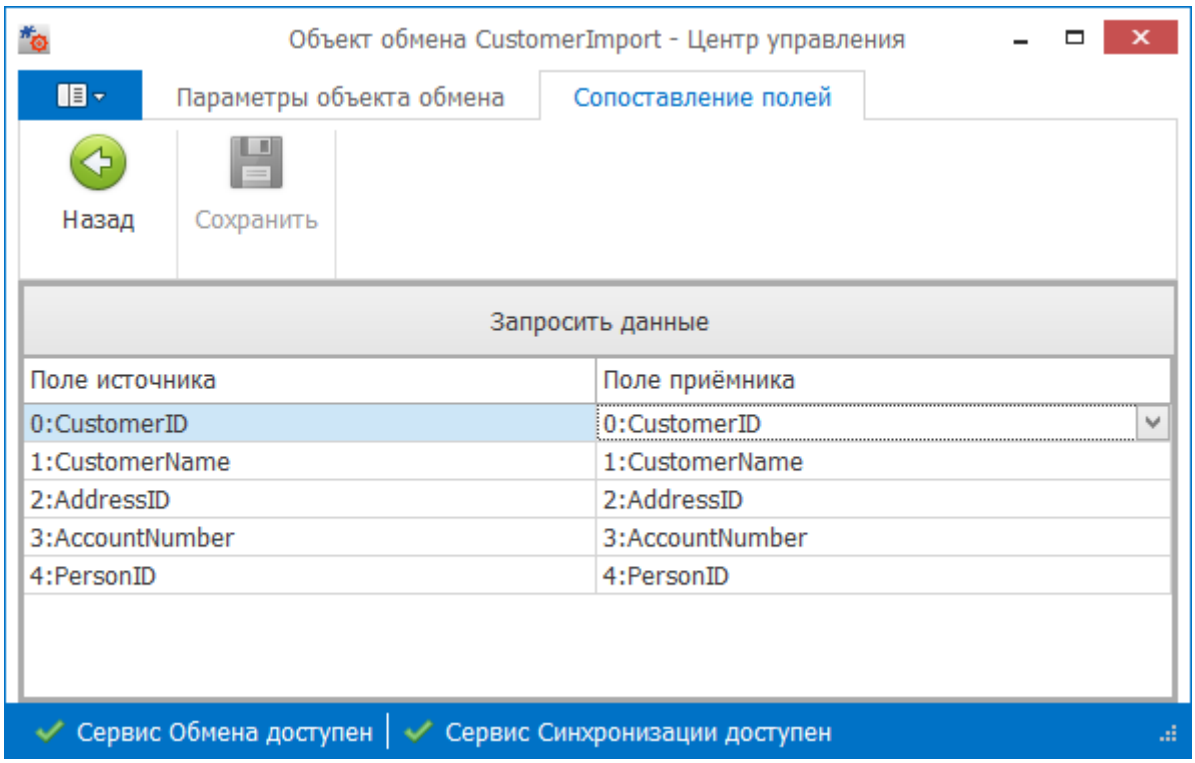
Для того чтобы отредактировать объект обмена, нажмите на его наименование. Откроется окно редактирования параметров объекта, содержащее вкладки:

- **Параметры объекта обмена**;
- **Сопоставление полей**.

На вкладке **Параметры объекта обмена** указываются соединение-источник и соединение-приёмник, а также сопутствующие сведения. В запросах на получение и обработку объекта реализована подсветка синтаксиса. Синтаксис определяется плагином.



На вкладке **Сопоставление полей** устанавливается связь между полями таблицы источника и полями таблицы приёмника. По нажатию на кнопку **Запросить данные** загружаются поля таблицы источника. В столбце **Поле приёмника** для каждого поля источника выбирается поле приёмника.

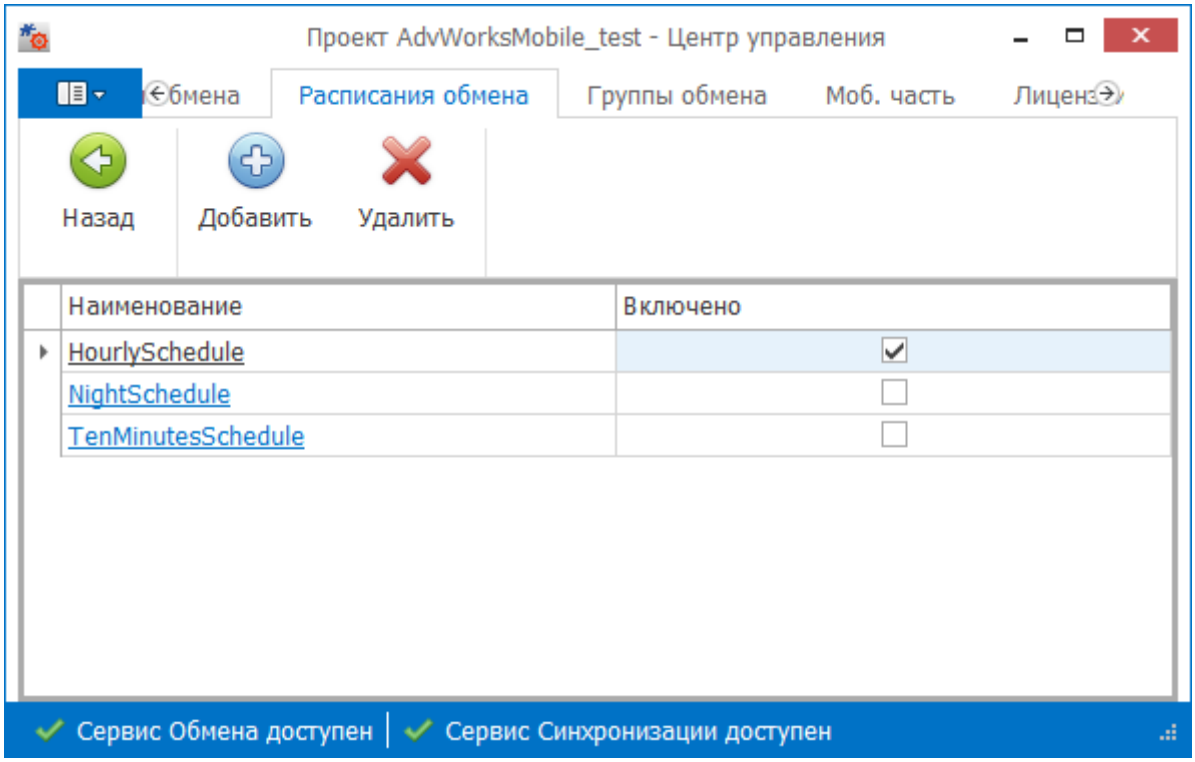


Расписания обмена

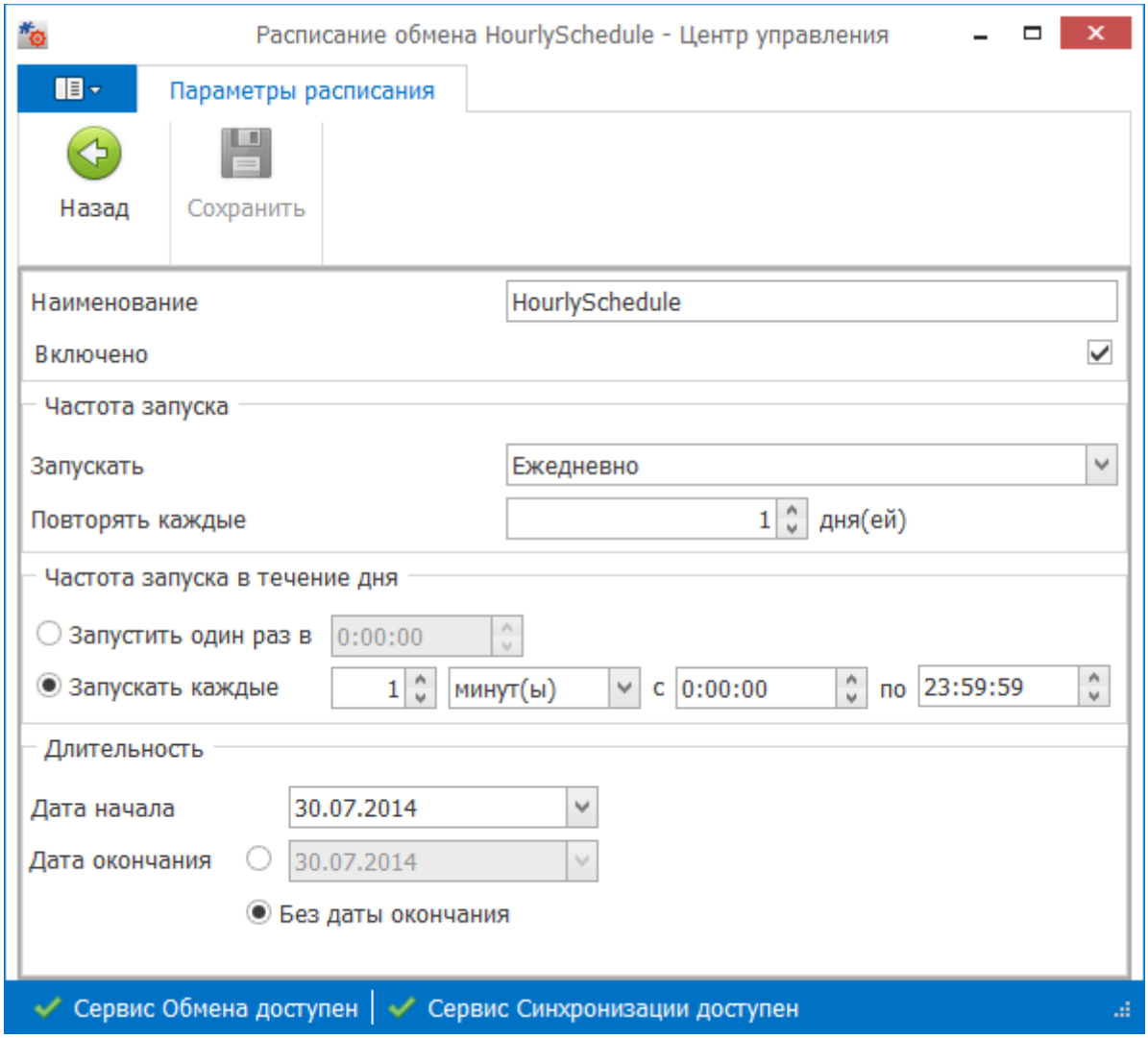
На этой вкладке формируются расписания обмена, для запуска обмена с BackEnd в определенное время.

Для каждого расписания выводится:

- Наименование – название расписания;
- Включено – активно расписание или нет.



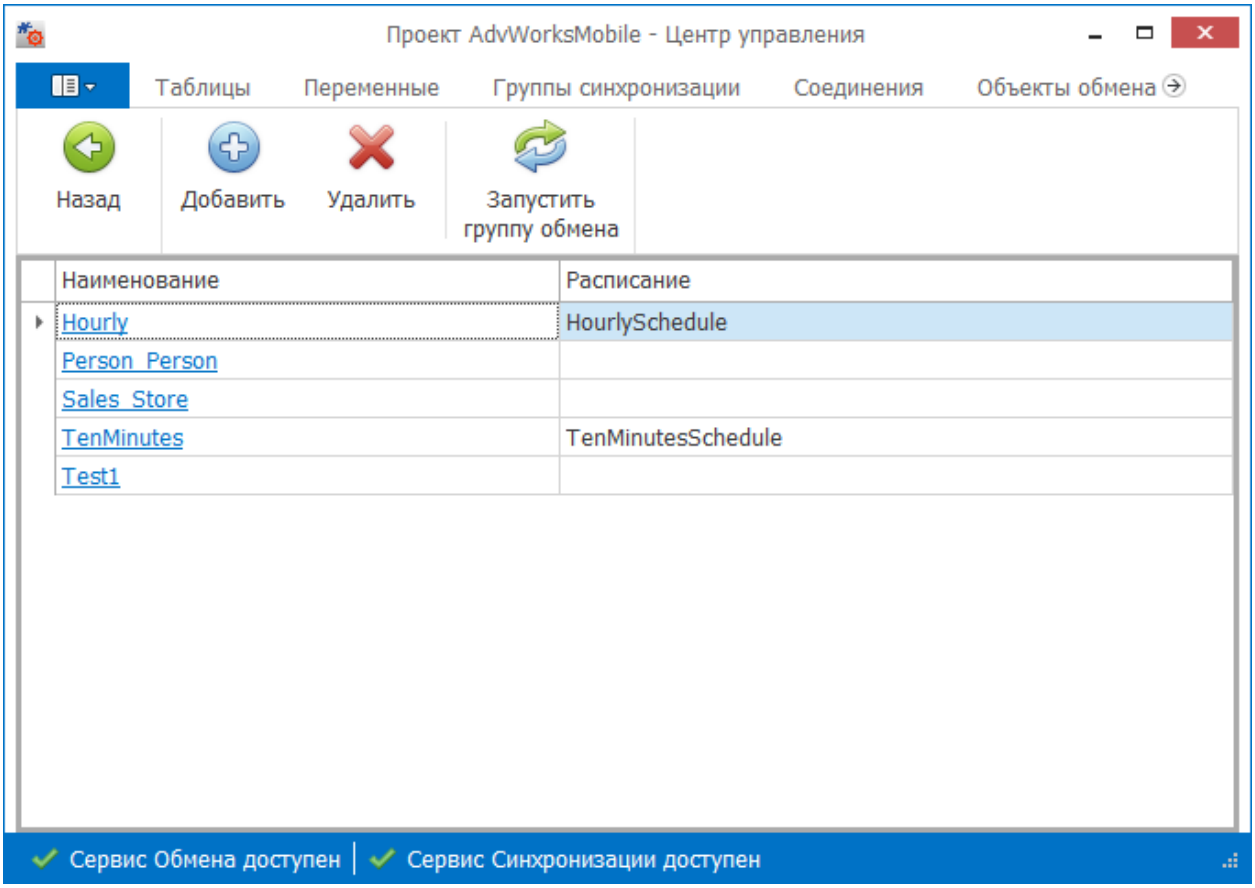
Для того чтобы отредактировать расписание, необходимо нажать на его название. Откроется окно с параметрами расписания.



Группы обмена

На этой вкладке формируются группы обмена. Для каждой группы выводится следующая информация:

- Наименование – имя группы;
- Расписание – установленное группе расписание обмена.



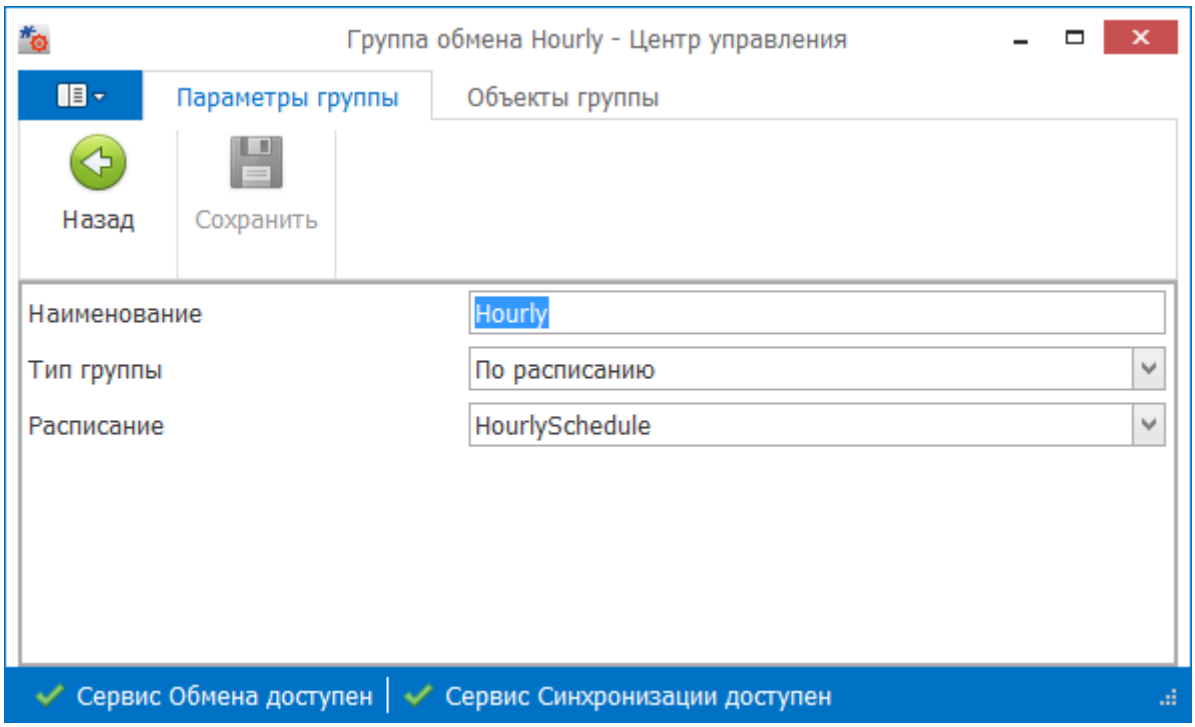
На панели инструментов среди прочих служебных кнопок расположена кнопка **Запустить группу обмена**, предназначенная для запуска обмена вручную. По нажатию на эту кнопку запускается обмен в выделенной группе обмена. В случае, если обмен в той или иной группе уже запущен по расписанию, и в этот момент пользователь пытается запустить обмен вручную в той же группе, обмен будет блокирован, а на экране появится сообщение о том, что обмен в этой группе уже запущен. И наоборот, если запущен обмен вручную, то обмен по расписанию будет блокирован для одной и той же группы обмена.

Для того чтобы отредактировать группу, нажмите на ее название. Откроется окно редактирования параметров группы обмена, содержащее вкладки:

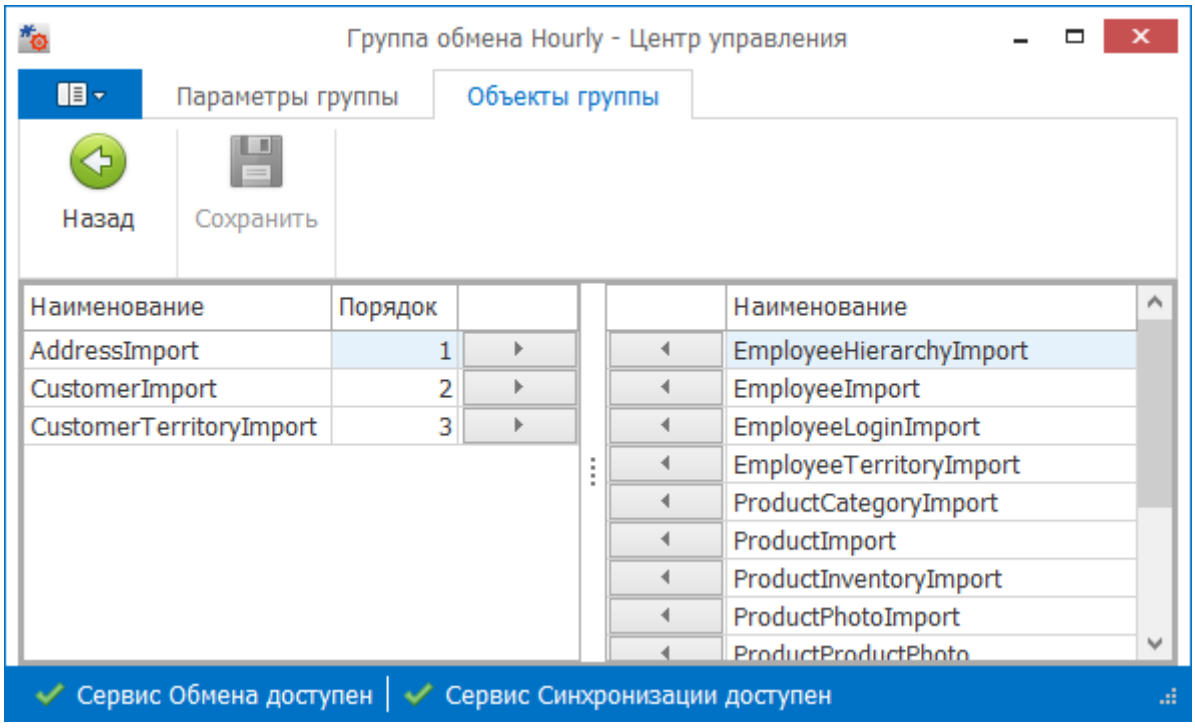
- **Параметры группы**;
- **Объекты группы**.

На вкладке **Параметры группы** указываются:

- Наименование – имя группы;
- Тип группы – инициация обмена по событию или по расписанию;
- Расписание – выбор расписания для группы обмена.

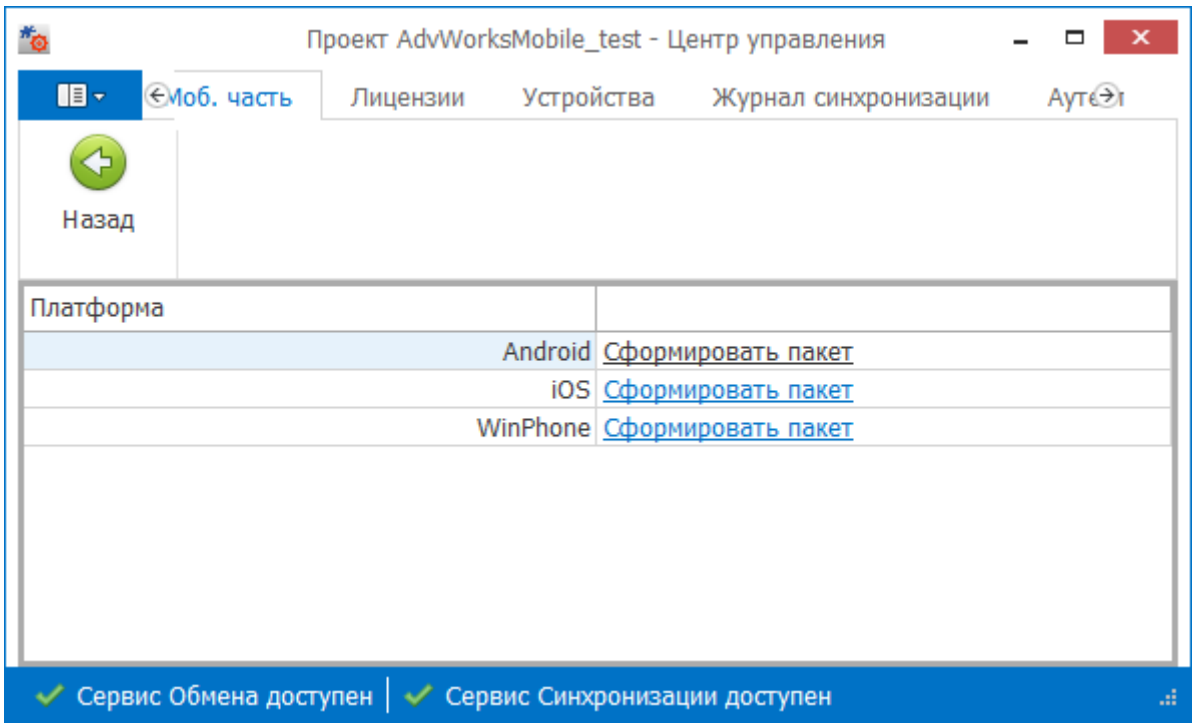


На вкладке **Объекты группы** формируется состав группы – включаются или исключаются объекты обмена.



Моб. часть

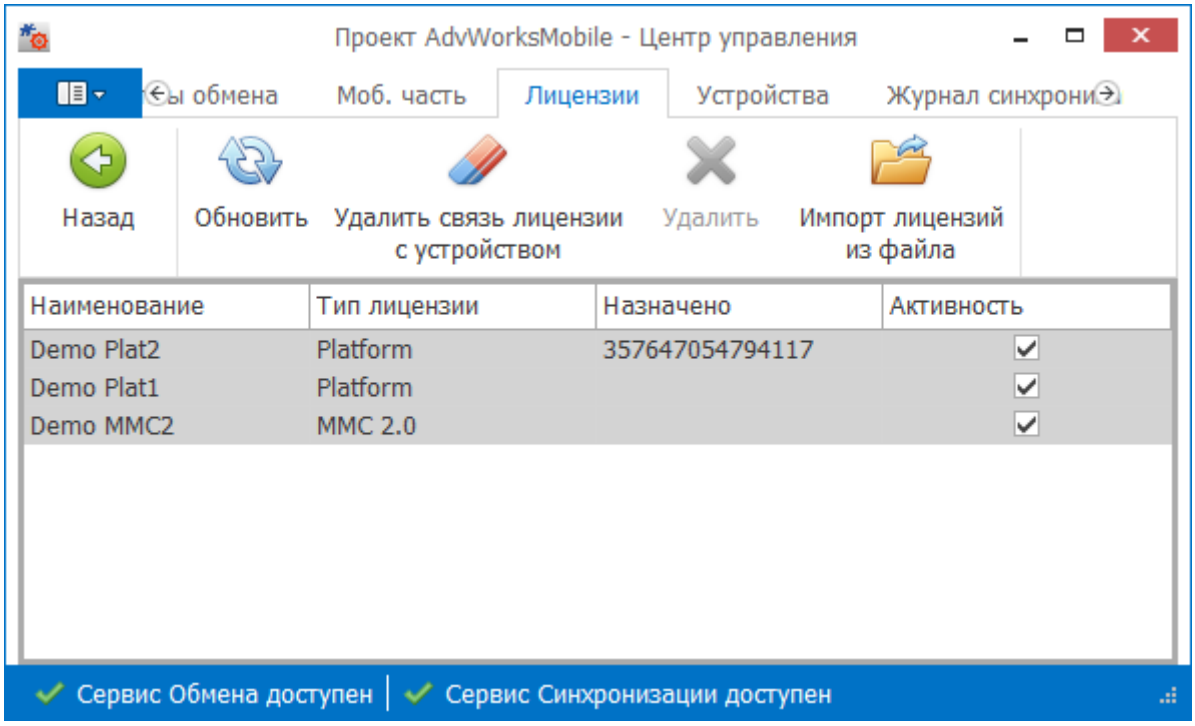
На этой вкладке формируется архив, который содержит файл с метаданными и набор библиотек, необходимых для разработки мобильного приложения под выбранную операционную систему. Для каждой мобильной платформы отдельная команда **Сформировать пакет**.



Лицензии

На этой вкладке выводится перечень лицензий текущего проекта и их свойства:

- Наименование;
- Тип лицензии;
- Назначено – кем занята лицензия;
- Активность – флаг активности.



На панели инструментов доступны следующие действия:

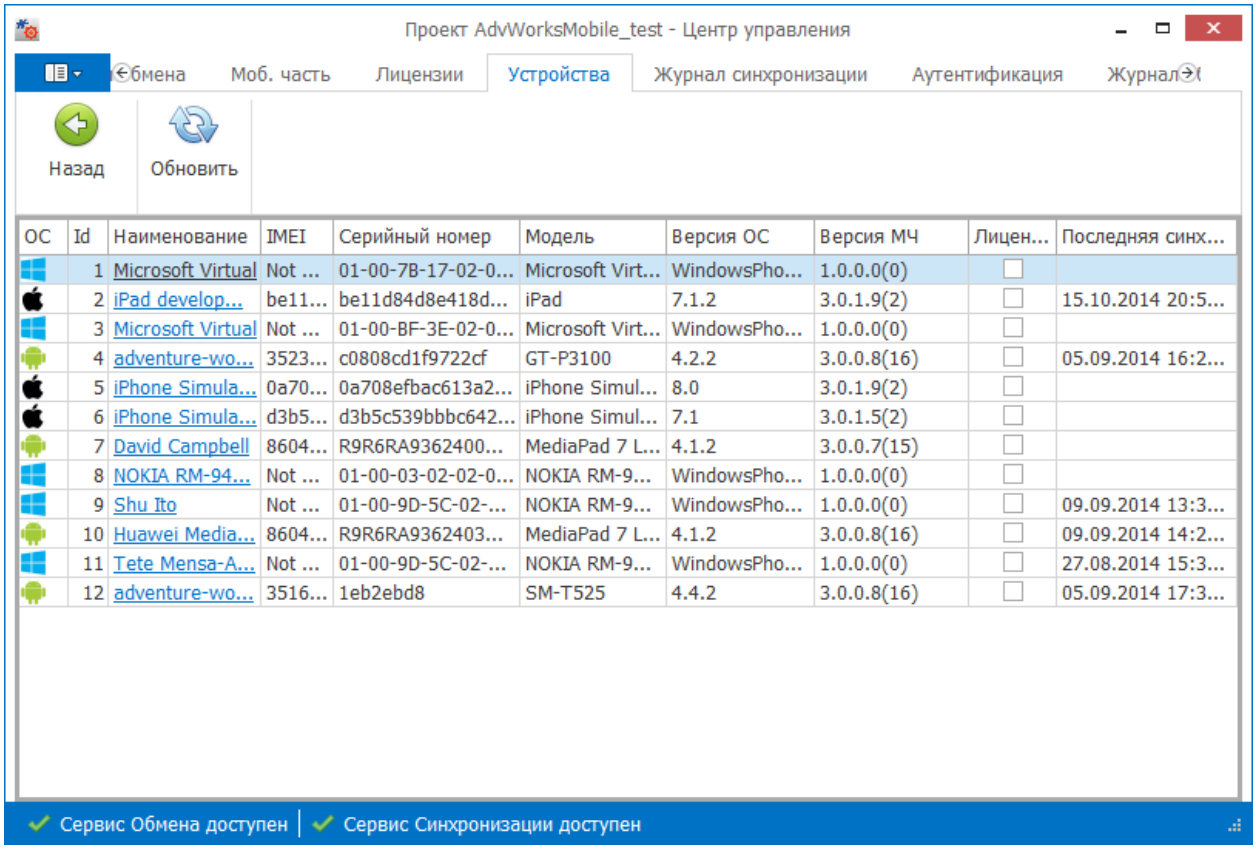
- **Обновить** – обновить список лицензий;
- **Удалить связь лицензии с устройством** – очистить лицензию;
- **Удалить** – удалить лицензию;
- **Импорт лицензий из файла** – загрузка лицензий из файла.

Устройства

На этой вкладке выводится список всех мобильных устройств, проводивших синхронизацию с платформой, к текущему моменту.

По каждому из устройств выводятся следующие данные:

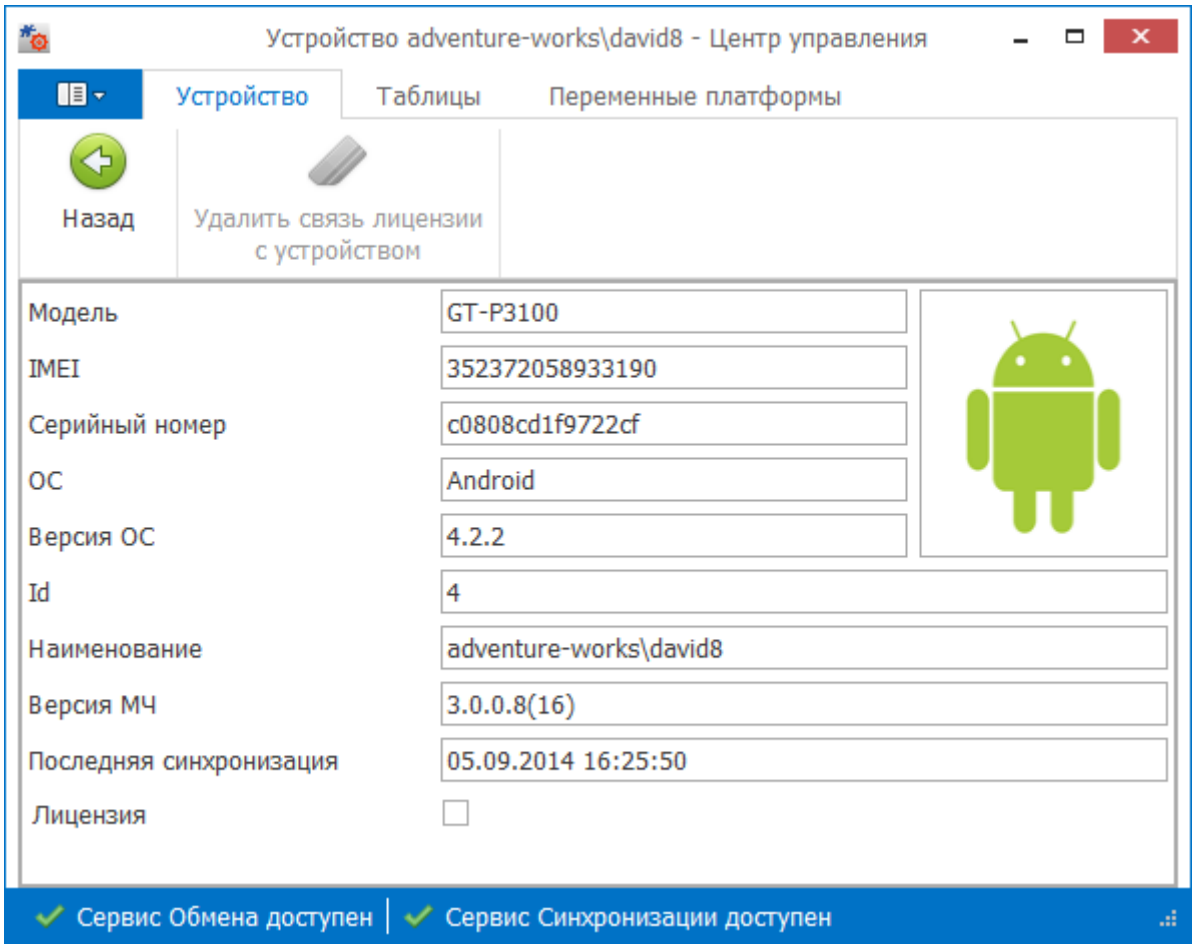
- ОС – пиктограмма операционной системы, установленной на мобильном устройстве;
- Id – порядковый номер устройства;
- IMEI – IMEI устройства;
- Серийный номер – серийный номер устройства;
- Модель – модель устройства;
- Версия ОС – версия операционной системы на устройстве;
- Версия МЧ – версия мобильной части на устройстве;
- Лицензия – разрешено ли устройству синхронизироваться с платформой (то есть зарегистрировано ли устройство в системе);
- Последняя синхронизация – дата и время последней синхронизации устройства.



Для просмотра сведений по тому или иному устройству, нажмите на его наименование. Откроется окно подробной информации по устройству, содержащее вкладки:

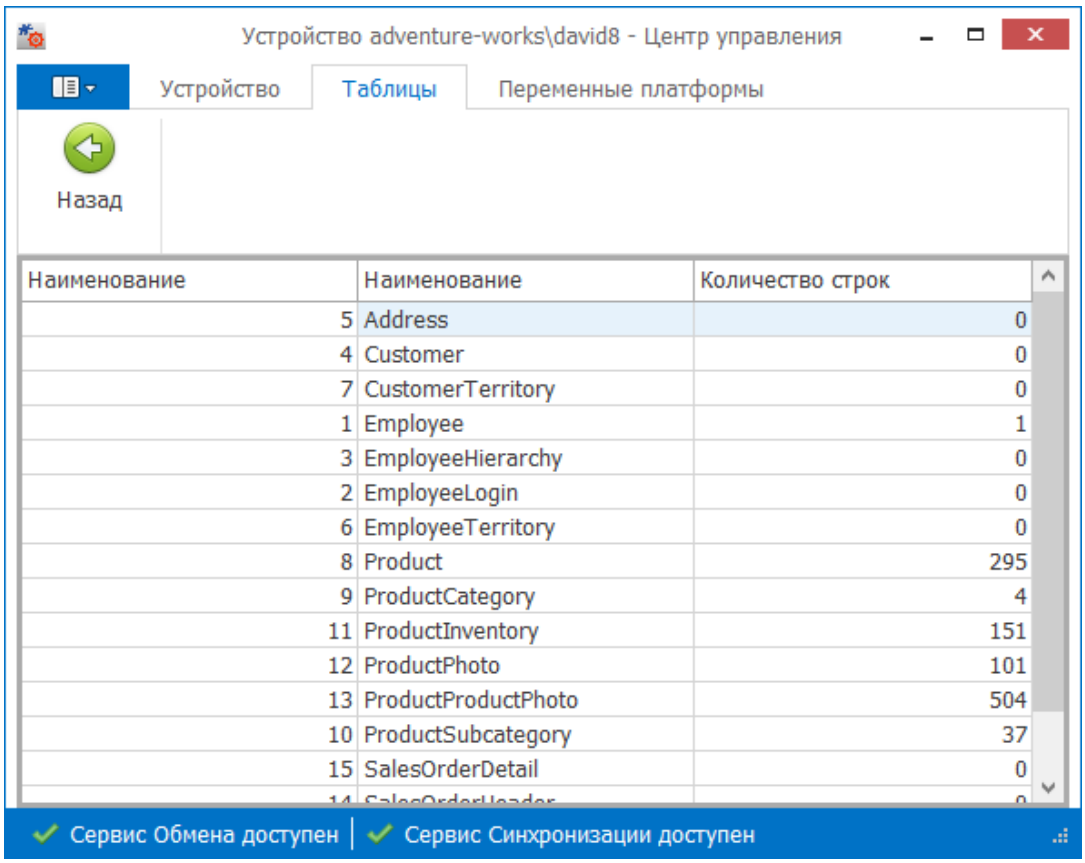
- **Устройство**;
- **Таблицы**;
- **Переменные платформы**.

На вкладке **Устройство** в режиме просмотра выводятся те же сведения, что и в списке устройств.

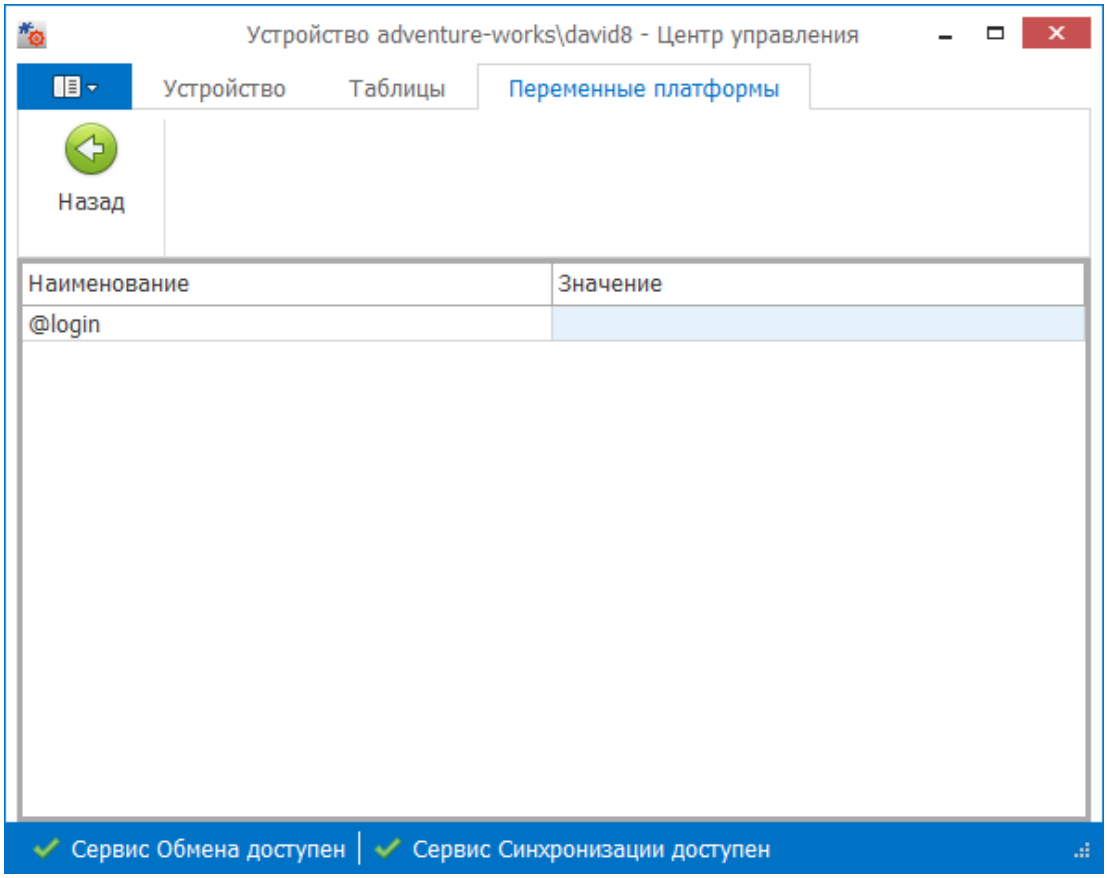


На вкладке **Таблицы** выводится список синхронизируемых таблиц, по каждой из которых отображаются следующие данные:

- Id – идентификатор таблицы;
- Наименование – имя синхронизируемой таблицы;
- Количество строк – количество строк этой таблицы в БД устройства.



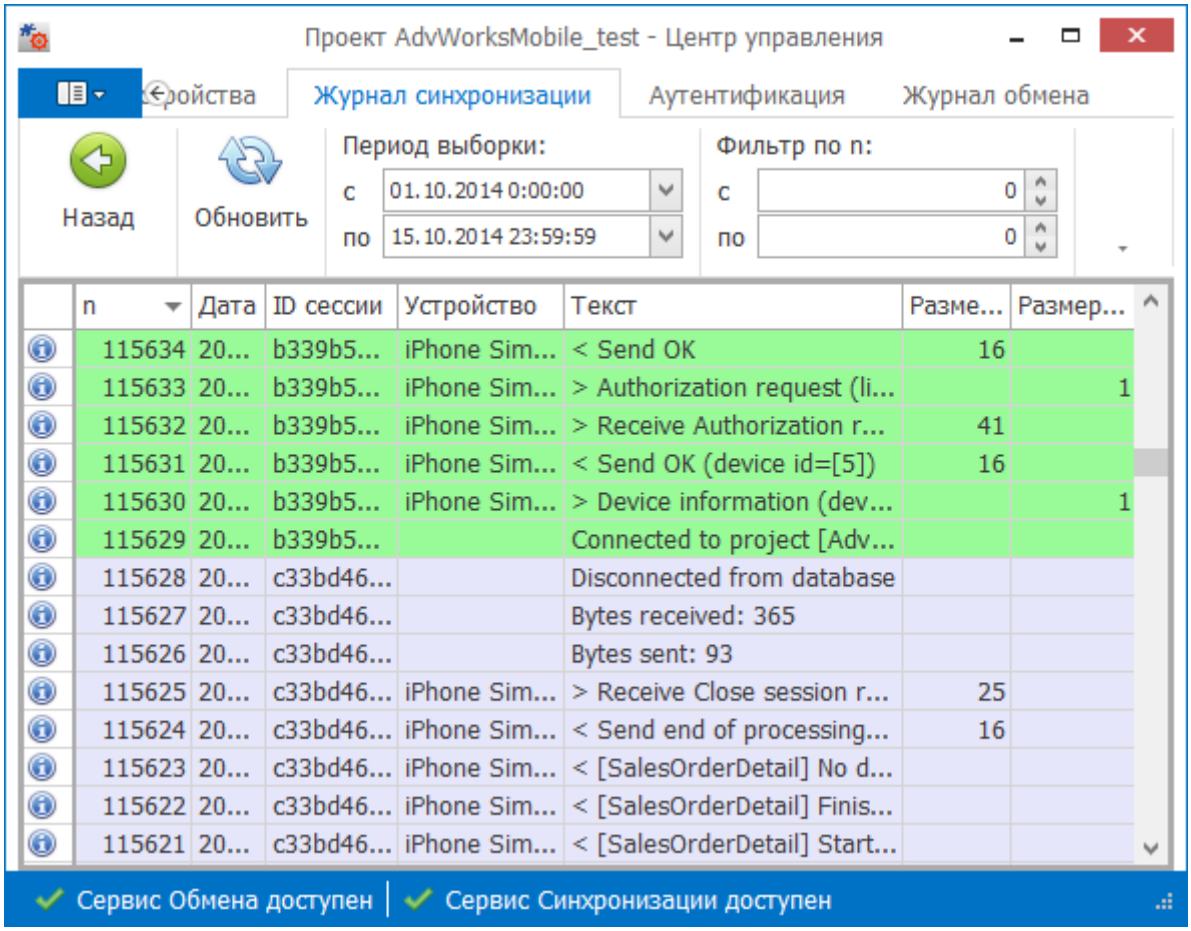
На вкладке **Переменные платформы** выводится имя и значение переменной для этого устройства.



Журнал синхронизации

В журнале синхронизации содержатся записи о событиях Сервиса синхронизации. Каждая сессия синхронизации выделяется отдельным цветом. Журнал выводится в виде таблицы со следующими столбцами:

- n – порядковый номер записи;
- Дата – дата и время формирования записи;
- ID сессии – идентификатор сессии;
- Устройство – имя устройства;
- Текст;
- Размер пакета (байт);
- Размер блока (строк).

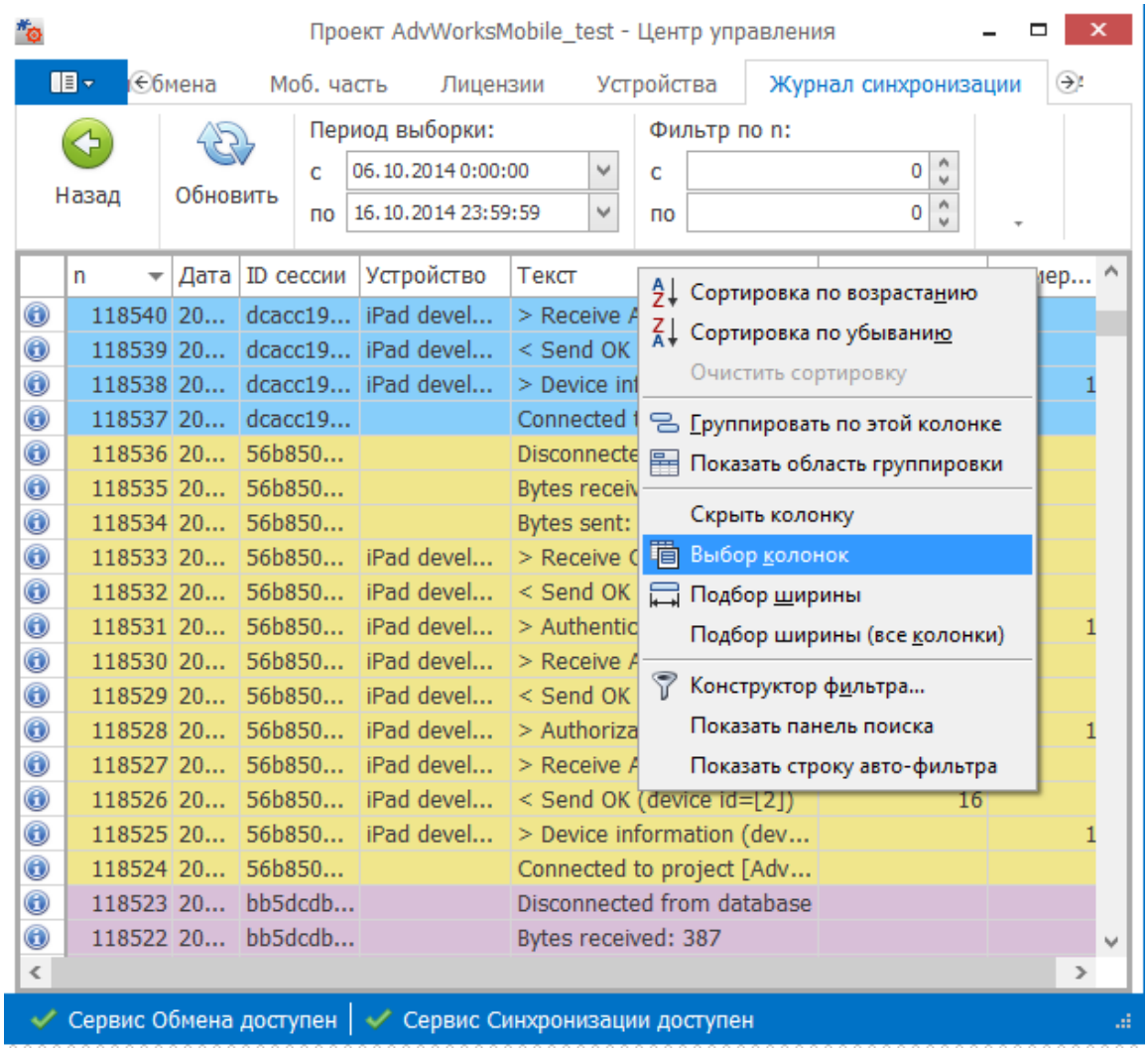


На панели инструментов доступны следующие фильтры:

- Период выборки – для отбора записей по дате и времени;
- Фильтр по n – для отбора записей по порядковому номеру;
- Устройство – для фильтрации записей по устройству;
- Только записи об ошибках – для отбора записей только об ошибках.

При первичном открытии вкладки журнал пуст, для получения лога необходимо установить значения фильтров и нажать на кнопку **Обновить**.

Дополнительные возможности содержатся в контекстном меню шапки таблицы. Например, для отображения или скрытия того или иного столбца предусмотрена функция **Выбор колонок**.



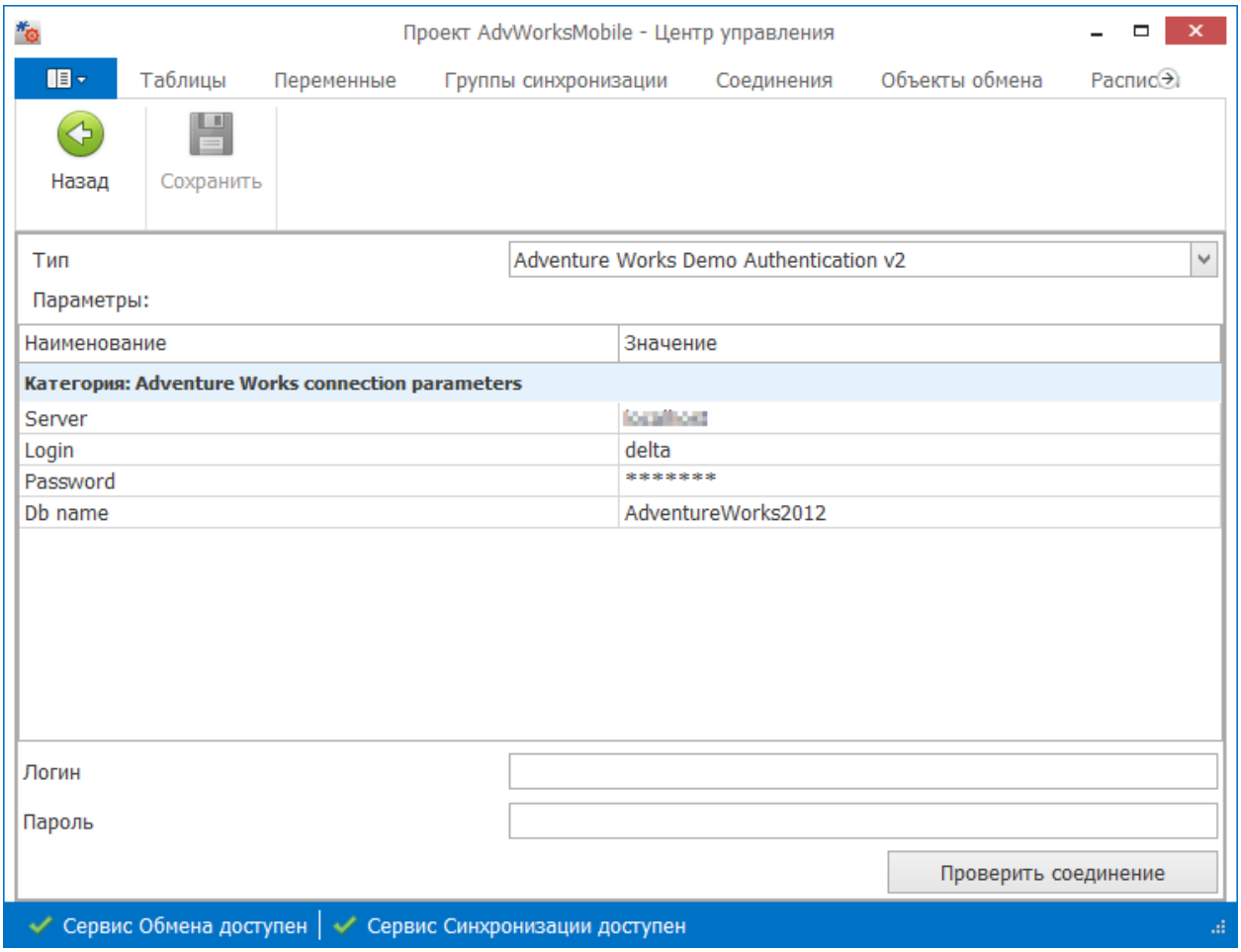
Аутентификация

Эта вкладка предназначена для настройки аутентификации проекта. На вкладке **Аутентификация** выбирается плагин аутентификации для текущего проекта и проверяется работа связки плагин-сервер аутентификации-логин/пароль пользователя мобильного приложения.

На вкладке располагаются поля:

- Тип – в этом поле выбирается плагин аутентификации из списка, полученного от СС. Список содержит доступные для данного проекта методы аутентификации. Выбранный в списке плагин определяет, как будет проходить аутентификацию каждый пользователь мобильного приложения.
- Параметры плагина –в рамках этих параметров пользователь указывает параметры работы плагина (например, для плагина Adventure Works Demo Authentication):
    - Server – SQL Server, на котором расположена БД AdventureWorks.
    - Login – логин для подключения к серверу.
    - Password – пароль для подключения к серверу.
    - Db name – имя БД AdventureWorks.

- Область проверки, содержащая поля «Логин» и «Пароль». В эти поля вводится любая пара логин/пароль, под которой будет проходить аутентификацию пользователь мобильного приложения.



По нажатию на кнопку **Проверить соединение** происходит проверка введенных логина/пароля через выбранный плагин с введенными настройками плагина. Цель этой проверки – полностью смоделировать процесс аутентификации пользователя мобильного приложения и убедиться, что при заданных настройках процесс работает.

Журнал обмена

В журнале обмена отображаются записи о событиях Сервиса обмена.

Журнал имеет древовидную структуру. Изначально записи группируются по событиям, например, по событию запуска расписания обмена. К каждому событию привязаны группы обмена, соответственно, событие вызывает обработку нескольких групп обмена. Таким образом, первый уровень журнала – это связка события и группы обмена. Второй уровень – это данные об ОО, входящих в группу обмена. Записи третьего уровня - данные об операциях в рамках объекта обмена - отображаются в журнале, только если в настройках СО установлен флаг «Расширенное логирование в БД».

По каждой записи первого уровня в журнале выводится следующая информация:

- Пиктограмма, обозначающая, успешно ли прошел обмен:

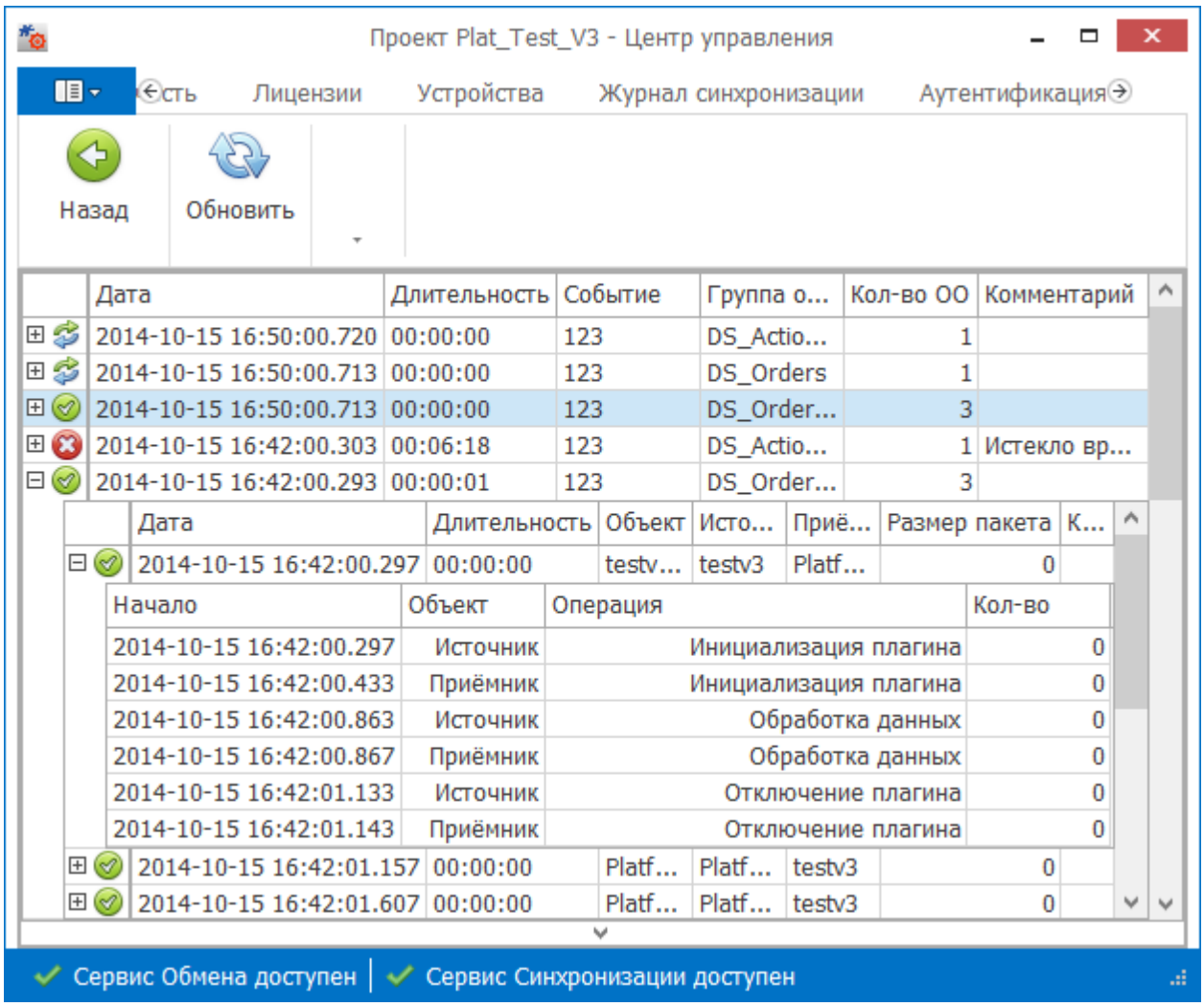
    -  - обмен в процессе;
    -  - обмен успешен;
    -  - обмен не прошел;
- Дата;
- Длительность – длительность эпизода обмена;
- Событие – инициатор обмена, например, расписание;
- ID группы обмена;
- Группа обмена – группа обмена, привязанная к расписанию обмена;
- Кол-во ОО – количество объектов обмена;
- Комментарий.

По каждой записи второго уровня выводится следующая информация:

- Дата;
- Длительность;
- Объект;
- Источник;
- Приёмник;
- Размер пакета;
- Комментарий.

По каждой записи третьего уровня выводится следующая информация:

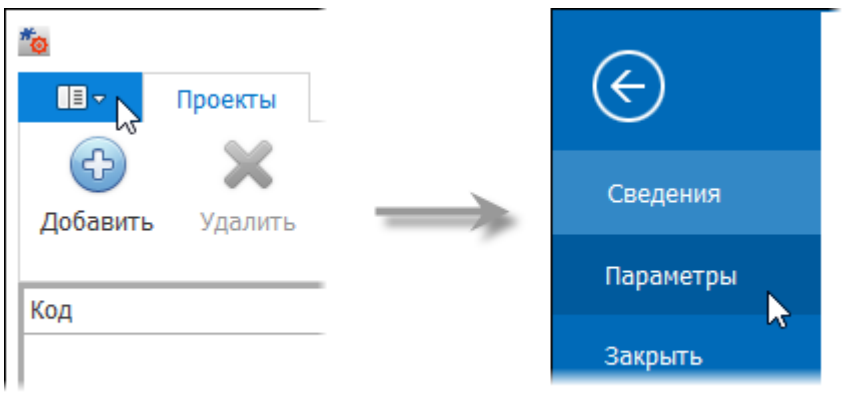
- Начало;
- Объект – источник или приёмник;
- Операция:
    - Инициализация плагина;
    - Инициализация объекта;
    - Обработка данных;
    - Освобождение объекта;
    - Отключение плагина.
- Кол-во.



Настройка Сервиса обмена

Центр управления взаимодействует с Сервисом обмена по протоколу TCP/IP для работы с плагинами обмена (выполнение запросов к источнику, подсветка запросов и т.д.). По умолчанию адрес, по которому ЦУ обращается к СО – localhost. В случае, если ЦУ и СС расположены на разных машинах, в ЦУ необходимо указывать адрес машины, на которой установлен СО.

Для настройки параметров Сервиса обмена откройте меню приложения и перейдите в раздел **Параметры**.



В группе настроек **Сервис обмена** доступны следующие настройки:

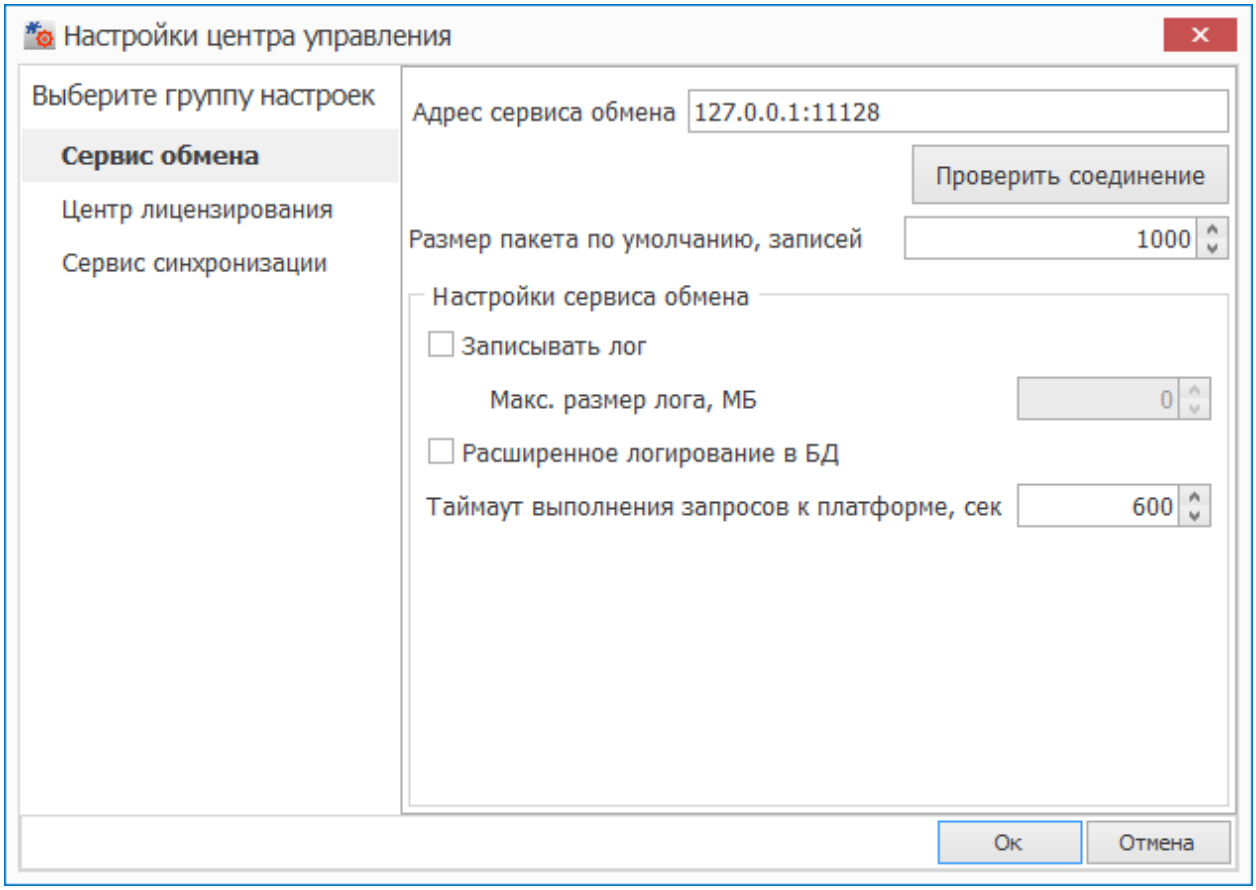
Адрес сервиса обмена - для ввода адреса и порта Сервиса обмена.

Размер пакета по умолчанию, записей - данные в СО передаются пакетами, а не целиком. В этом параметре задается количество записей в пакете для передачи в Сервис обмена.

Записывать в лог – записывать лог в файл или нет. Если флаг отсутствует, то лог-файл не формируется.

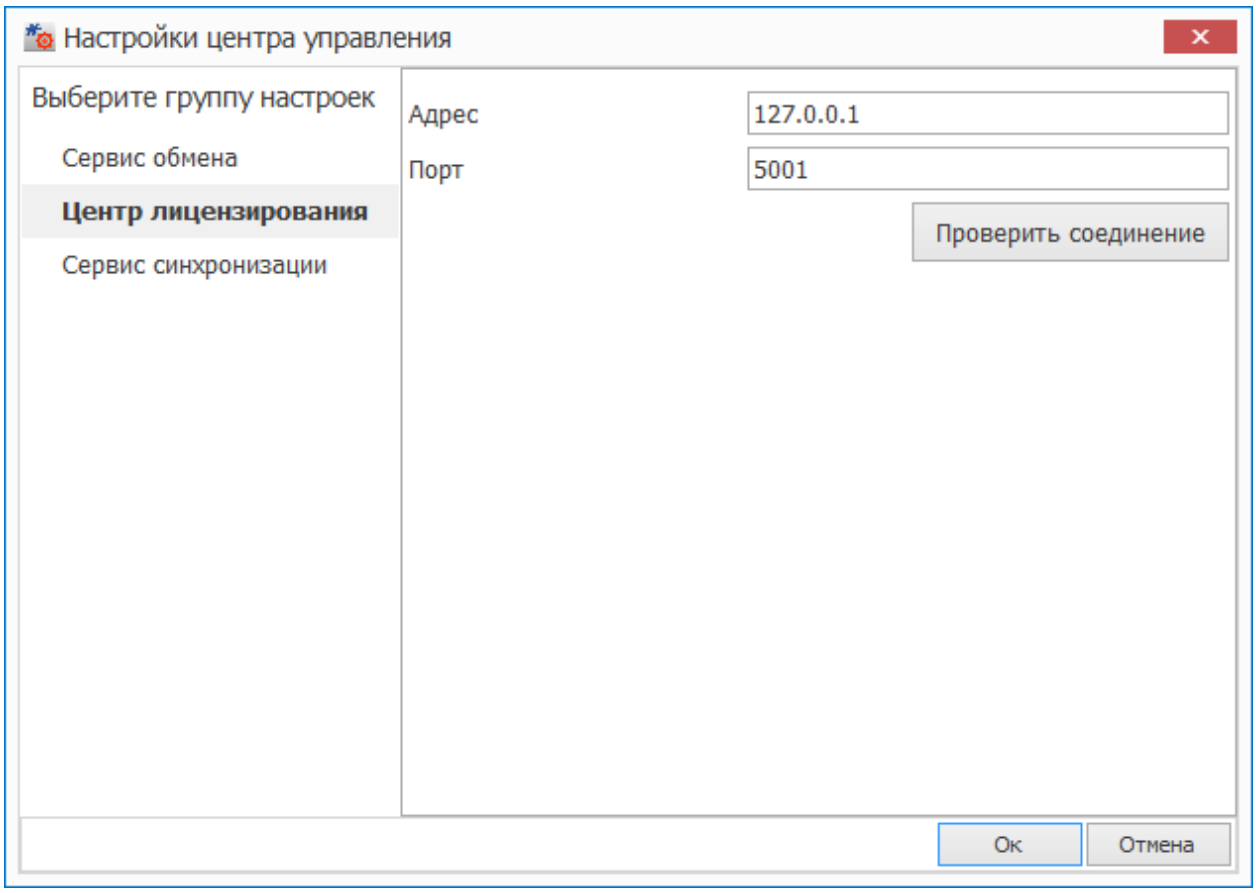
Макс. размер лога, МБ – максимальный размер лог-файла в МБ. По достижении указанного размера, лог-фал обрезается.

Расширенное логирование в БД – при установленном флаге лог записывается в таблицу EXCH_Log. В рамках обработки каждого объекта обмена, СО выполняет набор операций над источником и приёмником. Перед началом выполнения каждой операции при установленной настройке фиксируется запись в таблице EXCH_Log.



Настройка Сервиса лицензирования

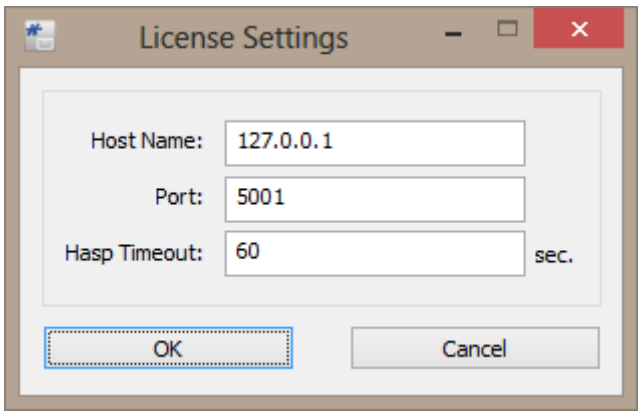
Для настройки параметров Сервиса лицензирования перейдите в группу настроек **Сервис лицензирования**. Введите адрес и порт Сервиса лицензирования. Для проверки соединения нажмите на кнопку **Проверить соединение**.



Помимо настройки параметров Сервиса лицензирования в Центре управления, необходимо прописать сетевые настройки Сервиса лицензирования. Это требуется в следующих случаях:

1) Если СЛ используется на другой машине (не на той, на которой установлен ЦУ).
2) Если СЛ не использует порт по умолчанию.

Для этого следует запустить файл licensesettings.exe в директории установки Сервиса лицензирования (C:\Program Files\CDC\Optimum\LicenseService).

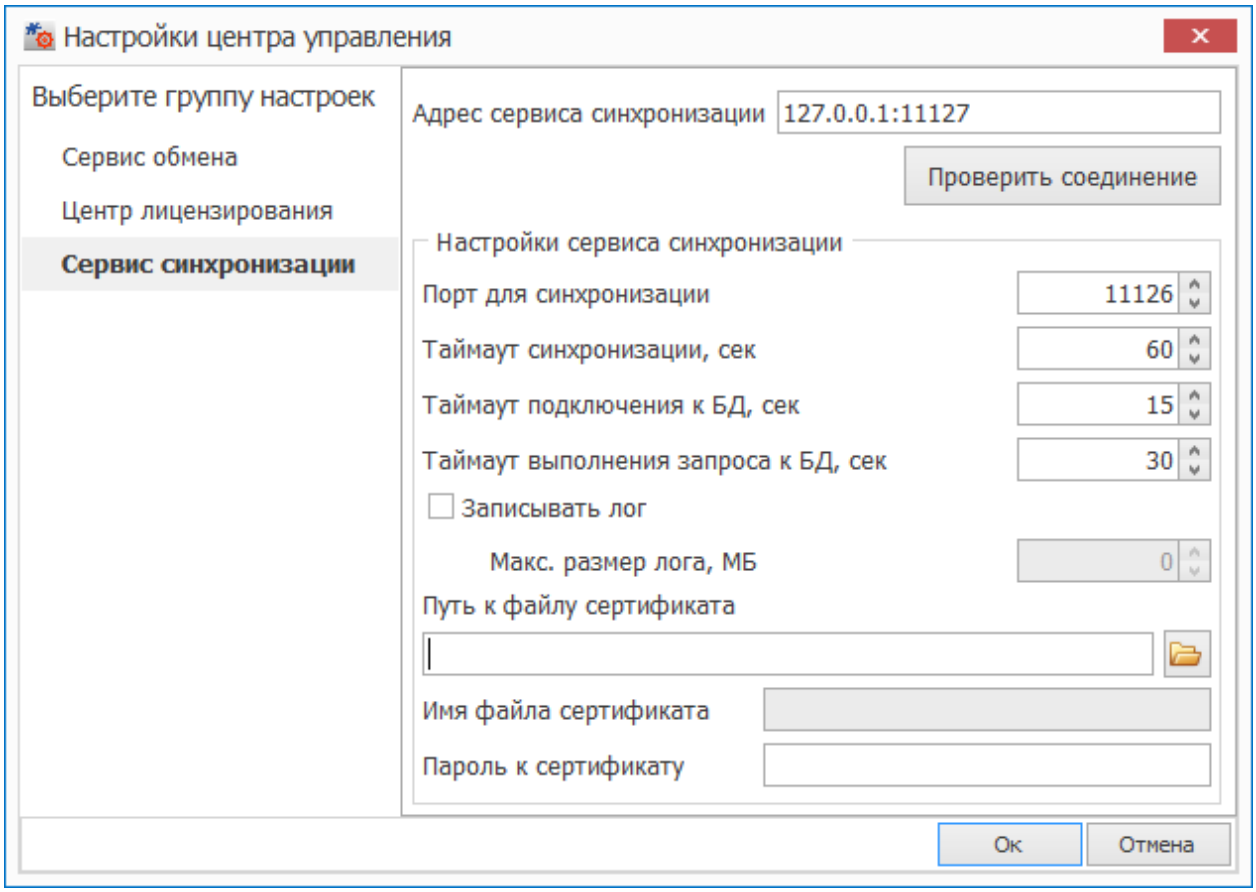


- Host Name: - IP-адрес машины, на которой установлен Сервис лицензирования и к которой подключен HASP-ключ.

- Port: - номер порта.
- Hasp Timeout: - допустимое время отклика от HASP-ключа при обращении к нему Сервиса лицензирования.

Настройка Сервиса синхронизации

Для настройки Сервиса синхронизации перейдите в группу настроек **Сервис синхронизации**. Введите адрес и порт Сервиса синхронизации. Для проверки соединения нажмите на кнопку **Проверить соединение**.



Настройки сервиса синхронизации:

- Порт для синхронизации - порт, прослушиваемый Сервисом синхронизации по умолчанию, допускается использовать другие порты.
- Таймаут синхронизации, сек.
- Таймаут подключения к БД, сек.
- Таймаут выполнения запроса к БД, сек.
- Записывать лог – записывать журнал СС в лог-файл или нет.
- Макс. размер лога, МБ - максимальный размер лог-файла в МБ. По достижении указанного размера, лог-фал обрезается.

- Путь к файлу сертификата – путь до пользовательского сертификата в формате «PFX» для шифрования. Если файл задан, то используется «SSL» протокол для связи с МУ.
- Имя файла сертификата.
- Пароль к сертификату.

Для обеспечения необходимой безопасности при передаче данных существует возможность добавления пользовательского сертификата в формате «PFX»(англ. Personal Information Exchange) в СС. Данный сертификат задает открытый и закрытый ключи, используемые для шифрования при соединении с МУ с использованием протокола SSL. При этом в случае отсутствия в СС пользовательского сертификата обмен данными проводиться не будет.

Методика разработки приложений

При разработке мобильного приложения с использованием платформы Оптимум необходимо выделить минимум две роли – разработчик серверной части и разработчик мобильной части. Разработчик серверной части должен уметь работать с используемым BackEnd, знать язык запросов SQL, обладать навыками проектирования БД. Разработчик мобильной части должен уметь разрабатывать нативные приложения под целевую мобильную платформу с использованием стандартных инструментов разработки. Кроме того, разработчику мобильной части также требуются навыки работы с базой данных SQLite. В реальных проектах обе роли может выполнять один разработчик.

После определения потребностей мобильного приложения необходимо спроектировать структуры данных, которые будут перемещаться между КИС, серверной частью платформы и мобильной частью платформы. Должен быть составлен список необходимых синхронизируемых таблиц и их полей, группы синхронизации и алгоритмы их использования, затем спроектированы методы выгрузки и загрузки данных из/в КИС. Состав данных должен быть согласован между серверным и мобильным разработчиком. Также на этом этапе должны быть определены и согласованы основные переменные платформы (если в них есть необходимость).

Далее серверный разработчик определяет свойства отдельных синхронизируемых таблиц, способы сегментации данных, после чего начинается этап реализации серверной части. На этапе реализации создаются все спроектированные структуры данных, им назначаются соответствующие свойства. Также на этом этапе реализуется интеграция между платформой и источниками данных.

После окончания создания синхронизируемых таблиц, групп синхронизации и переменных платформы в серверной части платформы может быть сгенерирован файл-описание данных и может начинаться этап разработки мобильной части.

Разработчик мобильной части должен реализовать несколько простых шагов по интеграции платформы в мобильное приложение, после чего разработка мобильного приложения происходит обычным образом.

Быстрый старт

Этот раздел показывает простейший сценарий использования платформы. В начале сценария быстрого старта имеется только БД с данными на сервере. Результатом выполнения сценария является мобильное приложение на устройстве Android, которое получает данные из серверной таблице в таблицу мобильной БД. По мере выполнения сценария проходятся минимально-необходимые шаги создания приложения на платформе ОПТИМУМ. Для других мобильных платформ подход практически идентичен.

Первоначальные требования

Сценарий быстрого старта предполагает, что вы работаете на рабочей станции под управлением ОС Windows 8, на которой установлен локально MS SQL Server 2014 Express Edition.

http://msdn.microsoft.com/ru-ru/evalcenter/hh230763.aspx

Также на этой станции установлены все компоненты платформы в соответствии с разделом «Установка».

На MS SQL сервере должна быть развернута стандартная демонстрационная база AdventureWorks, которая может быть свободно загружена по следующей ссылке:

http://msftdbprodsamples.codeplex.com/releases/view/93587

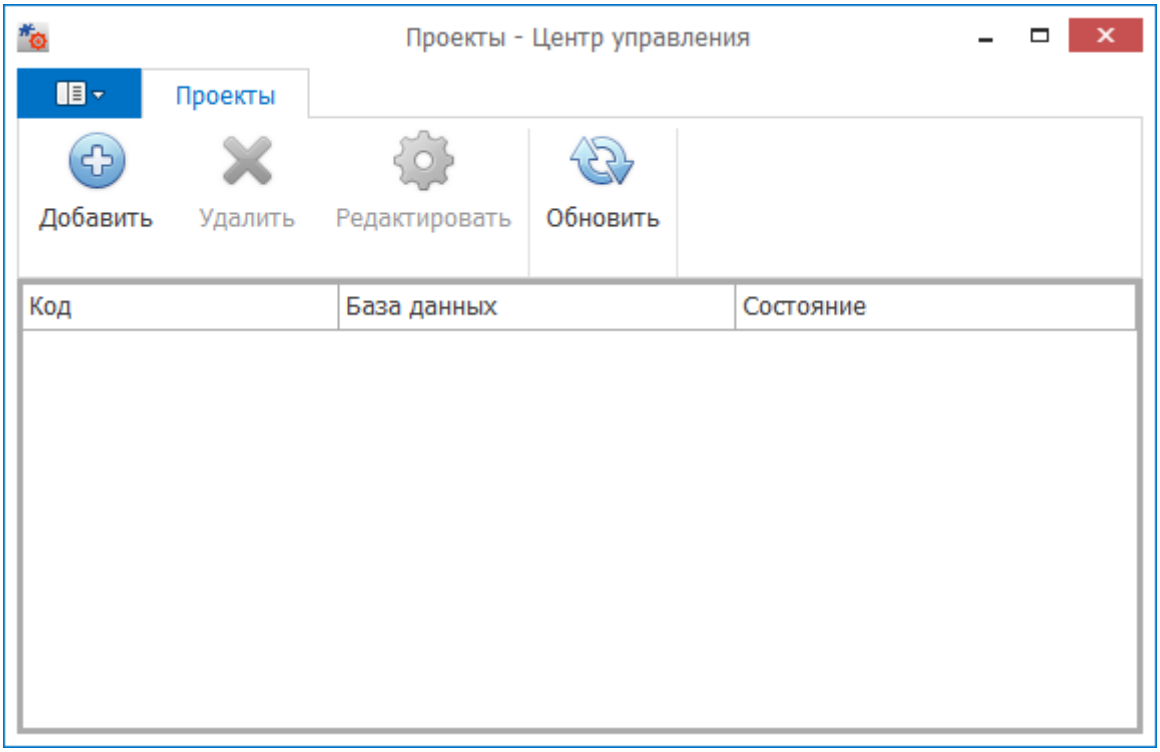
Для разработки примера мобильного приложения под платформу Android должна быть установлена Android SDK.

http://developer.android.com/sdk/index.html

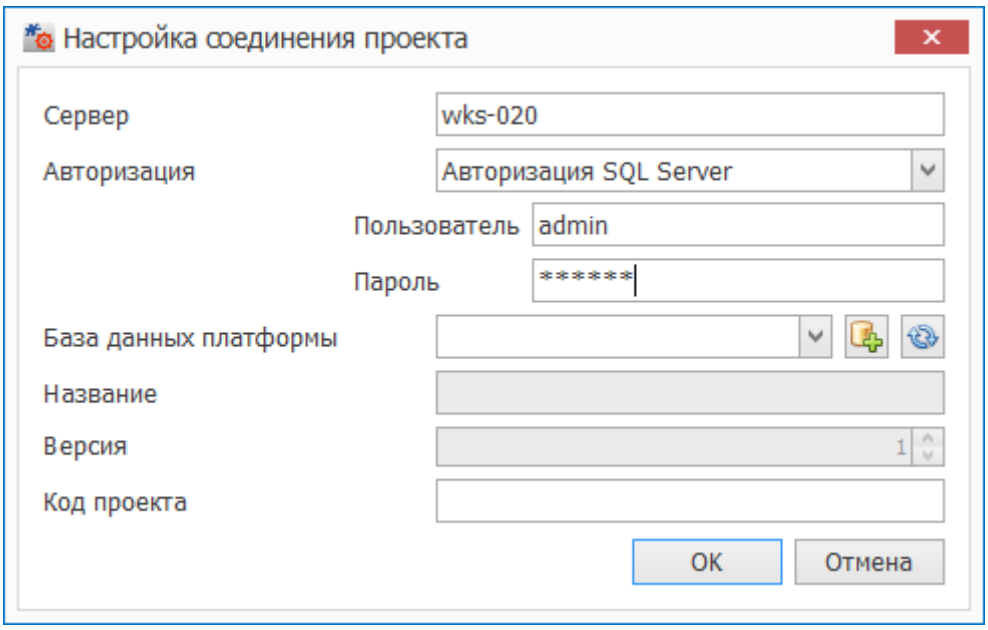
Создание и настройка проекта

Для создания проекта выполните следующие действия:

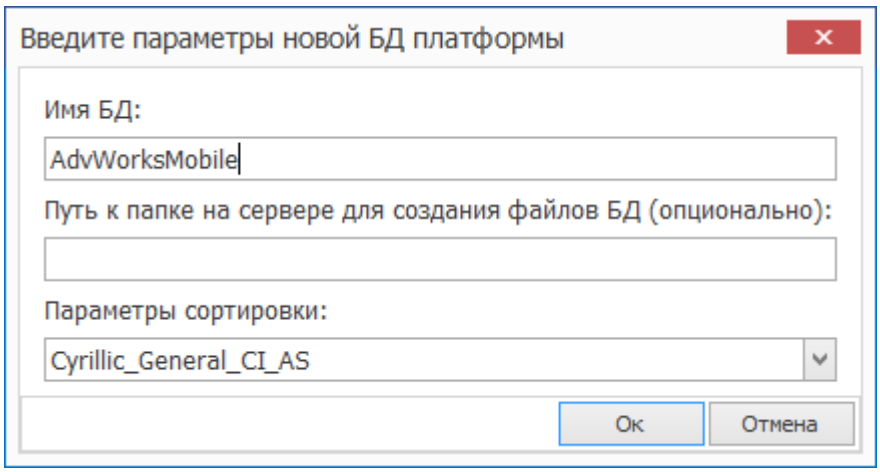
1. Запустите Центр управления и нажмите на кнопку **Добавить**.



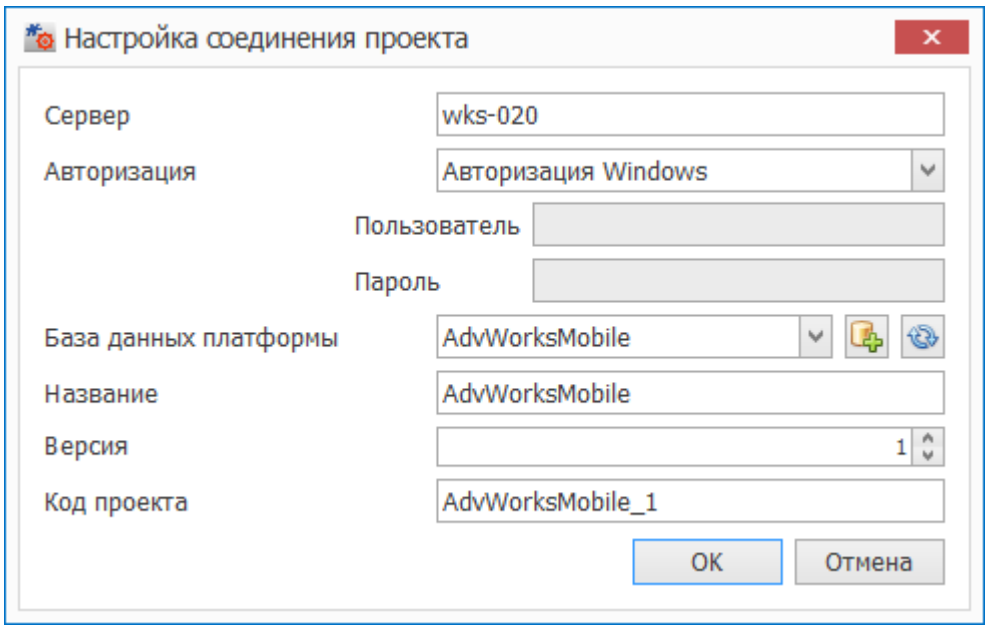
2. Введите параметры доступа к вашему MS SQL серверу, выберите имя БД из списка, если БД уже создана ранее, или воспользуйтесь кнопкой **Создать новую БД платформы**.



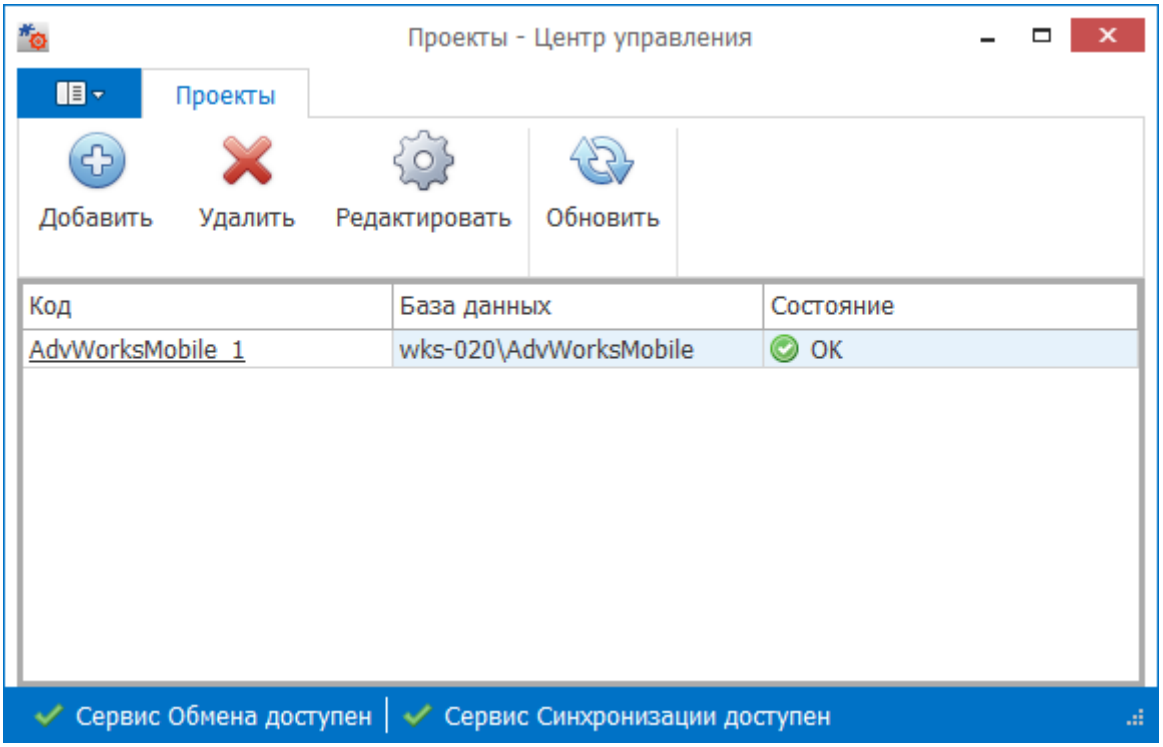
3. Если БД создается впервые, в открывшемся окне введите её название и нажмите на кнопку **Ок**.



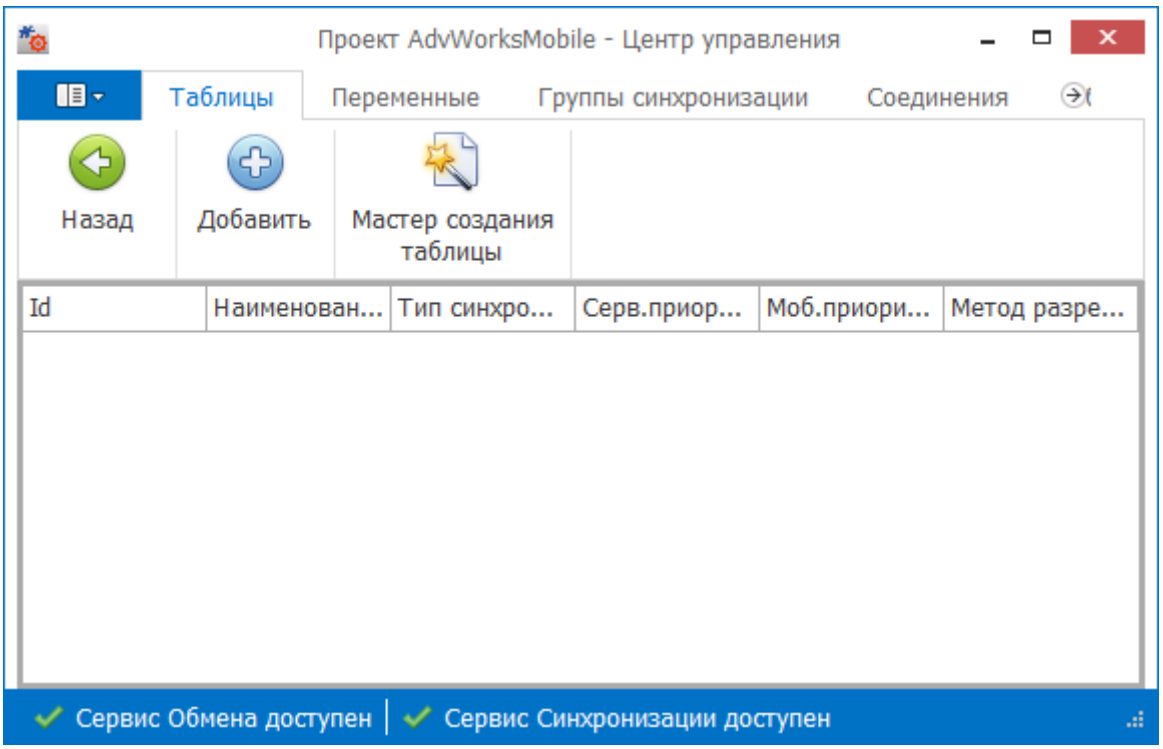
4. Поля **Название**, **Версия** и **Код проекта** будут заполнены автоматически.



5. В списке проектов появится база.



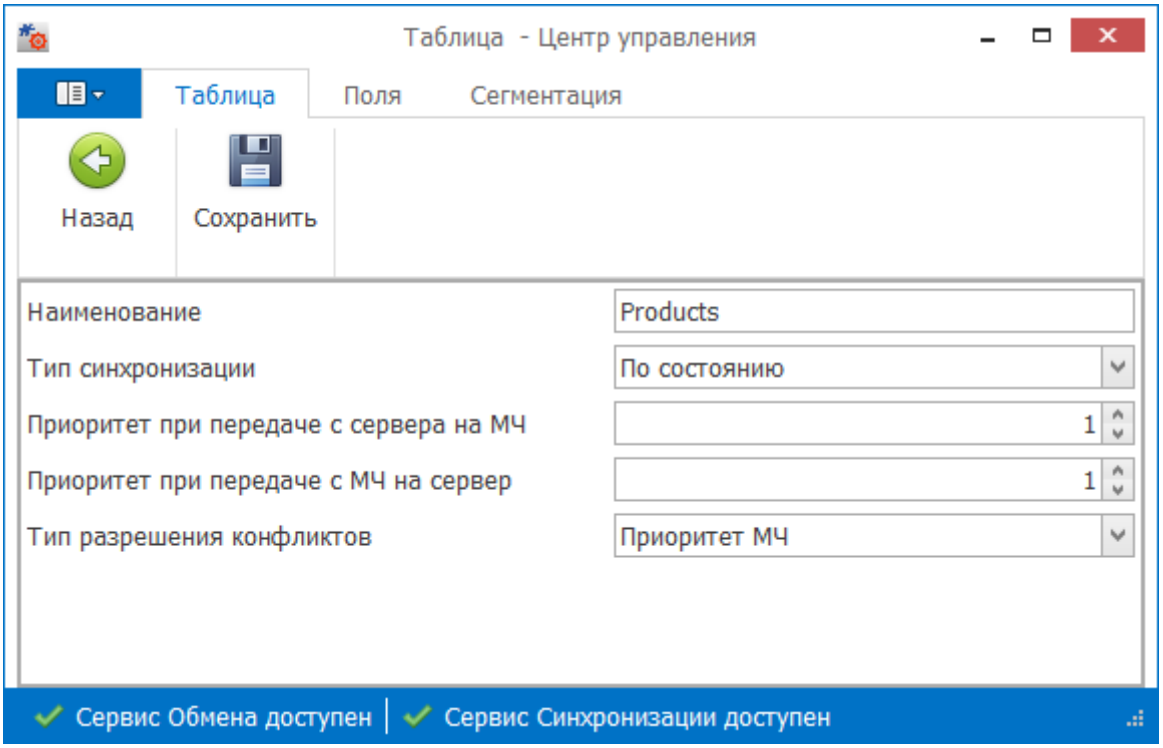
6. После создания базы нажмите на название проекта – в результате вы перейдете в режим работы с проектом.



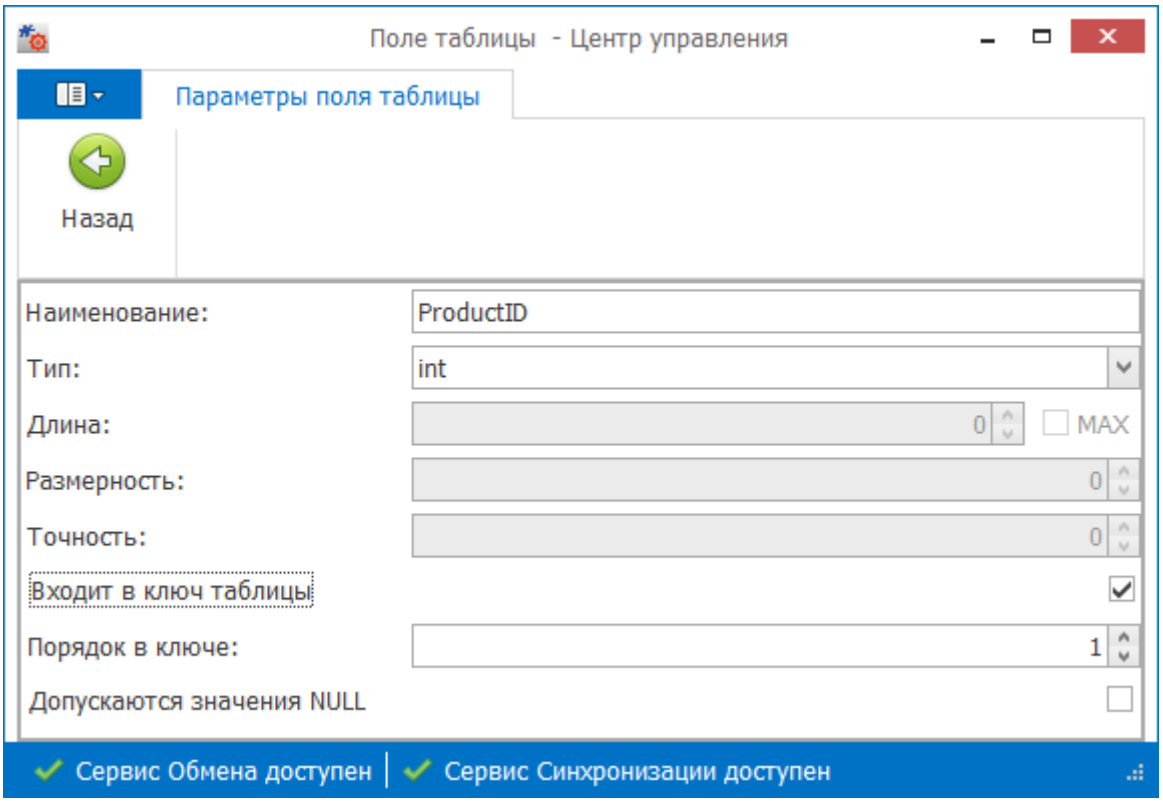
7. Далее необходимо создать в БД таблицы синхронизации.

Примечание: это можно сделать автоматически с помощью Мастера создания таблицы. Однако для того чтобы продемонстрировать на примере, какие таблицы и в какой последовательности создаются, в данной инструкции рассмотрено создание таблиц вручную.

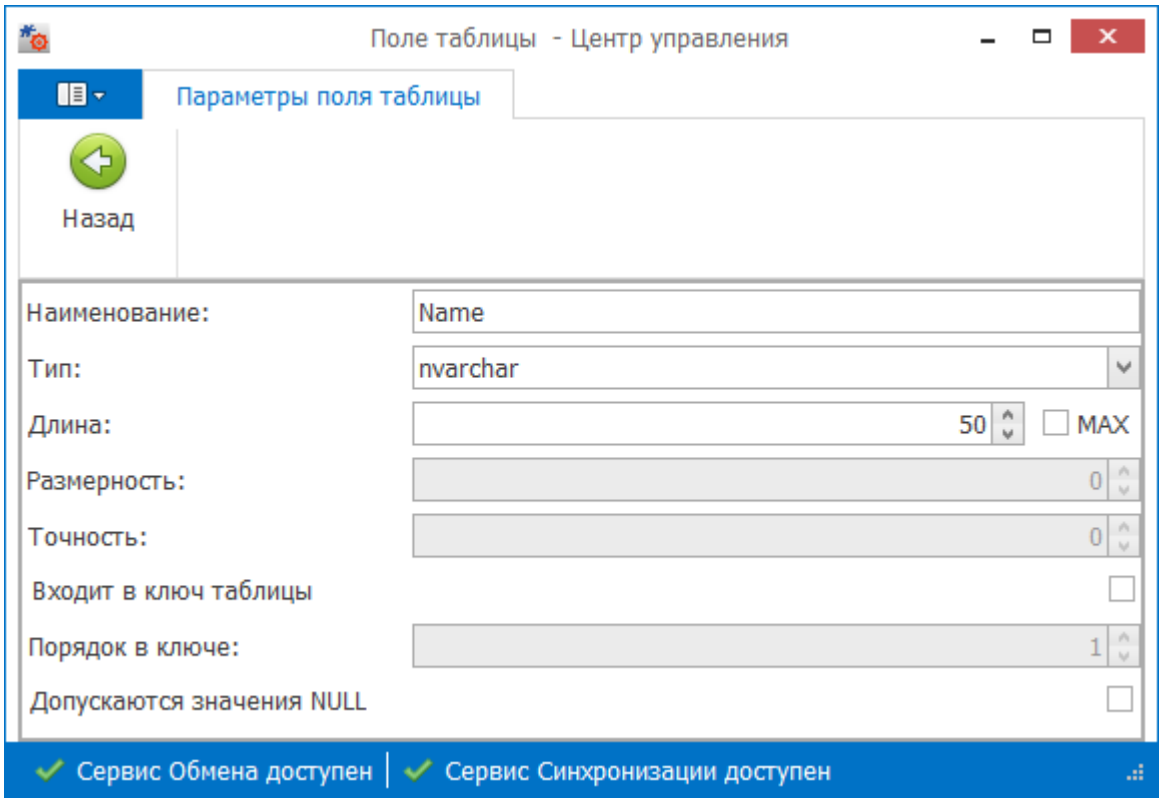
Для того чтобы добавить таблицу синхронизации вручную, на вкладке **Таблицы** нажмите на кнопку **Добавить** и введите параметры таблицы.

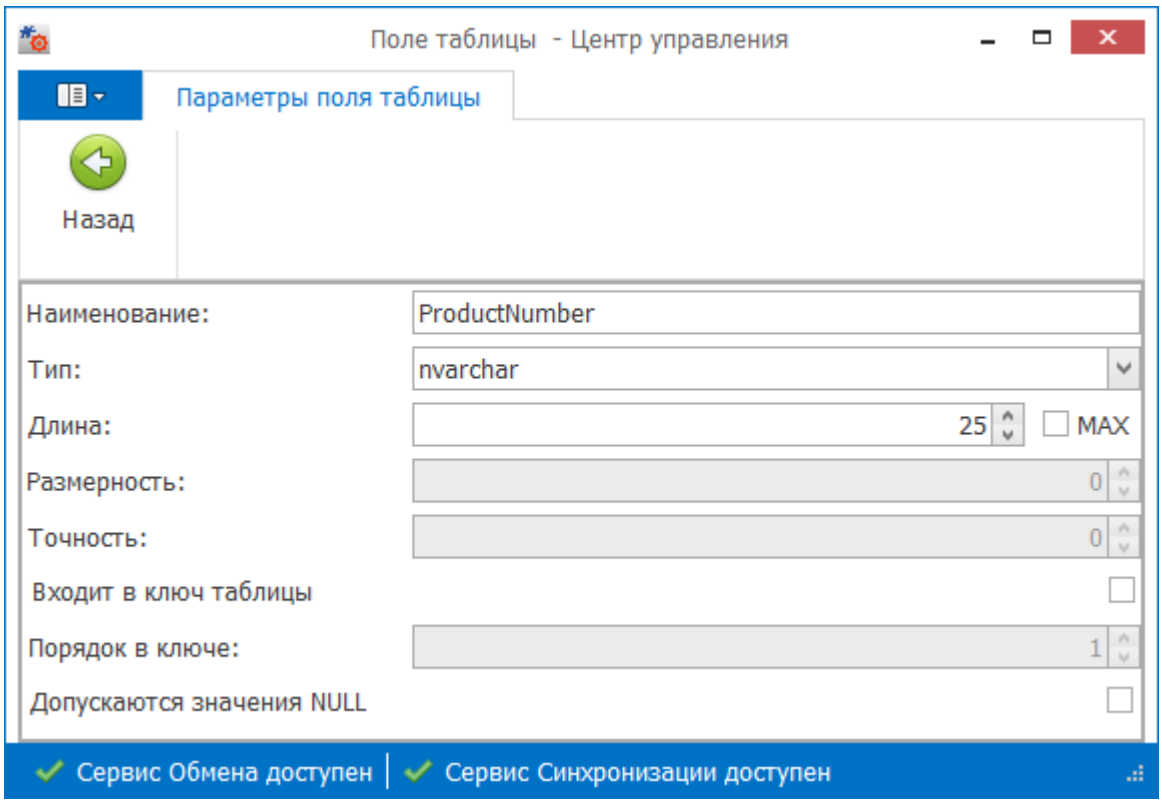


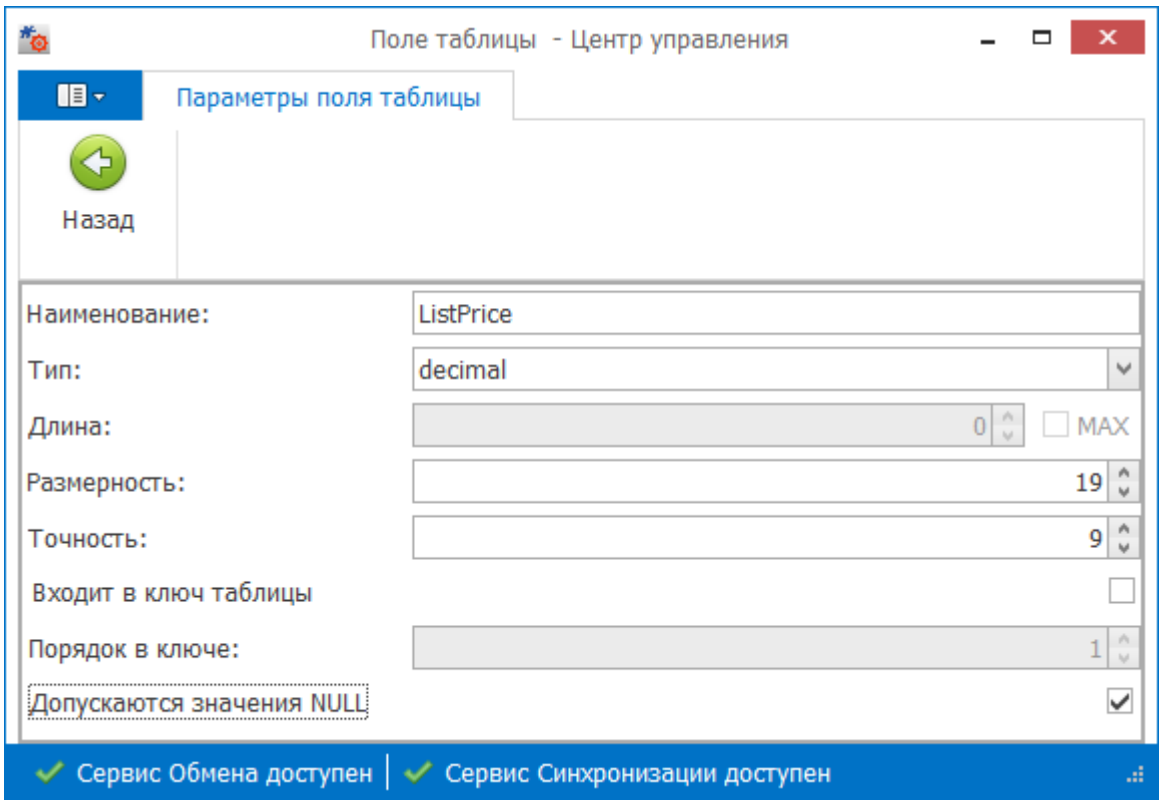
8. Перейдите на вкладку **Поля** и добавьте поле таблицы, установив ему следующие параметры:

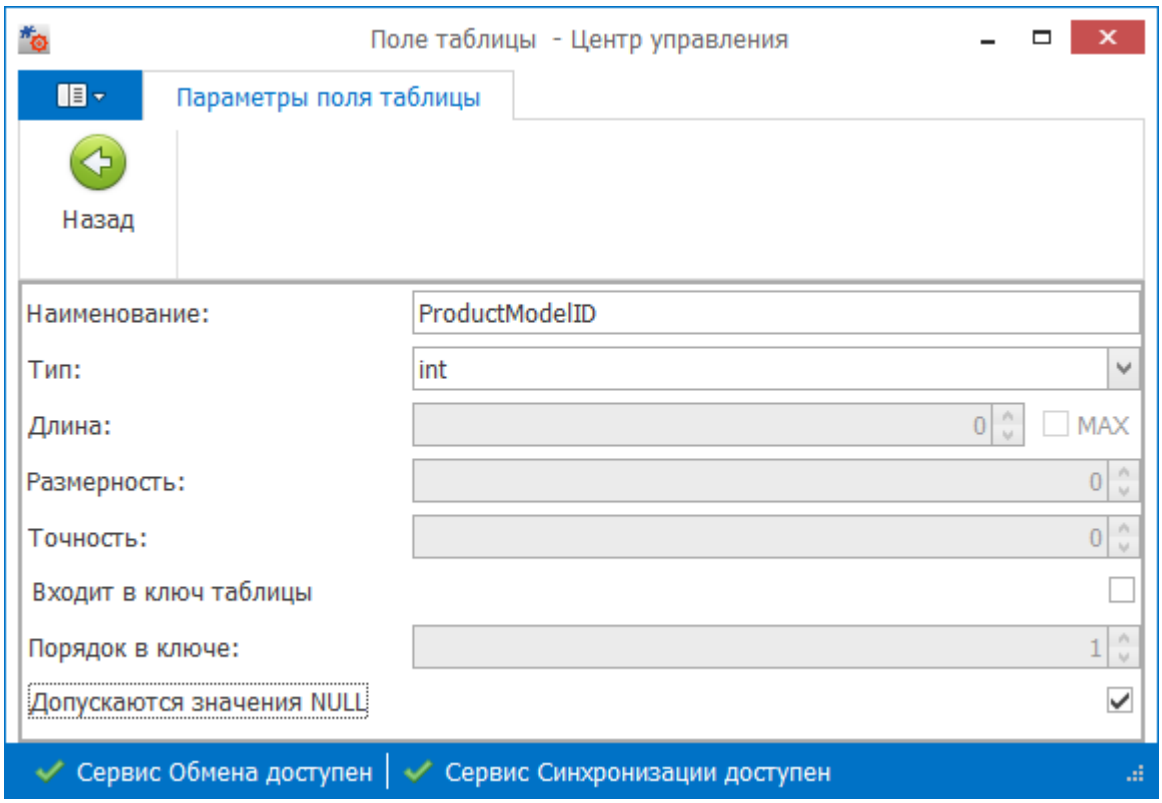


9. Аналогично добавьте следующие поля:

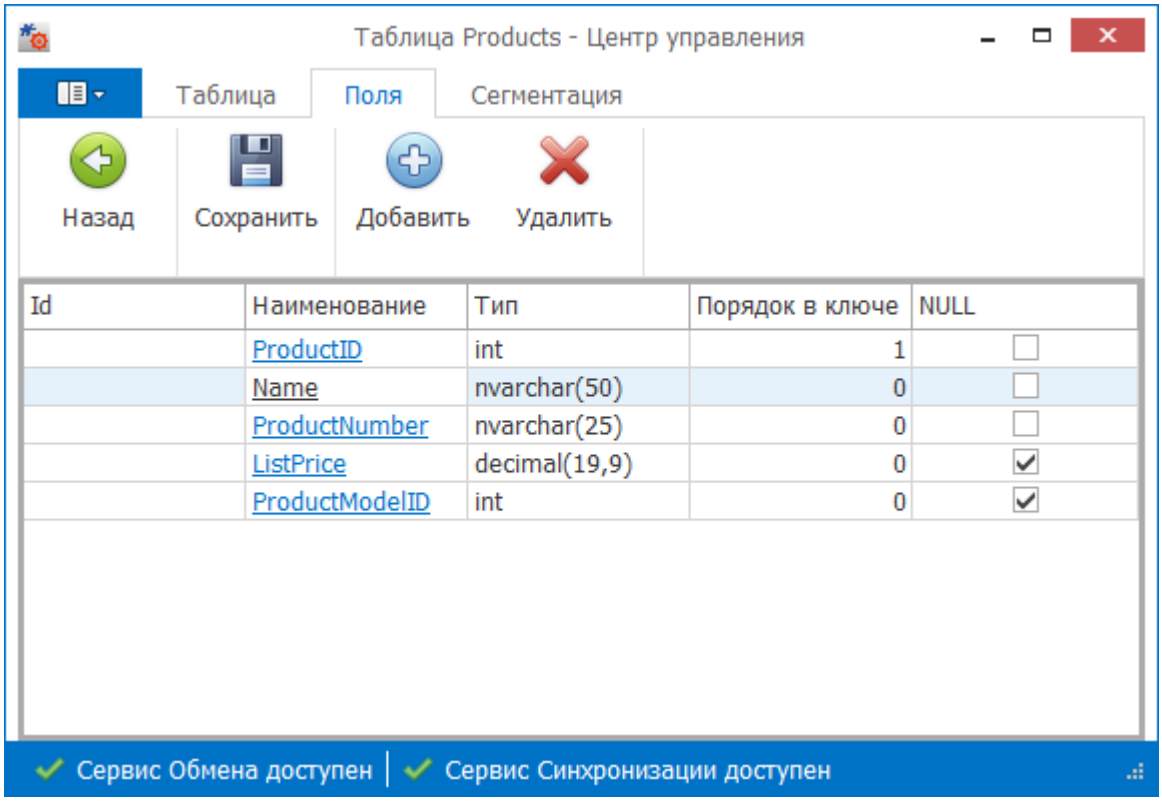



Поле таблицы - Центр управления
Параметры поля таблицы
Назад
Наименование:
ProductNumber
Тип:
nvarchar
Длина:
25
MAX
Размерность:
0
Точность:
0
Входит в ключ таблицы
Порядок в ключе:
1
Допускаются значения NULL
Сервис Обмена доступен
Сервис Синхронизации доступен

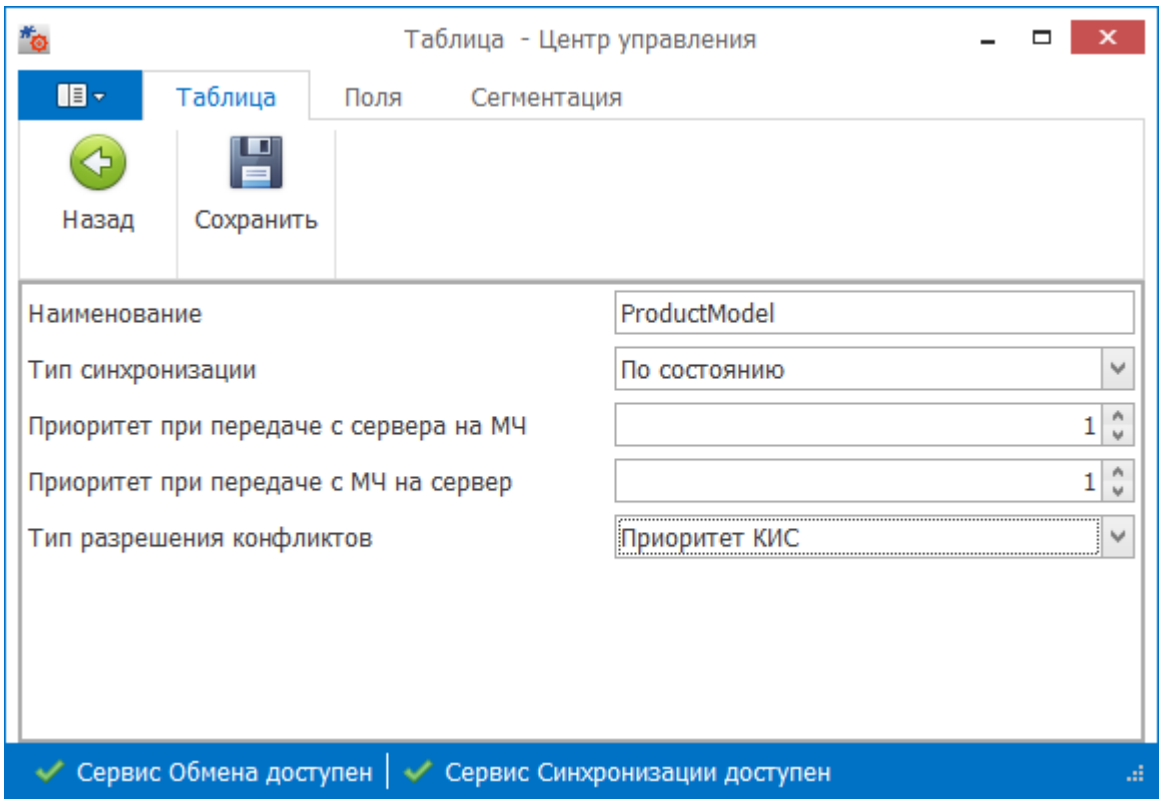

Поле таблицы - Центр управления
Параметры поля таблицы
Назад
Наименование:
ListPrice
Тип:
decimal
Длина:
0
MAX
Размерность:
19
Точность:
9
Входит в ключ таблицы
Порядок в ключе:
1
Допускаются значения NULL
Сервис Обмена доступен
Сервис Синхронизации доступен



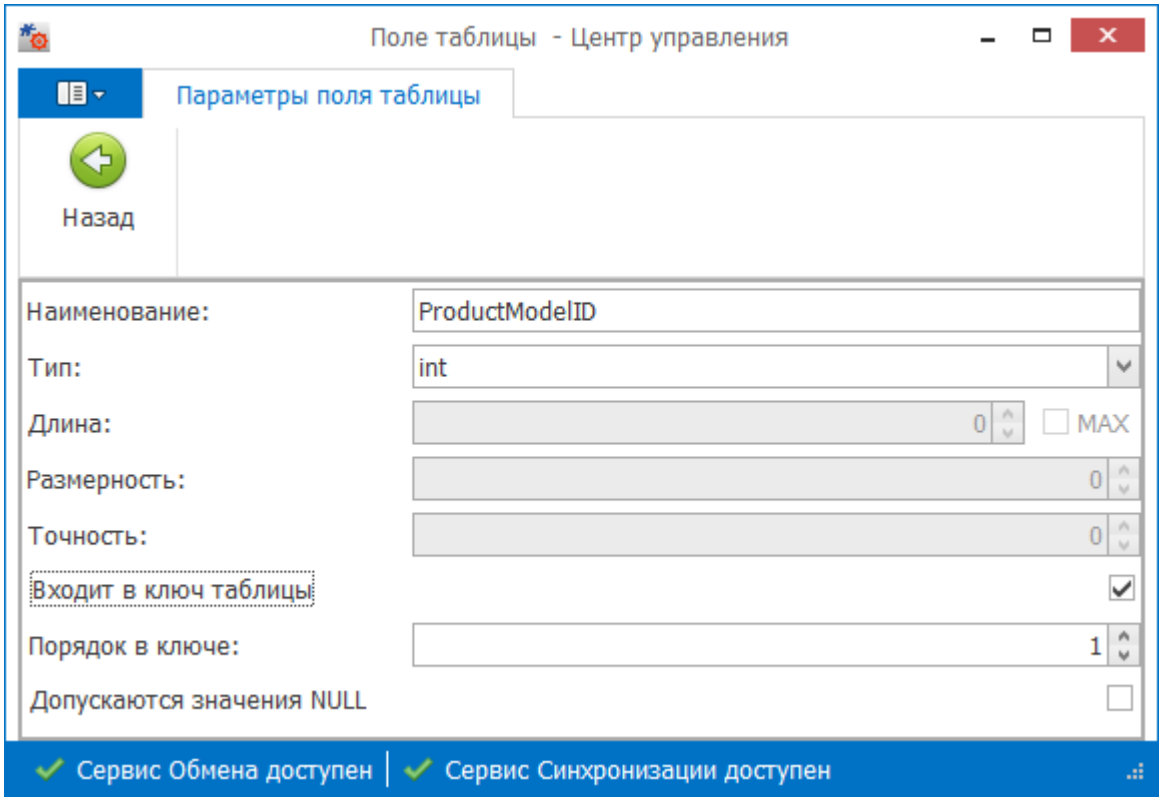
10. Вернитесь на вкладку **Поля** и нажмите на кнопку **Сохранить**.

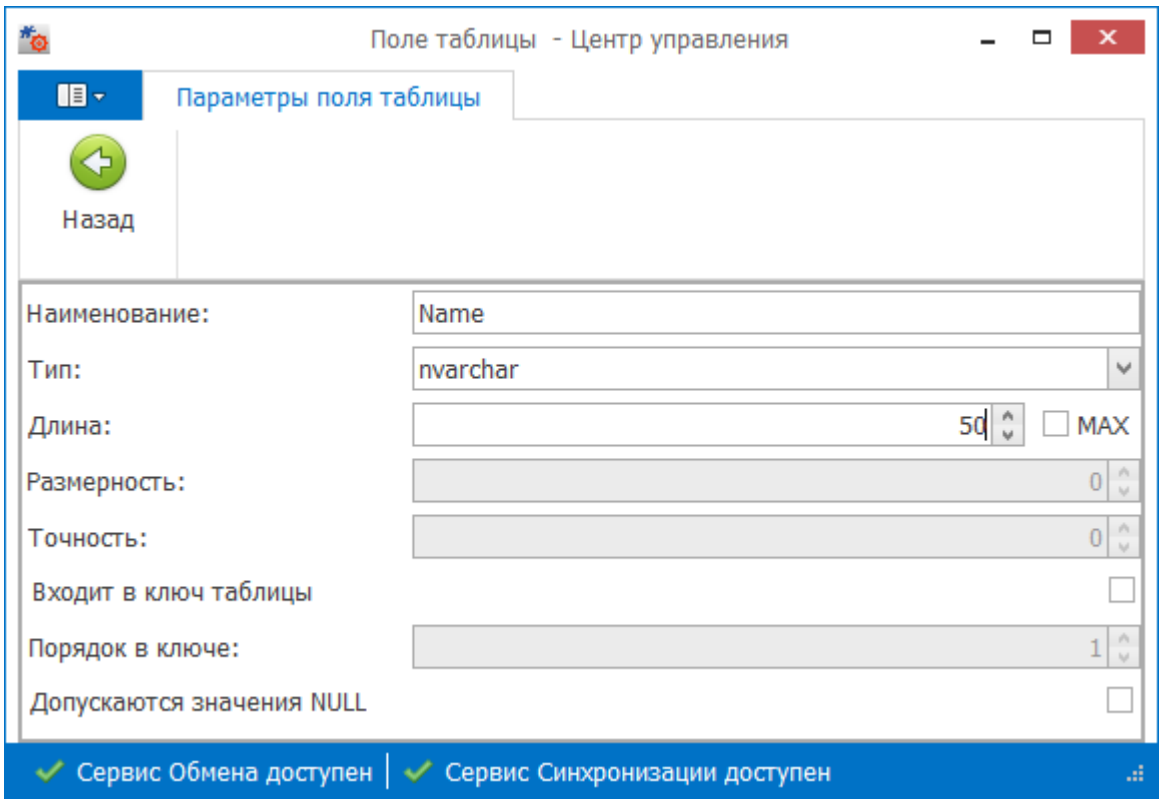


11. Добавьте и сохраните еще одну синхронизируемую таблицу со следующими параметрами:

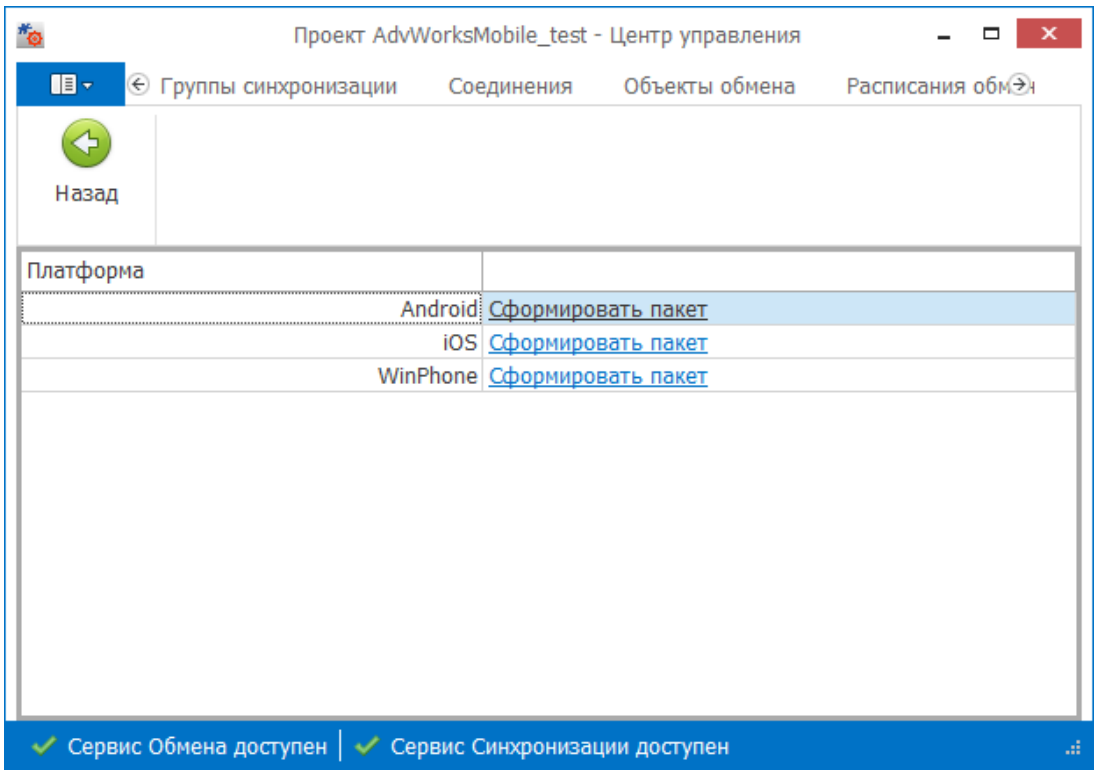


12. Перейдите на вкладку **Поля**, добавьте и сохраните следующие поля таблицы:





13. Перейдите на вкладку **Моб. часть** и скачайте подготовленный для проекта файл-архив с описанием БД и необходимыми библиотеками, нажав на **Сформировать пакет** для платформы Android.



14. Сохраните его под именем AdvWorksMobile.zip.

Результат: создан проект платформенного приложения. На MS SQL сервере развернута база платформы. В этой базе созданы две синхронизационные таблицы – Products и ProductModel, которые будут содержать данные по продуктам и их моделям из аналогичных таблиц БД AdventureWorks.

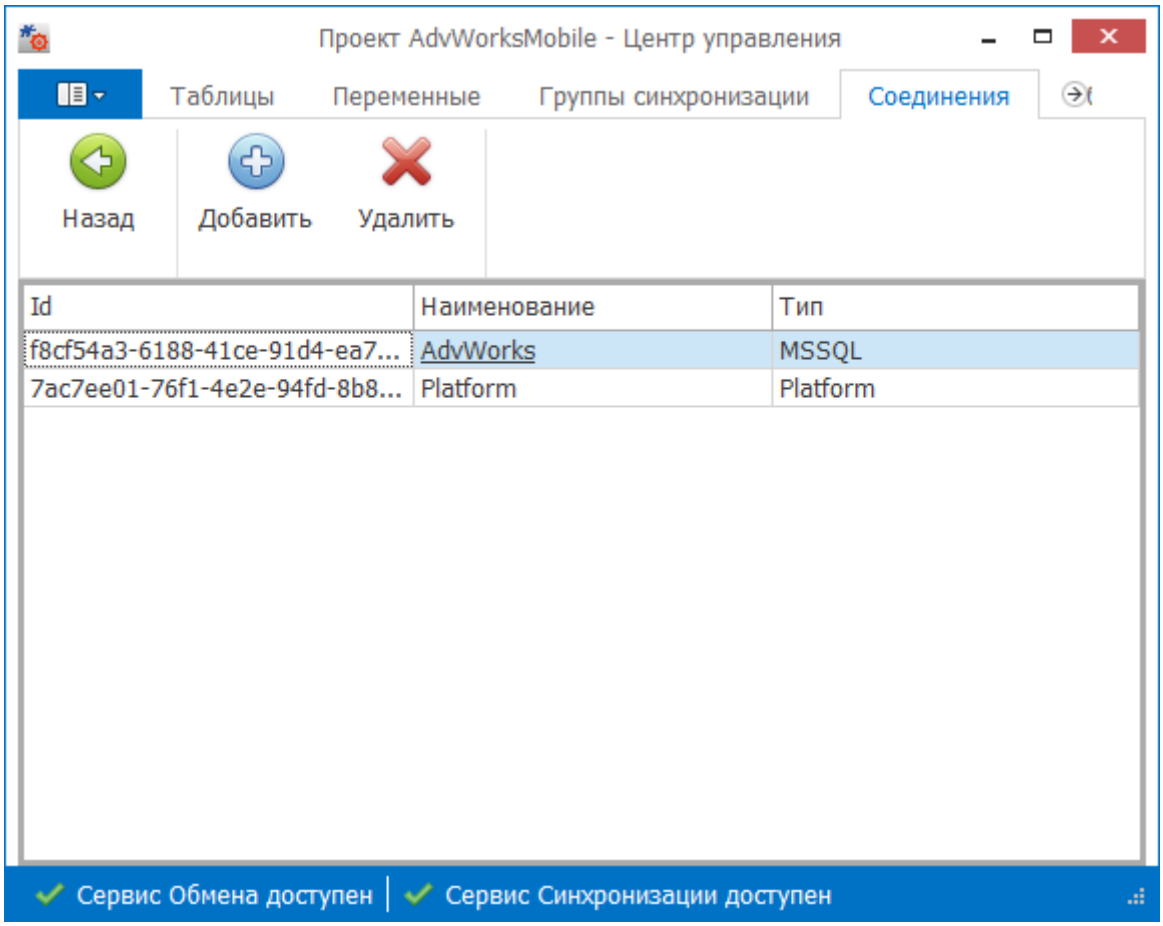
Обмен

На этом этапе настраивается передача данных из БД AdventureWorks в платформу. Для передачи данных используется поставляемый в составе платформы Сервис обмена данными, который настраивается через интерфейс Центра управления.

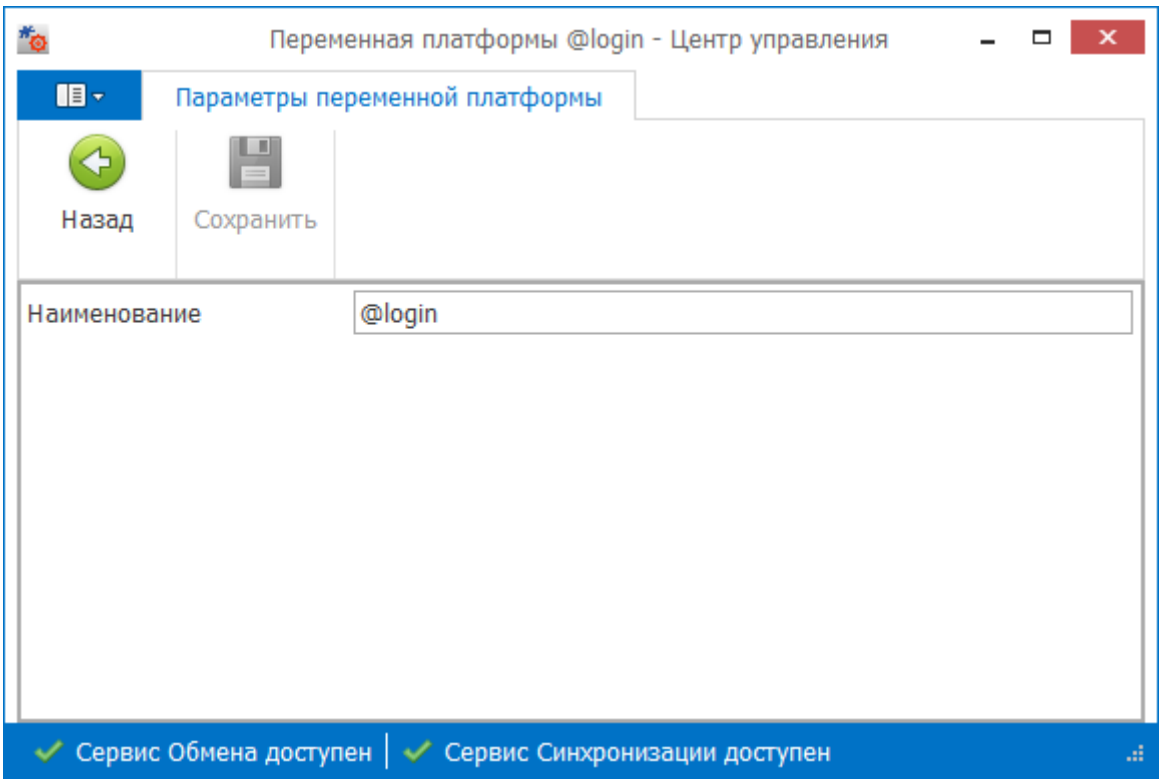
В первую очередь настраивается соединение.

1. На вкладке **Соединения** Центра управления нажмите **Добавить**.
2. На вкладке **Параметры соединения** введите название соединения «AdvWorks», выберите тип «MS SQL» и введите параметры доступа к своему SQL Server. В качестве базы данных выберите «AdventureWorks». Проверьте соединение и нажмите **Сохранить**.

Соединение «AdvWorks» будет использоваться как источник данных. В качестве приемника данных будет использоваться соединение с именем «Platform» типа «Platform», которое было создано автоматически при создании проекта.



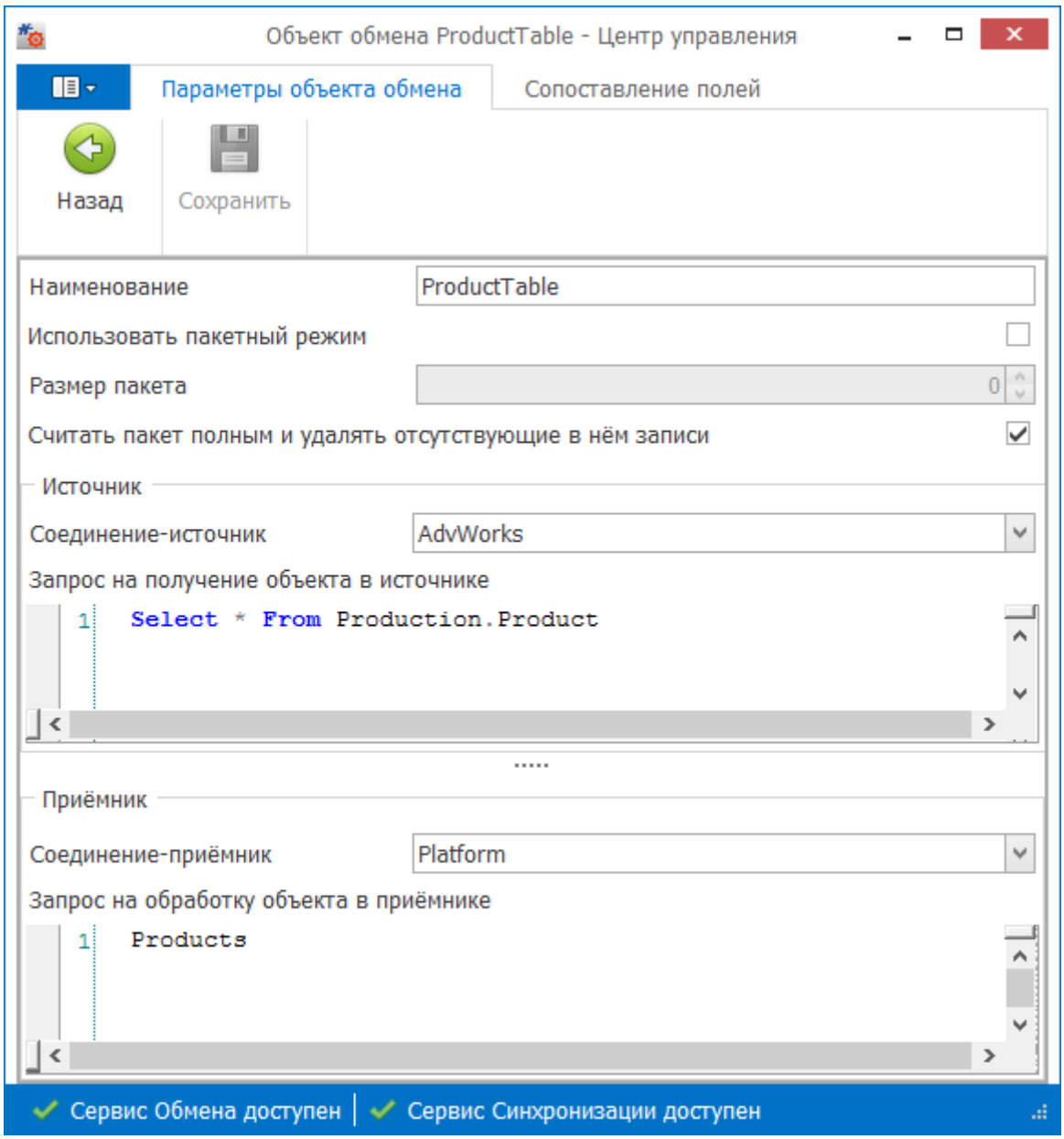
Далее перейдите на вкладку **Переменные** и нажмите на кнопку **Добавить**. Добавьте переменную @login и сохраните её.



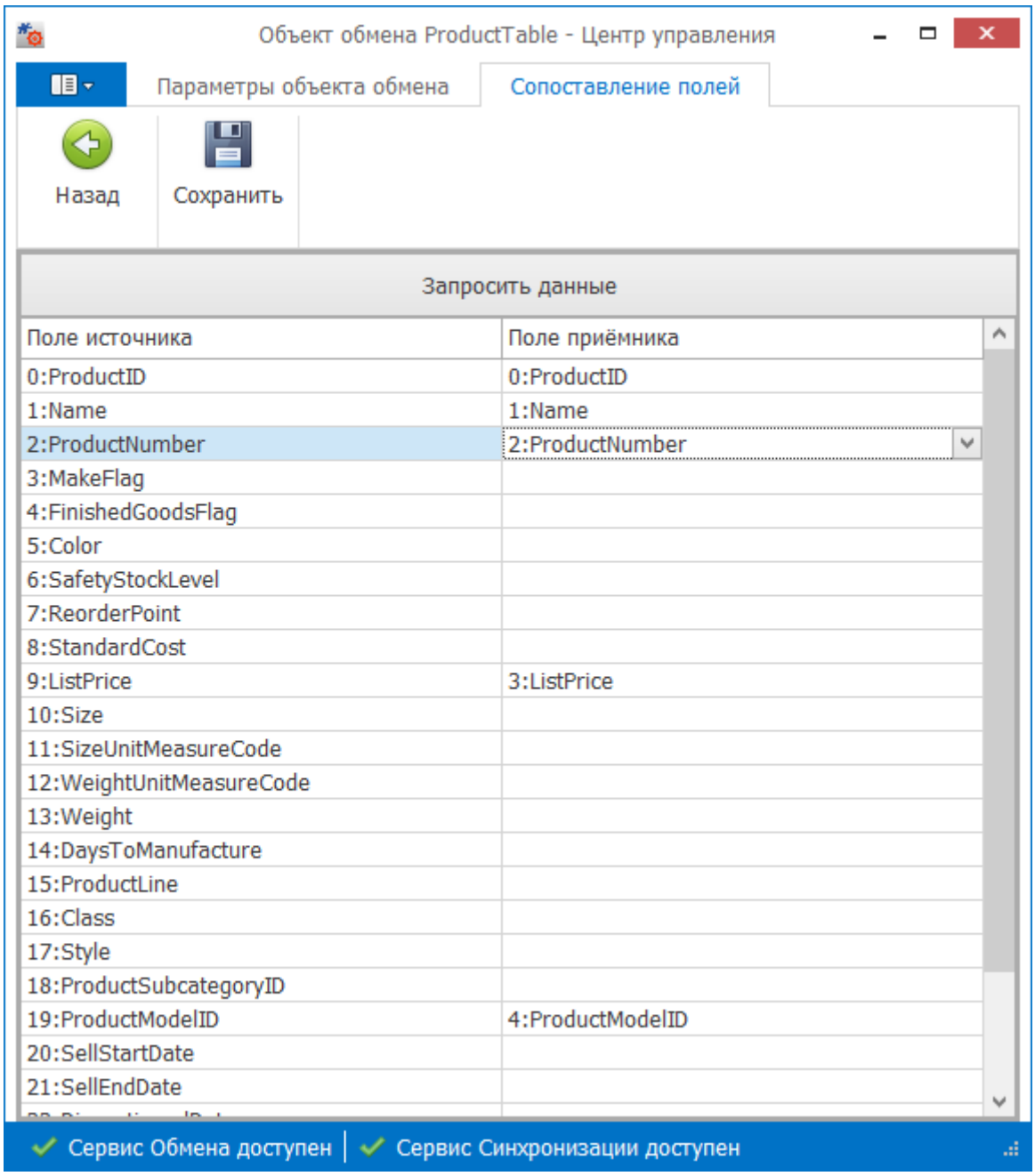
Далее перейдите на вкладку **Объекты обмена**.

1. Для передачи двух таблиц необходимо создать два объекта обмена. Нажмите на кнопку **Добавить**. В появившейся вкладке **Параметры объекта обмена** установите следующие значения:

Параметр	Значение
Наименование	ProductTable
Источник объекта	AdvWorks
Запрос на получение объекта	Select * From Production.Product
Приемник объекта	Platform
Запрос на обработку объекта	Products

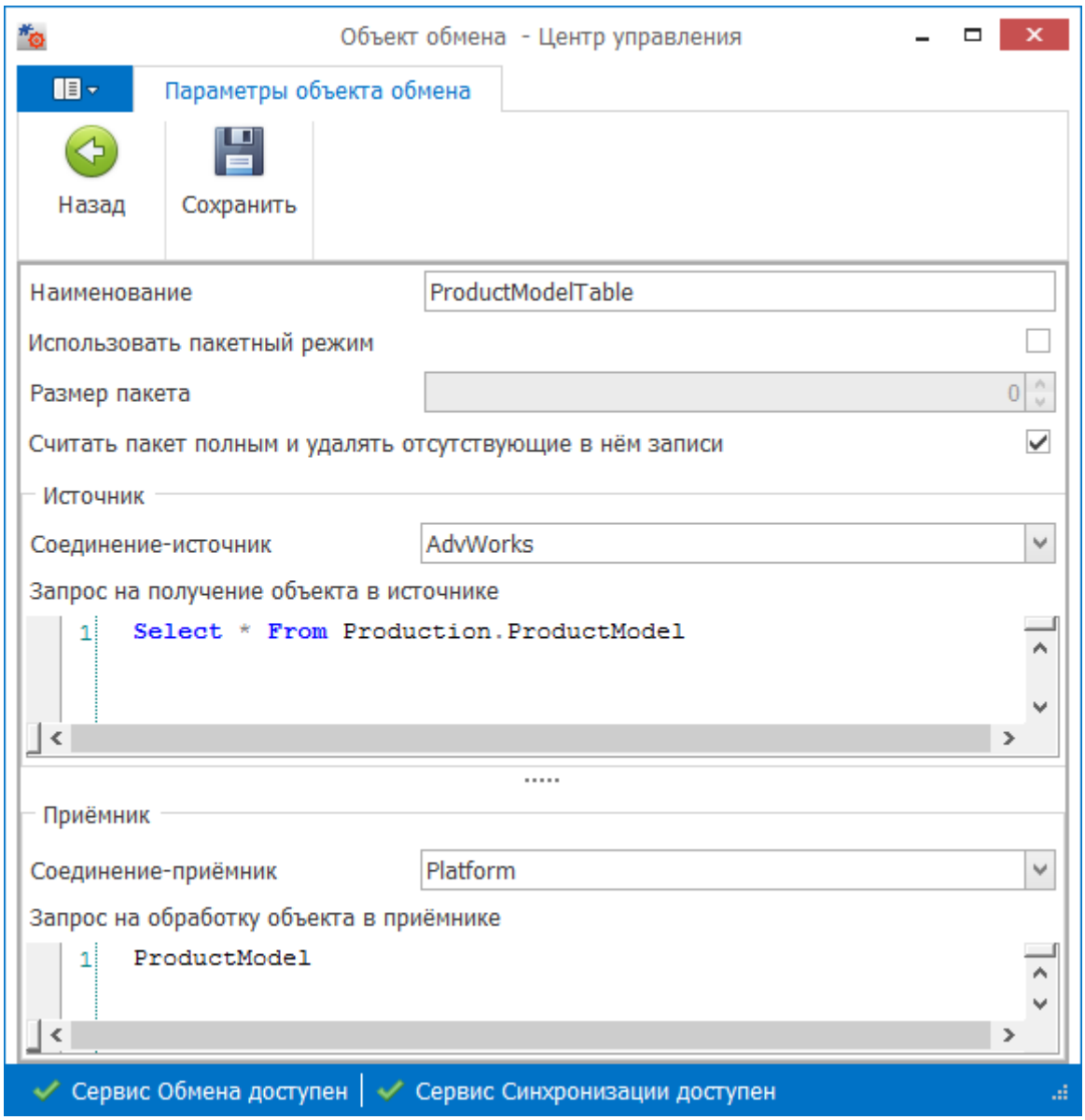


2. После установки свойств нажмите на кнопку **Сохранить** и перейдите на вкладку **Сопоставление полей**. Нажмите на кнопку **Запросить данные**, в левой колонке появятся поля источника данных (т.е. в данном случае все поля таблицы Production.Product из БД AdventureWorks). В правой колонке выберите соответствующие поля синхронизируемой таблицы Products и нажмите **Сохранить**.

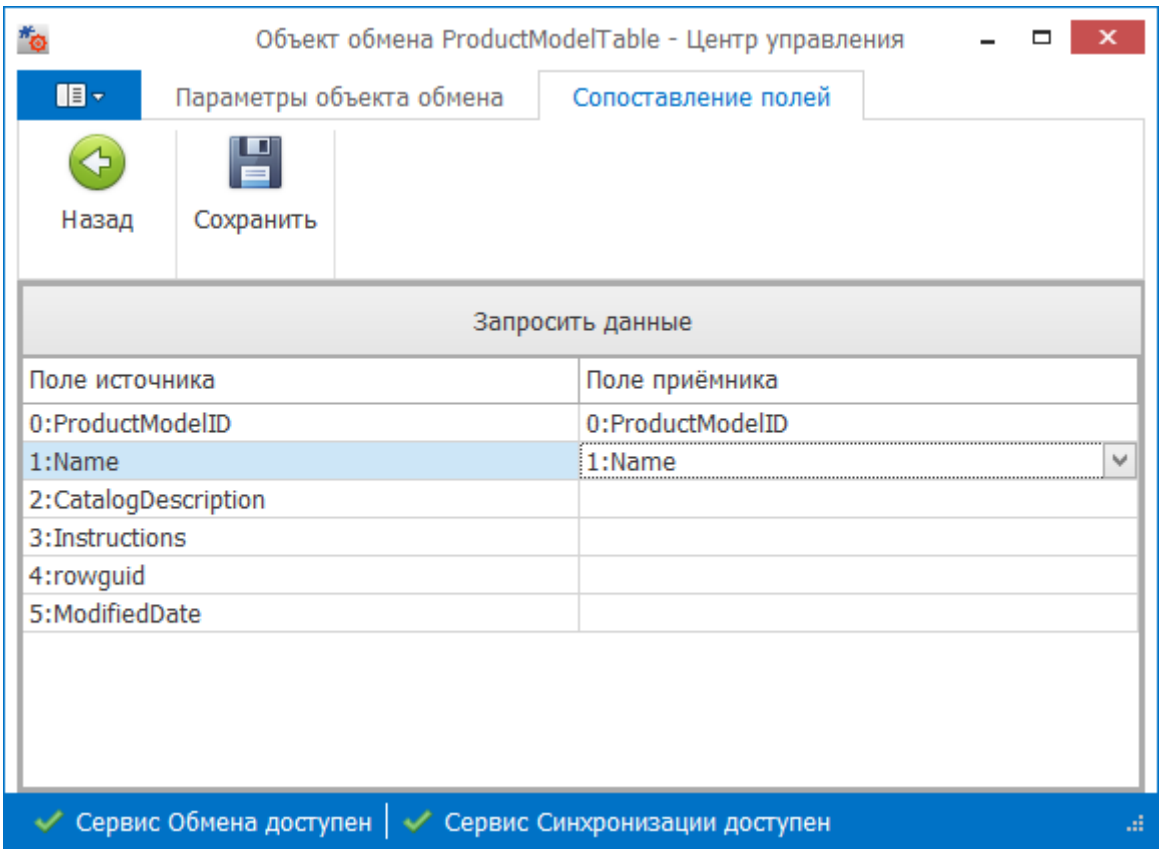


3. Добавьте еще один объект обмена, установите ему следующие параметры:

Параметр	Значение
Наименование	ProductModelTable
Источник объекта	AdvWorks
Запрос на получение объекта	Select * From Production.ProductModel
Приемник объекта	Platform
Запрос на обработку объекта	ProductModel



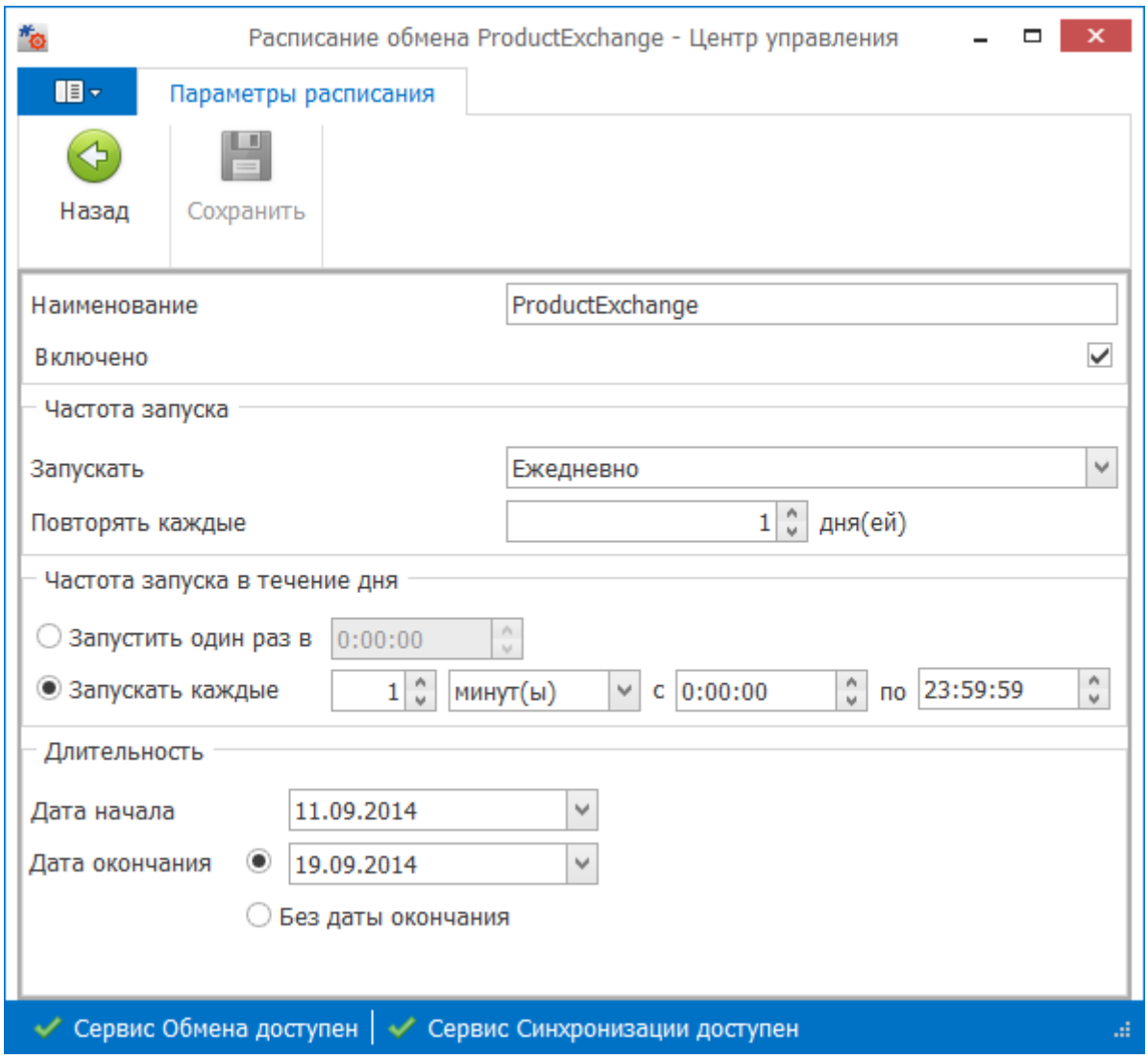
4. Аналогичным образом проставьте соответствия между полями исходной и синхронизируемой таблицы.



Теперь необходимо создать расписание, по которому будет происходить обмен. В данном случае будет использоваться только одно расписание, которое будет запускать оба наших объекта.

1. Перейдите на вкладку **Расписания обмена** и нажмите на кнопку **Добавить**.
2. На вкладке **Параметры расписания** введите значения:

Параметр	Значение
Наименование	ProductExchange
Включено	Да
Запускать	Ежедневно
Повторять каждые	1
Частота запуска	Запускать каждые 1 минуту с 00:00:00 до 23:59:59
Дата начала	Оставьте текущую

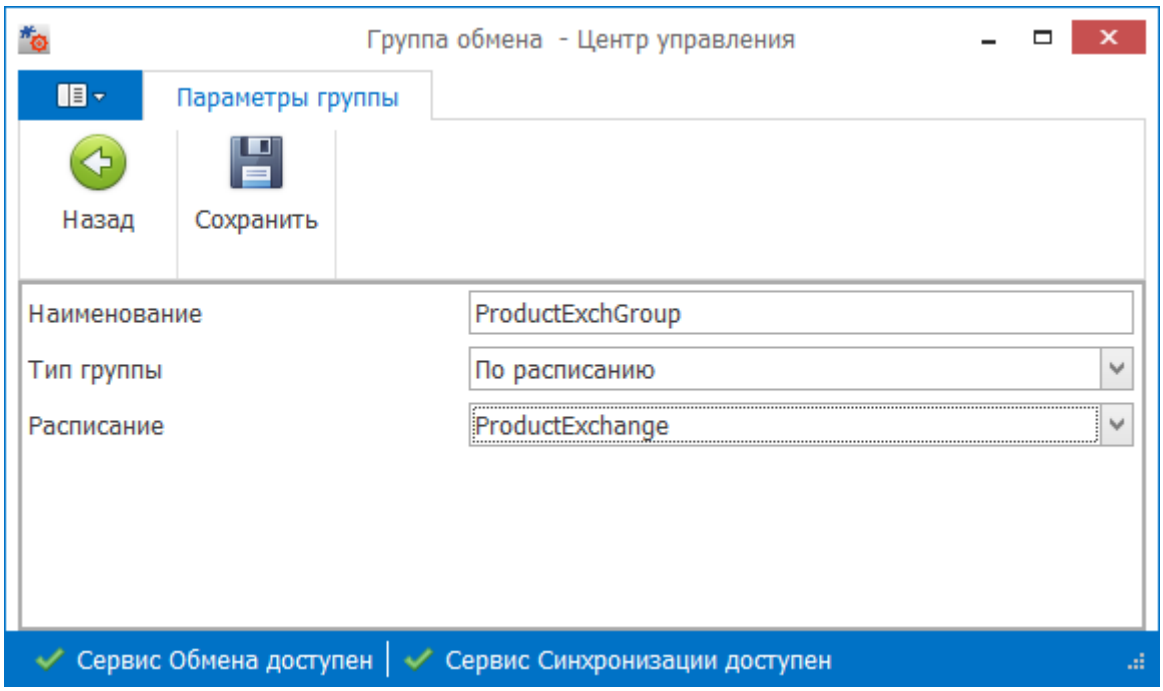


3. Нажмите **Сохранить**.

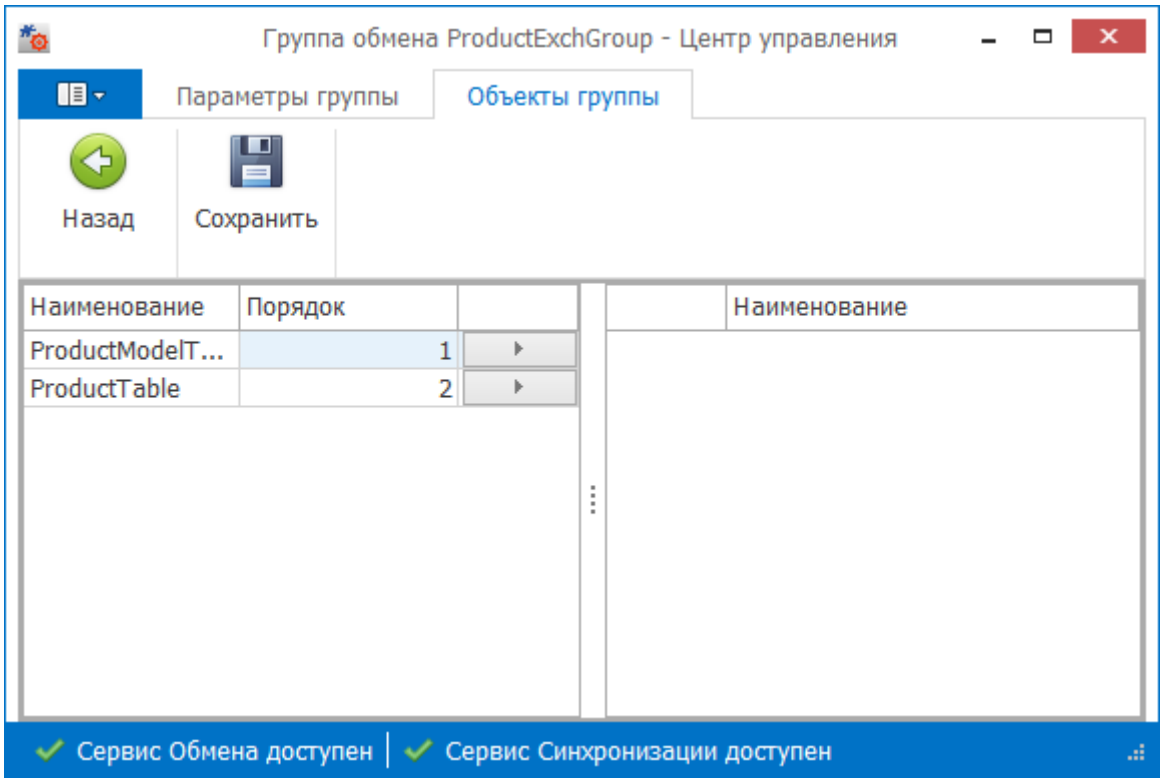
Теперь нужно создать группу обмена, указав для нее расписание и входящие в нее объекты обмена. Для этого:

1. Перейдите на вкладку **Группы обмена** и нажмите на кнопку **Добавить**.
2. В появившейся вкладке **Параметры группы** установите и сохраните следующие значения:

Параметр	Значение
Наименование	ProductExchGroup
Тип группы	По расписанию
Расписание	ProductExchange



3. Нажмите **Сохранить** и перейдите на вкладку **Объекты группы**. На этой вкладке слева показаны объекты, входящие в группу обмена, а справа – объекты, которые могут быть в нее добавлены. Нажмите на кнопку со стрелкой влево в строке объектов ProductModelTable и ProductTable.
4. Нажмите **Сохранить**.



Результат: настроена передача данных из таблиц Product и ProductModel базы данных AdventureWorks в соответствующие таблицы синхронизации базы данных платформы. Теперь

каждую минуту будет производиться чтение информации из источников, а также сравнение, определение измененных, добавленных и удаленных данных, и соответствующее обновление данных в БД платформы.

Создание проекта мобильного приложения Android

Выполните следующие действия:

1. Создайте пустой проект в среде разработки Android. Имя пакета установите в ru.cdc.optimum.quickstart.example1.
2. Возьмите подготовленный на предыдущем шаге AdvWorksMobile.zip и распакуйте его в папку проекта. В результате в папке assets должен появиться xml-файл с описанием мобильной БД, а в папке libs – библиотеки платформы Optimum optimum-x.x.x.x.jar и slf4j-android-x.x.x.jar.
3. Создайте два класса – Example1Application и OptimumOpenHelper.
4. Класс OptimumOpenHelper отвечает за создание и обновление БД. Он наследуется от класса платформы DbOpenHelper, который одним из параметров принимает сгенерированный на предыдущем шаге файл с описанием мобильной базы данных.

   Установите содержимое OptimumOpenHelper в

```
package ru.cdc.optimum.quickstart.example1;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import ru.cdc.optimum.db.DbOpenHelper;

public class OptimumOpenHelper extends DbOpenHelper{
      public OptimumOpenHelper(Context context, String name) {
            super(context, "xml1.xml", name, null, 1);
      }
      @Override
            public void onUpgrade(SQLiteDatabase db, int oldVersion, int
 newVersion) {
      }
}
```

Примечание: убедитесь в том, что имя xml-файла, указанное в конструкторе класса OptimumOpenHelper, строго совпадает с именем файла, который находится в папке assets.

5. Класс Example1Application замещает класс приложения по умолчанию. Он будет использоваться для создания базы данных, для передачи контекста приложения библиотеки (в общем – для инициализации библиотеки) и для доступа к базе данных.

   Установите содержимое Example1Application в

```
package ru.cdc.optimum.quickstart.example1;

import ru.cdc.optimum.Synchronization;
import android.app.Application;
import android.database.sqlite.SQLiteDatabase;

public class Example1Application extends Application {
```

```
	private OptimumOpenHelper helper;

	@Override
	public void onCreate() {
		super.onCreate();
		helper = new OptimumOpenHelper(this, "example1.db");

Synchronization.setApplicationContext(getApplicationContext());
	}

	public SQLiteDatabase db() {
		return helper.getWritableDatabase();
	}

 }
```

6. Создайте класс MainActivity. Этот класс представляет единственный экран (активность) приложения, в котором есть две кнопки **Login** и **Sync**. Также в этой активности реализованы функции обратного вызова библиотеки, которые будут показывать результат логина и синхронизации. После запуска приложения нужно нажать на кнопку **Login** (выполняется вход на сервер), и после успешного входа следует нажать на кнопку **Sync** (будет запущена синхронизация).

   Содержимое MainActivity установите в

```
package ru.cdc.optimum.quickstart.example1;

import ru.cdc.optimum.auth.AuthenticationCallback;
import ru.cdc.optimum.auth.AuthenticationResult;
import ru.cdc.optimum.auth.Credentials;
import ru.cdc.optimum.ConnectionParameters;
import ru.cdc.optimum.PlatformVariables;
import ru.cdc.optimum.Synchronization;
import ru.cdc.optimum.Synchronization.Error;
import ru.cdc.optimum.Synchronization.Result;
import ru.cdc.android.quickstart.R;
import android.os.Bundle;
import android.app.Activity;
import android.database.sqlite.SQLiteDatabase;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity
	implements OnClickListener, AuthenticationCallback,
Synchronization.Listener {

	private Button btnLogin;
	private Button btnSync;
	private String login;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		btnLogin = (Button)findViewById(R.id.btnLogin);
```

```
            btnLogin.setEnabled(true);
            btnLogin.setOnClickListener(this);

            btnSync =  (Button)findViewById(R.id.btnSync);
            btnSync.setEnabled(false);
            btnSync.setOnClickListener(null);
      }


    @Override
    protected void onResume() {
      super.onResume();
            Synchronization.registerSyncronizationHandler(this);
            Result result = Synchronization.getResult();
            if (result != null) {
                  onSynchronizationEnd(result);
            }
    }

      @Override
    protected void onPause() {
            Synchronization.unregisterSyncronizationHandler(this);
            super.onPause();
    }

      @Override
      public void onClick(View v) {
            switch (v.getId()) {
                  case R.id.btnLogin:
                        btnLogin.setOnClickListener(null);
                        btnLogin.setEnabled(false);
                        setSyncParameters("x", y, 300);
                        login("adventure-works\\login", "password");
                        return;
                  case R.id.btnSync:
                        startSync();
                        return;
            }
      }

      private SQLiteDatabase db() {
            return ((Example1Application)getApplication()).db();
      }

      private void setSyncParameters(String serverAddress, int
serverPort, int timeout) {
            ConnectionParameters cp =
                  Synchronization.getConnectionParameters();
            cp.setHostName(serverAddress);
            cp.setPortNumber(serverPort);
            cp.setConnectionTimeout(timeout * 1000);
            cp.setReadTimeout(timeout * 1000);
      }

      private void login(String loginString, String passwordString) {
            login = loginString;
            Credentials credentials =
            new Credentials(loginString, passwordString);
            Synchronization.authenticate(db(), credentials, true, this);

      }

      @Override
```

```
            public void onComplete(AuthenticationResult authResult) {
                  if (authResult.getCode() ==
AuthenticationResult.Code.SUCCESS) {
                        showToast("Successful authentification");
                        PlatformVariables.newInstance(db())
                        .set("@login", login).commit(db());
                        btnSync.setOnClickListener(this);
                        btnSync.setEnabled(true);
                  } else {
                        showToast("Authentification failed: " +
                              authResult.getCode().name());
                  }
            }

            private void startSync() {
                  Synchronization.execute(db(), "default");
            }

            @Override
            public void onSynchronizationStart() {
                  showToast("Synchronization started");
            }

            @Override
            public void onSynchronizationError(Error error) {
                  showToast("Synchronization error: " + error.name());
            }

            @Override
            public void onSynchronizationEnd(Result result) {
                  if (result != Result.FAIL) {
                        showToast(result.name());

                        String readableName =
PlatformVariables.getString(db(),
                              "@login");
                        if (readableName != null) {
                              Synchronization.setDeviceReadableName(db(),
                                    readableName);
                        }
                  }
            }

            private void showToast(String message) {
                  Toast toast = Toast.makeText(this, message,
Toast.LENGTH_LONG);
                  toast.show();
            }
      }
```

Примечание: в строке «setSyncParameters("10.0.2.2", 1126, 300);» укажите параметры подключения к Серверу синхронизации: x – ip-адрес сервера, на котором установлен Сервис синхронизации, y – порт.

7. В манифест приложения (AndroidManifest.xml) добавьте следующие разрешения:

```
      <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
      <uses-permission android:name="android.permission.INTERNET"/>
      <uses-permission
      android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

А также установите атрибут в теге application

```
android:name="ru.cdc.optimum.quickstart.example1.Example1Application"
```

В итоге манифест приложения должен принять следующий вид:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ru.cdc.android.quickstart"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="18" />

    <uses-permission
android:name="android.permission.READ_PHONE_STATE"/>
     <uses-permission android:name="android.permission.INTERNET" />
     <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"

android:name="ru.cdc.optimum.quickstart.example1.Example1Application"
        >
        <activity

android:name="ru.cdc.optimum.quickstart.example1.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

8. Создайте файл activity_main.xml и приведите его к следующему виду:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
```

```
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="38dp"
        android:text="Login" />

    <Button
        android:id="@+id/btnSync"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/btnLogin"
        android:layout_marginLeft="38dp"
        android:text="Sync" />

</RelativeLayout>
```

После выполнения всех перечисленных шагов будет получено мобильное приложение, которое создает базу данных со структурой, совпадающей с серверной.

Соединение и синхронизация

Проверьте соединение и синхронизацию с сервером. Для этого выполните следующие действия:

1. Запустите приложение. Запустится эмулятор. Из двух кнопок будет активна только **Login**. Нажмите на неё – будет выполнена попытка входа на сервер. Если попытка удачна, то появится сообщение «SUCCESS», а затем «Authentication successfull». Если вход не был выполнен, то появится сообщение об ошибке.
2. После успешного входа кнопка **Sync** станет активной. Нажмите на неё, запустится синхронизация. Если синхронизация прошла успешно, то появится сообщение «SUCCESS». Если синхронизация не прошла, то появится сообщение об ошибке.
3. В Eclipse откройте перспективу DDMS. Перейдите на вкладку File Explorer, и по пути /data/data/ru.cdc.android.optimum.quickstart.example1/databases найдите файл example1.db. Скачайте файл на компьютер и с помощью любого менеджера SQLite убедитесь, что таблицы Products и ProductModel заполнены данными.

Руководство разработчика серверной части

Определение синхронизируемых таблиц

Определение синхронизируемых таблиц осуществляется в Центре управления. Нужно создать таблицу, задать ее имя, свойства и набор полей, в которых будут храниться данные.

Приоритет синхронизации необходимо задавать в случае, если какие-либо таблицы должны быть переданы раньше/позже других, в противном случае можно всем таблицам проставить одинаковое значение.

Тип синхронизации задается исходя из назначения таблицы. Таблица может не синхронизироваться вообще, синхронизироваться одинаково для всех устройств или синхронизироваться с сегментацией.

Если таблица не синхронизируется, данные в нее приходят из КИС, но не уходят на мобильное устройство. Данные из этой таблицы, например, могут использоваться в запросах сегментации для связки данных других таблиц.

Типы «По дате изменения» и «По состоянию» определяют синхронизацию без сегментации. Это означает, что все данные из этой таблицы будут передаваться на все мобильные устройства. При этом процедуры определения дельты будут работать, и в каждом сеансе синхронизации будут передаваться только новые, удаленные, добавленные или измененные для конкретного устройства данные. В большинстве случаев для синхронизации без сегментации следует использовать вариант «По состоянию», этот вариант формирует кеш. Вариант «По дате изменения» не отслеживает удаление записей из синхронизируемой таблицы, поэтому может применяться только в случаях, когда из таблицы не удаляются данные или их удаление не критично для работы программы, и должен использоваться только в особых случаях для оптимизации работы платформы. Также этот вариант не предусматривает формирование кеша, соответственно служебные функции Get_Device для таких СТ будут отдавать пустые выборки.

Типы «Сегментация простая» и «Сегментация сложная» включают сегментацию для конкретного устройства. Это означает, что необходимо задать пользовательский алгоритм, который будет выбирать данные, необходимые для конкретного устройства. После чего к этой выборке будет применен алгоритм (встроенный) определения дельты, и измененные данные будут отправлены на устройство.

В случае простой сегментации надо написать один запрос на T-SQL, который вернет поля, входящие в ключ, желательно, в порядке вхождения полей в первичный ключ.

В случае сложной сегментации можно написать скрипт на T-SQL, который может реализовать сложный алгоритм. В результате отработки этого скрипта данные должны быть вставлены во временную таблицу, из которой их заберет платформа для дальнейшей обработки. Эту временную таблицу не нужно создавать пользователю, она существует в рамках процесса синхронизации. Имя этой таблицы: #sync_<имя СТ>.

Выбор стратегии сегментации данных

Для того чтобы выбрать, какой тип сегментации проставить таблице, надо представить, как будут использоваться данные таблицы.

Если все данные из таблицы должны присутствовать на всех устройствах, то поставьте таблице тип синхронизации без сегментации, например, «По состоянию».

Если на устройство должно передаваться только подмножество данных таблицы, то, как правило, для этого случая будет достаточно типа синхронизации «Сегментация простая».

Примеры сегментации

Простая сегментация

В таблице А лежат записи, надо отобрать только те, у которых поле F1 = Condition

Бизнес-пример: Записи о клиентах, поле F1 – код сотрудника, обслуживающего клиента. Надо отобрать только тех клиентов, которых обслуживает сотрудник-владельца МУ.

Ссылочная сегментация

В таблице А лежат записи, поле F1 ссылается на аналогичное поле в таблице В. В таблице В есть поле F2, надо отобрать записи из таблицы А по условию A.F1 = B.F1 AND B.F2 = Condition.

Бизнес-пример: Таблица А – список товаров, поле F1 – категория товара. Таблица В – категории товаров, поле F2 – признак категории. Например, «отобрать товары, у категории которых стоит признак «Сигареты».

Иерархическая сегментация

В таблице А лежат записи, есть два поля – F1 и F2. В таблице В есть поле F3, надо отобрать записи из таблицы В по условию F3 = Condition OR (F3 = F2 AND F1=Condition).

Бизнес-пример: Таблица А – список сотрудников. Поле F1 – код сотрудника, поле F2 – код его начальника. Таблица В – список клиентов, поле F3 – код обслуживающего клиента сотрудника. Надо отобрать всех клиентов, которые обслуживаются указанным сотрудником или его подчиненными.

У данной сегментации могут быть варианты. В терминах бизнес-логики, например, в А для начальника запись может отсутствовать или может присутствовать с F1=F2 («подчинен самому себе»). Может быть также различный уровень вложенности – например, региональный менеджер – ему подчинена команда районных менеджеров, каждому из них подчинена команда сотрудников.

Использование специальных функций для работы с данными

Для выборки данных в запросах сегментации необходимо использовать специальные функции платформы. Эти функции создаются для каждой таблицы и называются Get_Server_<наименование СТ> и Get_Device_<наименование СТ>.

Функция Get_Server_<наименование СТ> возвращает срез данных, которые в настоящий момент имеются на сервере. Возвращаются все поля таблицы.

Функция Get_Device_<наименование СТ> возвращает срез данных, которые есть на мобильном устройстве. Возвращаются только ключевые поля таблицы. Эта функция работает только в контексте сеанса синхронизации, поскольку возвращает данные для конкретного мобильного устройства. При использовании функций типа Get_Device_<наименование СТ> для СТ необходимо правильно указывать «Приоритет при передаче с сервера на МЧ». Например, на МУ передаются:

- товары согласно ассортименту, привязанному к сотруднику;
- прайс-листы – согласно договору клиента, по списку всех клиентов, переданных на МУ сотрудника;
- цены (большой справочник).

Для цен можно в сегментации использовать функции Get_Device_<наименование СТ>, возвращающие срез «как есть на данном МУ сейчас» по прайс-листам и товарам. Для того чтобы на

этапе синхронизации таблицы цен можно было воспользоваться функциями Get_Device_<наименование СТ>, необходимо расположить СТ в следующем порядке:

1. товары;
2. прайс-листы;
3. цены.

Использование переменных платформы

Переменные платформы могут использоваться в серверной части платформы, например, в целях сегментации. С точки зрения общей разработки приложения, их использование должно быть согласовано между серверной и мобильной частью. Это означает, что разработчик серверной части должен создать определенные переменные, назвать их определенным способом и использовать в своих серверных алгоритмах. Разработчик мобильной части, в свою очередь, должен заполнить переменные платформы на мобильном устройстве нужными значениями.

На текущий момент поддерживается только один тип переменных платформы – строка. Но допускается упаковка в строку значений других типов (при условии, что алгоритмы упаковки/распаковки будут поддерживаться пользователем).

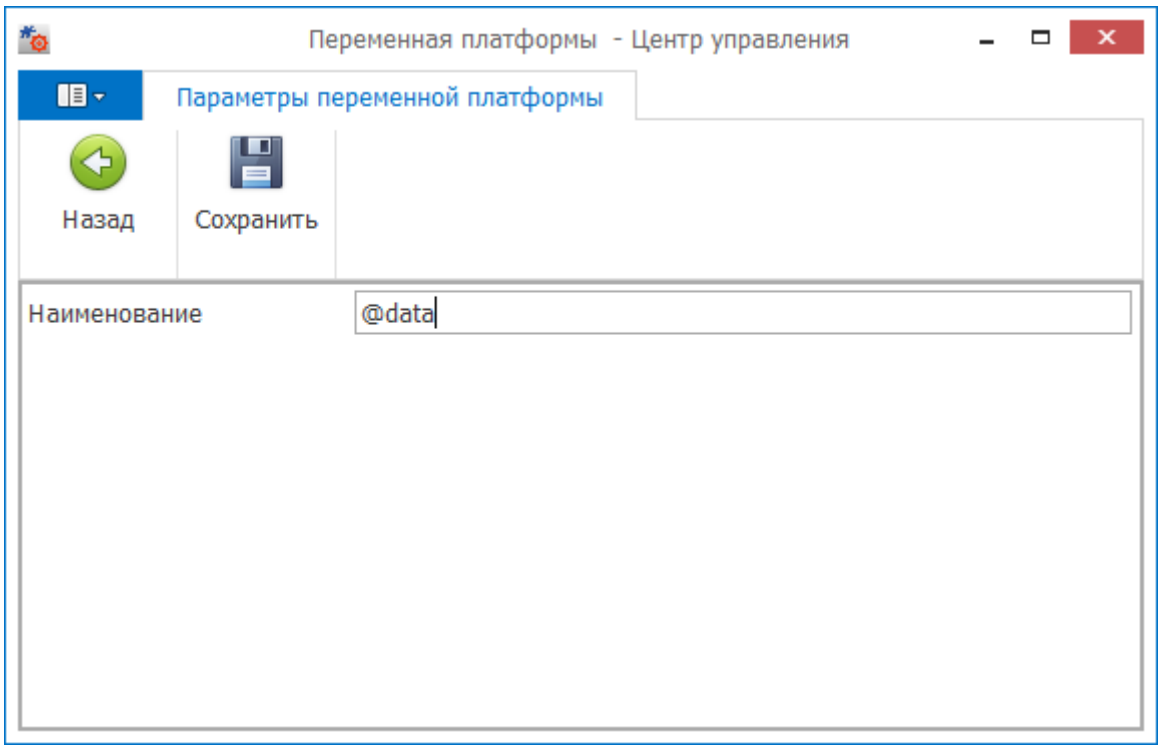
Имена ПП должны соответствовать ограничениям для имен локальных переменных T-SQL (язык запросов для MS SQL Server) и начинаться с символа «@».

Переменные могут быть использованы в тексте запросов сегментации. При этом в контексте процесса синхронизации конкретного устройства Система установит значение этой переменной, равное значению, установленному в МЧ для конкретного МУ. При этом, если в МЧ не установлено значение, Система установит переменной значение NULL.
Пример запроса сегментации с использованием переменной:

```
Select Employees.EmployeeID
        From Get_Server_Employee() As Employees
        inner join Get_Server_EmployeeLogin() As Logins
        on Employees.EmployeeID = Logins.EmployeeID
 Where LoginID = @login
```

Для создания переменной в Центре управления необходимо:

1. Открыть вкладку **Переменные** и нажать на кнопку **Добавить**.
2. В открывшейся вкладке **Параметры переменной платформы** необходимо ввести имя переменной и нажать на кнопку **Сохранить**.



Прямая вставка данных в синхронизируемые таблицы

В редких случаях, когда в КИС заложена логика определения дельты, вместо использования Сервиса обмена может понадобиться прямая вставка данных в кеширующую БД платформы. Для этого используется хранимая процедура EXCH_Table_Record_Set.

На вход процедура принимает три параметра – имя таблицы, xml-строку с данными и признак активности записей.

Признак активности записей показывает, что следует делать с данными: если он установлен, то данные добавляются или обновляются. Если он сброшен – данные будут удалены.

Xml-строка с данными содержит непосредственно данные для синхронизируемой таблицы. Строка может содержать произвольное количество записей. Минимальный набор записей – все ключевые поля таблицы. Из не ключевых полей могут передаваться только измененные поля (в случае обновления данных).

Формат строки с данными:

```
<root>
  <row>
    <field name="<Имя поля1>">Значение поля1</field>
    <field name="<Имя поля2>">Значение поля2</field>
  </row>
</root>,
```

где root – корневой элемент, row – строка таблицы, field – поле строки.

Значения NULL передаются как текст «NULL».

Процедура прямой работы с данными оказывает воздействие только на те записи, которые были переданы в параметры процедуры.

Генерация данных для разработчика мобильной части

После завершения разработки серверной части необходимо сгенерировать файл-описание БД и отдать его мобильному разработчику. Центр управления генерирует архив, который содержит файл-описание и необходимые библиотеки под выбранную платформу.[1]

Имя архива содержит следующие данные:

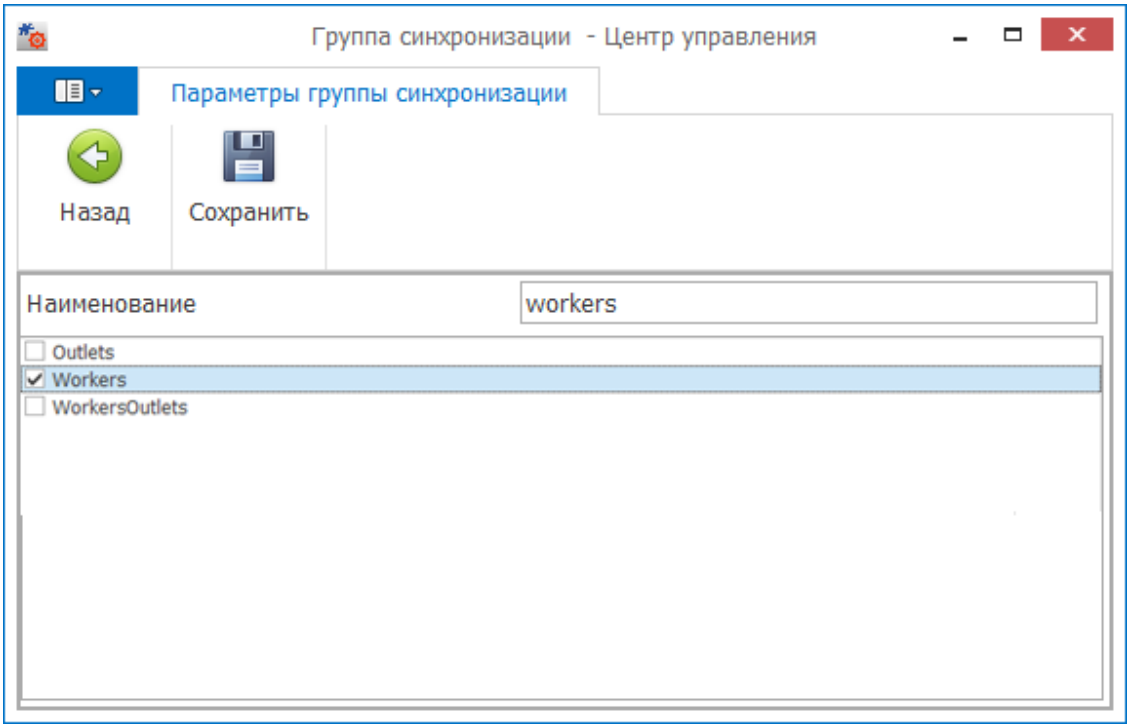
- Имя проекта платформы;
- Мобильная платформа;
- Дата формирования архива;
- Время формирования архива в секундах;

Пример имени архива*: AdvWorksMobile_test_1_iOS_20140827_132632.zip.*

Использование переменных платформы

Есть три таблицы – сотрудники, магазины и связи (какие сотрудники какой магазин обслуживают) с именами Workers, Outlets и WorkersOutlets соответственно. У сотрудников есть мобильные устройства, на которых будет установлена наша программа. При первом запуске программы на устройство должен загрузиться общий список сотрудников. После этого сотрудник должен выбрать себя из списка, и получить на устройство список магазинов, которые он обслуживает.

1. В Центре управления создаем три таблицы, аналогичные исходным.
2. Создаем группу синхронизации «workers».
3. Таблице Workers ставим тип синхронизации «По состоянию», таблице Outlets – «Сегментация простая», а таблице WorkersOutlets – «Не синхронизируется». Кроме этого, таблицу Workers добавляем в группу синхронизации «workers».



4. Создаем переменную платформы с именем «@worker_id».
5. Для таблицы «Outlets» устанавливаем запрос сегментации:

[1] Пакет для iOS необходимо распаковывать в Mac OS. Распаковка архива в Windows нарушит структуру файлов пакета.

```
Select distinct Outlets.outlet_id
From
        Get_Server_Outlets() As Outlets
      Inner Join Get_Server_WorkersOutlets() As WorkersOutlets
            On Outlets.outlet_id = WorkersOutlets.outlet_id
Where WorkersOutlets.worker_id = @worker_id
```

Обратите внимание, что вместо имен таблиц используются специальные функции. Таблиц с исходными именами в промежуточной БД не существует, и для обращения к данным должны использоваться именно специальные функции.

В результате такого построения серверной части, в мобильной части можно применить следующий алгоритм:

- Приложение регистрируется и проводит синхронизацию группы «workers». Поскольку в эту группу входит только таблица Workers, у которой стоит тип синхронизации «По состоянию» (т.е. без сегментации) – любому сотруднику отправится полный список сотрудников. Этот список можно отобразить на экранной форме.
- Пользователь выбирает себя из списка, и приложение записывает id выбранного сотрудника в переменную платформы «@worker_id», после чего запускает синхронизацию группы «default».
- В группу «default» входит две таблицы – Outlets и WorkersOutlets. Поскольку для WorkersOutlets установлен тип синхронизации «Не синхронизировать», для нее никаких действий и передачи данных осуществляться не будет. Для таблицы же Outlets сначала применится запрос сегментации, который отберет магазины, которые обслуживает работник с указанным в переменной worker_id. После чего к этим записям будет применен алгоритм определения дельты, и они будут отправлены на мобильное устройство.

Руководство разработчика мобильной части

Мобильная часть платформы в основном берет на себя задачи создания структуры мобильной БД по приведенному описанию, создание всех необходимых служебных структур и данных в этой БД, а также предоставляет алгоритмы определения дельты и методы синхронизации данных. При этом мобильная БД остается полностью открытой для разработчика, который может использовать прямые sql-запросы.

Файл-описание БД и библиотеки платформы

Для разработки мобильного приложения необходимо подключить предоставляемые библиотеки платформы и положить в ресурсы файл-описание БД.

Разработка под Android[2]

Примечание – полная информация по классам библиотеки платформы под Android приведена в отдельной документации – «Optimum API для Android». В этом разделе представлена краткая информация.

Работа с БД

Для корректной работы платформы нужно использовать экземпляр класса ru.cdc.optimum.db.DbOpenHelper или унаследовать от него свой DbHelper. DbOpenHelper принимает в качестве одного из параметров файл-описание БД, который был сгенерирован при разработке серверной части платформы. В остальном работа с классом не отличается от работы со стандартным android.database.sqlite.SQLiteOpenHelper.

Основные настройки

При старте приложения необходимо передать платформе контекст приложения, сделав следующий вызов:

```
Synchronization.setApplicationContext(getApplicationContext());
```

Чтобы задать параметры соединения с сервером, нужно получить экземпляр параметров соединения с помощью следующего вызова:

```
ConnectionParameters cp = Synchronization.getConnectionParameters();
```

После чего установить адрес сервера СС, номер его порта и, не обязательно, другие параметры – таймаут соединения, таймаут чтения и признак безопасного соединения.

```
cp.setHostName("10.0.2.2");
cp.setPortNumber(11126);
```

Параметры соединения с сервером нужно задавать при каждом запуске программы.

[2] В качестве примера в руководстве использована платформа Android, но общий подход для остальных мобильных платформ практически идентичен.

Синхронизация

Синхронизация всегда проводится асинхронно, в отдельном потоке. Для запуска синхронизации надо вызвать метод:

```
Synchronization.execute(db, "default");
```

Передав ему первым параметром экземпляр БД (может быть получен через DbHelepr.getWritableDatabase()), а вторым – группы синхронизации, которые должны быть просинхронизированы в этом сеансе. Имена групп синхронизации являются регистро-зависимыми.

Для отслеживания процесса синхронизации нужно использовать класс, реализующий интерфейс `Synchronization.Listener`.

Класс Synchronization предоставляет методы, позволяющие зарегистрировать и отменить регистрацию класса-слушателя, а также классы Result и Error, описывающие результат и ошибку синхронизации соответственно.

Поскольку синхронизация выполняется в отдельном потоке, она может проходить и завершиться в тот момент, когда не зарегистрирован ни один слушатель, который мог бы изменить состояние интерфейса соответствующим образом. Для обхода этой проблемы рекомендуется в месте регистрации слушателя вызывать метод Synchronization.getResult() и проверять результат последней синхронизации.

Переменные платформы

Для работы с переменными платформы библиотека предоставляет класс ru.cdc.optimum.PlatformVariables. Этот класс позволяет прочитать или записать значение переменной платформы по ее имени. После изменения значения переменной платформы обязателен вызов метода commit, который запишет изменения в базу.

Примеры

В качестве комплексного примера использования платформы предлагается мобильное приложение к бизнес-сценарию демонстрационной БД AdventureWorks.

Бизнес-сценарий AdventureWorks можно посмотреть по ссылке:

http://technet.microsoft.com/en-us/library/ms124825%28v=sql.100%29.aspx

Приложение автоматизирует деятельность торговых представителей компании «Adventure Works Cycles». Компания Adventure Works Cycles – вымышленная компания, описанная фирмой Microsoft для создания на её основе бизнес-процессов демонстрационной БД для MS SQL Server. Компания занимается производством и продажей велосипедов, запчастей и аксессуаров.

Под AdventureWorks стоит понимать как компанию, так и БД, в зависимости от контекста. Предполагается, что БД AdventureWorks является корпоративной информационной системой, для которой разработано мобильное приложение. Бизнес-сценарии AdventureWorks разделены условно на четыре области:

- Продажи и маркетинг;

- Товароведение;
- Закупки и производители (партнеры);
- Производство.

В приложении работа, в основном, ведется со сценарием «Продажи и маркетинг», однако используются таблицы и данные других сценариев.
В качестве клиентов компании выступают как частные лица, так и магазины (склады в терминологии AdventureWorks).

Продажи производятся по нескольким каналам – в том числе самостоятельные заказы через интернет и через торговых представителей.

Автоматизирована деятельность торговых представителей с допущением, что они работают только с магазинами, а не с частными лицами.

В целом, данные из AdventureWorks передаются в платформу с такой же или схожей структурой, хотя в некоторых случаях используется упрощение или трансформация данных с целью более полного задействования возможностей и сценариев использования платформы (исходные данные в AdventureWorks выстроены способом, оптимальным для демонстрации возможностей и сценариев использования MS SQL Server).

Приложение разработано для всех мобильных ОС, поддерживаемых платформой ОПТИМУМ.

Подробное описание комплексного примера содержится в отдельном документе «Описание демо-приложения.pdf», поставляемом с дистрибутивом платформы.

Рекомендации

В этом разделе рассмотрены некоторые рекомендации, проиллюстрированные на примере платформы Android. Эти рекомендации реализованы в примере, который поставляется в составе дистрибутива, для каждой из поддерживаемых мобильных платформ.

Рекомендации по реализации периодической автоматической синхронизации

В этом разделе приведен пример кода, который иллюстрирует возможный подход к реализации в мобильном приложении функции автоматической синхронизации с заданным периодом повторения.

Код реализован в классе `AutoSync`. Синхронизация запускается автоматически через каждые 15 минут. Отсчет времени в данном примере ведется с точностью до 5 секунд.

Модифицировав данный код, можно реализовать также настройку периода автоматической синхронизации пользователем через пользовательский интерфейс, а также функцию включения и выключения автоматической синхронизации.

```
import java.util.Date;
import android.os.Handler;

public class AutoSync {
  // 15 minutes
  private static final int AUTO_SYNS_PERIOD_IN_SECONDS = 15 * 60;

  // 5 seconds
  private static final int DELAY_INTERVAL_IN_SECONDS = 5;

  private Handler handler;
  private Runnable autoSyncRunnable;

  public AutoSync() {
    handler = new Handler();
    autoSyncRunnable = new AutoSyncTask();
  }

  public void start() {
    handler.removeCallbacks(autoSyncRunnable);
    handler.postDelayed(autoSyncRunnable,
                    DELAY_INTERVAL_IN_SECONDS * 1000);
  }

  public void stop() {
    handler.removeCallbacks(autoSyncRunnable);
  }

  private class AutoSyncTask implements Runnable {
    private long lastAttemptTime;

    public AutoSyncTask() {
      lastAttemptTime = System.currentTimeMillis();
    }

    @Override
    public void run() {
      int autoSyncPeriod = AUTO_SYNS_PERIOD_IN_SECONDS;
      long currentAttempt = System.currentTimeMillis();
      if (currentAttempt >= lastAttemptTime + autoSyncPeriod*1000) {
        lastAttemptTime = currentAttempt;

        // perform auto sync
        performSync();
      }
      handler.postDelayed(autoSyncRunnable,
```

```
                    DELAY_INTERVAL_IN_SECONDS*1000);
      return;
    }
  }

  private boolean performSync() {
    String serverAddress = . . . //load from preferences
    int serverPort =. . . //load from preferences

    // set parameters
    ConnectionParameters cp = Synchronization.getConnectionParameters();
    cp.setHostName(serverAddress);
    cp.setPortNumber(serverPort);
    cp.setConnectionTimeout(300*1000); // 300 sec
    cp.setReadTimeout(300*1000); // 300 sec


    if (Synchronization.getResult() == null) {
      // sync is already started
      return false;
    }

    // start sync
    Synchronization.execute(. . . );
    return false;
  }
}
```

Чтобы инициализировать в приложении автоматическую синхронизацию, после успешного логина на сервер следует создать экземпляр класса и AutoSync и однократно вызвать его метод start().

AutoSync autoSync = new AutoSync();
autoSync.start();

Далее синхронизация будет запускаться автоматически с заданным периодом. Если же требуется прекратить автоматическую синхронизацию, то следует вызвать метод autoSync.stop().

Рекомендации по отправке логов и другой технической информации в службу техподдержки

В этом разделе приведен пример кода, который иллюстрирует возможный подход к реализации в мобильном приложении функции автоматического формирования файла данных для последующей отправки по электронной почте письма в службу техподдержки продукта.

Формируемый файл данных для техподдержки содержит копию мобильной БД, лог, извлеченный из LogCat, а также дополнительный текстовый файл, в котором содержится информация о версии программы и некоторые технические характеристики мобильного устройства (модель, версия ОС, объем памяти и т.п.).

Код реализован в классе `SupportFile`. Формирование файла данных и инициализация отправки письма реализованы в методе `sendEmailToSupport`, в который в качестве аргумента передается `Context` (это должен быть контекст активности). Файл данных представляет собой ZIP-архив, который создается на SD-карте мобильного устройства. После записи файла с данными формируется Intent на отправку e-mail в службу техподдержки. Отправка e-mail выполняется через стандартное приложение электронной почты, которое установлено на устройстве. Файл с данными прикладывается к сформированному письму в качестве аттачмента. Пользователю остается только ввести текст сообщения (необязательно) и отправить письмо.

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Date;
import android.app.ActivityManager;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Environment;

public class SupportFile {
  public static void sendEmailToSupport(Context context) {
    File sdcardDir  = Environment.getExternalStorageDirectory();
    File dbFilename = new File(sdcardDir, "demo.db3");
    String dbCopyFilename = dbFilename.getAbsolutePath();

    // copy Database to file
    String dbPath = context.getDatabasePath(LOCAL_DATABASE_NAME).getPath();
    copyFile(dbPath, dbCopyFilename);

    String attachmentFilename = createSupportFile(context, dbCopyFilename);
    if (attachmentFilename == null) {
      return false;
    }
    sendEmail(context, attachmentFilename);
    return true;
  }

  private static void sendEmail(Context context, String attachmentFilename) {
    final Intent emailIntent =
      new Intent(android.content.Intent.ACTION_SEND);

    emailIntent.setType("plain/text");
    emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
      new String[]{ "login@server.com"});
    emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
      "Support");
    emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
      "Support file");
    emailIntent.putExtra(Intent.EXTRA_STREAM,
      Uri.parse("file://" + attachmentFilename));

    context.startActivity(Intent.createChooser(emailIntent,
      "Send mail..."));
  }

  private static String createSupportFile(Context context,
String dbCopyFilename) {

    File sdcardDir  = Environment.getExternalStorageDirectory();
    final String root = sdcardDir.toString();
    final File zip = new File(root, "support_data.zip");

    // list off support files
    final ArrayList<File> files = new ArrayList<File>();

    // add database file
    files.add(new File(dbCopyFilename));

    // add logcat file
    File lcFilename = new File(root, "LogCat.txt");
    if (extractLogCat(lcFilename)) {
      files.add(lcFilename);
    }

    // add system info file
    try {
      File oiFilename = new File(root, "sysinfo.txt");
```

```
      FileWriter fw = new FileWriter(oiFilename);
      fw.write(context.getVersionName());
      fw.write("\r\n");

      ActivityManager am =
        (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
      fw.write(am.getMemoryClass() + " Mb memory available\r\n");

      for (Field f : android.os.Build.class.getDeclaredFields()) {
        try {
          fw.write("\r\n");
          fw.write(f.getName() + " = " + f.get(String.class));
        } catch (Exception ex) {
        }
      }
      fw.close();
      files.add(oiFilename);
    } catch (IOException e) {
      // info file generation failed
    }

    // prepare archive
    boolean fileSaved = Compress.zip(files, zip.getAbsolutePath());

    // remove temporary files
    if (fileSaved == true) {
      for (File file : files) {
        file.delete();
      }
    }
    return fileSaved ? zip.getAbsolutePath(): null;
  }

  private String getVersionName(Context context) {
    try {
      return context.getPackageManager()
        .getPackageInfo(getPackageName(), 0)
        .versionName;
    } catch (NameNotFoundException e) {
      return null;
    }
  }

  private static boolean extractLogCat(File filename) {
    try {
      if (filename.exists() == false) {
        filename.createNewFile();
      }
      String cmd = "logcat -d -v time -f " + filename.getAbsolutePath();
      Runtime.getRuntime().exec(cmd);
    } catch (IOException e) {
      // LogCat extracting failed
      return false;
    }
    return true;
  }
}
```

Примечание: исходный текст метода copyFile() здесь не приводится, поскольку он не представляет интереса для данного примера. Полный исходный текст представлен в прилагаемом примере.

Для создания zip-архивов в приведенном выше примере используется класс `Compress`.

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.util.Collection;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class Compress {
  private static final String COMPRESS_TAG = "Compress";
  private static final int BUFFER = 2048;

  public static boolean zip(Collection<File> files, String zipFile) {
    try {
      ZipOutputStream out = null;
      try {
        out = new ZipOutputStream(
                new BufferedOutputStream(
                  new FileOutputStream(zipFile)));

        byte data[] = new byte[BUFFER];
        for(File file : files) {
          if (file.exists() == false) {
            continue;
          }
          BufferedInputStream origin = null;
          try {
            origin = new BufferedInputStream(
                       new FileInputStream(file), BUFFER);

            ZipEntry entry = new ZipEntry(file.getName());
            out.putNextEntry(entry);

            int count;
            while ((count = origin.read(data, 0, BUFFER)) != -1) {
              out.write(data, 0, count);
            }
            out.closeEntry();
          } finally {
            if (origin != null) origin.close();
          }
        }
      } finally {
        if (out != null) out.close();
      }
      return true;
    } catch (Exception e){
      return false;
    }
  }
}
```

Ролевая модель данных

Передача данных в зависимости от роли пользователя.

Задача передачи данных в зависимости от роли пользователя может решаться несколькими способами.

Во-первых, поддержка ролевой модели может быть заложена в структуру данных и обеспечиваться кодом клиентского приложения. Например, если используются роли «Продавец» и «Менеджер», то приложение менеджера может знать о дополнительном наборе синхронизируемых таблиц (их можно выделить в отдельную группу синхронизации), запрашивать и отображать данные по ним.

Такой подход имеет смысл использовать, когда одна роль использует расширенный набор данных по сравнению с другой ролью. Под расширенным набором данных в данном случае имеются в виду дополнительные структуры данных (синхронизируемые таблицы), отсутствующие у более простой роли.

Во-вторых, поддержка ролевой модели может быть обеспечена на сервере путем предоставления большего набора данных, содержащихся в одних и тех же структурах данных. В таком случае структура и клиентские приложения для разных ролей совпадают (могут совпадать). Например, есть роль «Продавец», обладающая данными по продажам конкретного сотрудника, и роль «Менеджер», обладающая данными по продажам всех сотрудников, подчиненных конкретному менеджеру. Структура данных не меняется в зависимости от роли, меняется лишь их объём.

Определение роли, назначенной конкретному устройству происходит на основе каких-то данных, введенных первоначально на устройстве – например, логина. Введенный на МУ логин сохраняется в переменную платформы, которая при ближайшей (начальной) синхронизации передается на сервер. Предполагая, что на сервере есть соответствие между логинами и ролями пользователей, находим по логину роль.

Далее вводим сегментацию в синхронизируемые таблицы, которые предполагают разные наборы данных в зависимости от роли.

Например, сегментация таблицы с продажами Sales может выглядеть следующим образом:

```
If @Role = 'Manager'
Begin
        Select distinct Sales.ID
        From Get_Server_Sales() As Sales
        Inner join Get_Device_Customer() As DevCust
        On Sales.CustomerID = DevCust.CustomerID
        Left Join Get_Server_WorkersHierarchy() As WH
        On Sales.PersonID = WH.EmployeeID
        Where Sales.EmplID = @EmployeeID or WH.ManagerId = @EmployeeID
End
Else
Begin
        Select distinct Sales.ID
        From Get_Server_Sales() As Sales
        Inner join Get_Device_Customer() As DevCust
        On Sales.CustomerID = DevCust.CustomerID
        Where Sales.EmplID = @EmployeeID
End
```

В данном примере предполагается, что есть таблица Sales, с полем, содержащим ID сотрудника, создавшего продажу. На мобильном устройстве устанавливается переменная платформы @EmployeeID, содержащая ID сотрудника, зарегистрированного на этом устройстве (это упрощение, скорее всего эффективней получать эту переменную в запросе сложной сегментации) и переменная платформы @Role, определяющая роль пользователя. И есть таблица WorkersHierarchy, содержащая ID менеджера и подчиненных ему сотрудников.
В случае, если отбираются данные для роли «Менеджер», в результат будут включены все продажи менеджера и продажи подчиненных ему сотрудников, а для роли «Продавец» - только продажи данного сотрудника.

Рекомендации по построению высоконагруженной, отказоустойчивой, масштабируемой системы

Для построения высоконагруженной, отказоустойчивой, масштабируемой системы можно использовать программные решения:

1. Отказоустойчивый кластер SQL Server (http://msdn.microsoft.com/en-us/library/hh231721.aspx);

2. Компонент балансировки сетевой нагрузки (NLB) операционной системы Windows Server (http://technet.microsoft.com/en-us/library/hh831698.aspx).

Настройка этих программных средств отличается для разных версий продуктов MS SQL Server и Windows Server. Ссылки на документацию производителя приведены для примера и ссылаются на статьи документации производителя, относящиеся к версиям продуктов MS SQL Server 2014 и Windows Server 2012. Детальные инструкции по настройке других версий можно найти на сайте производителя (http://msdn.microsoft.com).

Отказоустойчивый кластер SQL Server используется для обеспечения бесперебойной работы, высокой доступности кеширующей БД.

Балансировка сетевой нагрузки позволяет распределять входящие запросы МУ между отдельными копиями СС, установленными на разных узлах в кластере. За счет использования двух и более серверов, объединённых в единый виртуальный кластер NLB, повышается доступность и масштабируемость СС. Трафик сетевого протокола TCP\IP синхронизации мобильной и серверной частей распределяется по используемым узлам кластера.

Узлы виртуального кластера NLB могут быть использованы для установки дополнительных отдельных копий СО и СЛ.