



**TranzAxis Manager**  
**Справочное руководство**

27 июня 2019 г.

© ООО "Компас Плюс", 1998-2019

## Лист регистрации изменений

Версия	Дата	Автор	Описание
2.0.0	21.09.2018	И. Гредяев	Начальная версия
2.0.1	26.09.2018	И. Гредяев	Обновлены пункты <ul style="list-style-type: none"><li>• 3.4.6 – описан новый тип редактора EntityRef.</li></ul> Добавлены пункты: <ul style="list-style-type: none"><li>• 3.6.5 – описан модуль ярлыков команд.</li></ul>
2.0.2	28.09.2018	И. Гредяев	Обновлены пункты <ul style="list-style-type: none"><li>• 4.6.1 – добавлено описание команды редактора свойства «Релизная версия» для автоматического выбора подходящей ссылки.</li></ul>
2.0.3	07.10.2018	И. Гредяев	Обновлен раздел 3.6.5
2.2.1	27.06.2019	И. Гредяев	Внесены все изменения с версии 2.0.3 до 2.2.1

## 1. Введение

Данный документ является технической спецификацией приложения TranzAxis Manager и платформы, на базе которой приложение построено. Приложение предназначено для упрощения процедуры сопровождения процесса разработки продуктов на базе платформы RadixWare (TranzAxis) и разнообразных конфигураций его запуска.

## 2. Обозначения

Важный пункт, на который следует обратить внимание.

**Наименование свойства** – свойство сущности, обязательное для заполнения.

**Наименование свойства** – свойство сущности, НЕ обязательное для заполнения, либо уже имеющее значение по умолчанию, либо пустое значение не предусмотрено прикладным типом данных.

**Наименование свойства** – свойство сущности доступное для заполнения при определенных условиях.

**Наименование свойства** – динамическое свойство сущности, не редактируется.

**[Селектор | Редактор] Наименование свойства** – свойство отображается только в селекторе или редакторе соответственно.

### 3. Платформа приложения

Основная задача платформы - реализовать конструктор объектов конфигурации и исполнения команд над данными объектами, а также предоставить удобный интерфейс для взаимодействия пользователя с объектами конфигурации.

Исходные коды платформы доступны по адресу: <https://github.com/graphixx-mgn/codex>

### 3.1. Основные понятия

В основе платформы лежат ряд объектов общего назначения, напрямую не связанных с конкретной прикладной задачей, но которые могут быть использованы для конструирования приложения:

- [Сущности, каталоги и параграфы](#)
- [Свойства](#)
- [Типы данных и редакторы](#)
- [Команды сущностей](#)
- [Модули](#)
- [Сервисы](#)

## 3.2. Сущности, каталоги и параграфы

### 3.2.1 Сущность

**Сущность** является моделью некоего объекта (ближайший аналог – класс в ООП), содержит в себе следующие компоненты:

- Иконку для визуального распознавания типа сущности
- Модель сущности с набором системных и прикладных свойств
- Набор прикладных команд обработки сущности
- Системные обработчики, реализующие поведение пользовательского интерфейса сущности.

Если модель сущности содержит хотя бы одно незаполненное обязательное свойство, такая сущность считается некорректной и команды будут недоступны.

Сущность может быть заблокирована, в частности на время выполнения длительных задач, в процессе исполнения которых редактирование сущности недопустимо. В этом случае команды также недоступны.

### 3.2.2 Каталог

**Каталог** является наследником сущности и должен использоваться в качестве контейнера других сущностей, в связи с этой задачей каталог обладает собственными специфическими особенностями и функциями:

- невозможность изменения наименования (поскольку каталог предположительно создается программно).
- отображение селектора дочерних объектов при просмотре каталога в проводнике.
- наличие механизма автоматической загрузки дочерних объектов.

### 3.2.3 Параграф

**Параграф** является наследником каталога и размещается на втором уровне в дереве проводника. Представляет собой корень дерева сущностей, связанных единой бизнес задачей.

### 3.3. Свойства

**Свойство** – модель некоего признака сущности (ближайший аналог – свойство в ООП), содержащего внутри себя значение данного признака, в соответствии с одним из прикладных типов данных.

Свойства подразделяются на хранимые и динамические. Хранимые редактируются пользователем или программно, динамические – только программно, согласно алгоритму вычисления значения. Динамические свойства могут зависеть от других свойств, в этом случае при изменении зависимых свойств данное динамическое значение пересчитывается.

Свойства могут содержать признак обязательности наличия значения, в этом случае пустое значение в нем интерпретируется как ошибка.

Свойства имеют следующие ограничения области видимости:

- Свойство отображается в редакторе и селекторе проводника
- Свойство отображается только в редакторе – селектор не должен быть перегружен информацией, как следствие, некоторые не значимые свойства можно опустить.
- Свойство отображается только в селекторе – возможное применение: создание динамических свойств, агрегирующих данные из редакторов.
- Свойство нигде не отображается – применимо для хранимых свойств, заполняемых программно и хранящих внутренние данные.

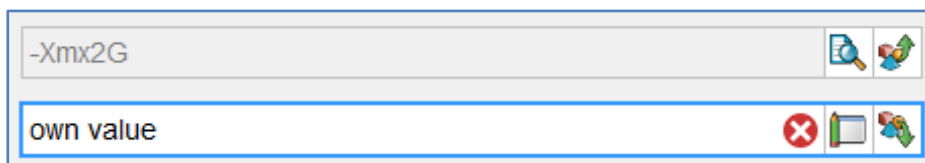
Свойства объединяются в модель сущности, являющуюся техническим контейнером, обеспечивающую жизненный цикл свойства:

- Хранение истории изменения свойств.
- Сохранение или откат изменений хранимых свойств.
- Расчет последовательности вычисления значений зависимых динамических свойств.
- Производство редакторов свойств.
- Генерация событий изменения значений.
- Хранение настроек видимости свойства в редакторе и селекторе.



### 3.3.1 Перекрытие свойств

Поскольку сущности образуют древовидную структуру, появляется возможность наследования значения из родительской сущности. В случае если родительская сущность имеет одноименное свойство того же типа, редактор свойства расширяется командой переключения режима наследования. По умолчанию, значение наследуется, но возможно перекрыть значение у дочерней сущностей, указав собственное, которое не теряется при возвращении унаследованного значения. В момент перекрытия, значение родительской сущности копируется, после чего может быть изменено пользователем.



Изображение 3.3.1 - пример унаследованного и перекрытого свойства

## 3.4. Типы данных и редакторы

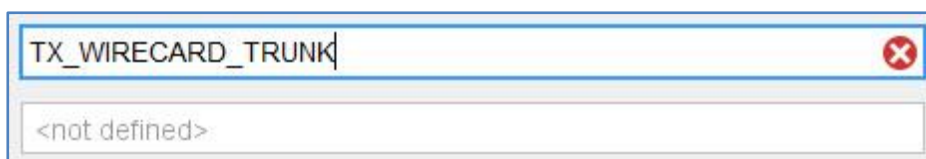
Все свойства сущностей различаются типами данных, различия между ними в основном определяет внешний вид и поведение редактора свойства данного типа. В основе каждого прикладного типа данных лежит тип данных Java, поэтому фактически можно считать каждый прикладной тип своеобразной «оберткой» типа Java.

Редактор свойства – визуальный компонент, предназначенный для ввода значения свойства. В дополнение к наименованию свойства имеют расширенное описание, которое появляется в виде подсказки при наведении мыши на наименование свойства на форме.

### 3.4.1 Str (Java-тип – java.lang.String)

Тип **Str** предназначен для хранения текстовой строки. Поле поддерживает маски ввода, для проверки корректности значения.

Редактор представлен в виде поля ввода. На изображении представлен вид редактора без фокуса ввода и редактируемого в данный момент.

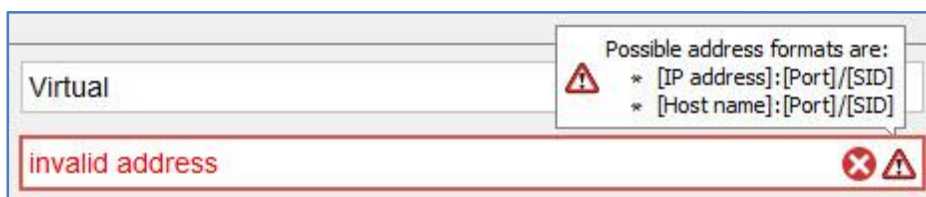


Изображение 3.4.1 - редактирование свойства Str

Если свойство имеет значение, при выделении поля справа появляется кнопка удаления значения.

Поле ввода текста в процессе изменения значения не отслеживает различия между изначальным значением и текущим, поскольку потребуется динамически сверять значение с маской, что может быть, во-первых, ресурсоемко, а во-вторых, раздражать пользователя. Фиксация изменения значения происходит в момент потери фокуса ввода или по нажатию клавиши ENTER.

Тип данных поддерживает установку маски ввода, которая ограничивает диапазон допустимых значений и содержит краткое пояснение как следует заполнить поле в соответствии с маской. Если текущее значение не соответствует маске – поле подсвечивается красным цветом и появляется значок предупреждения, нажав на который можно увидеть пояснение:



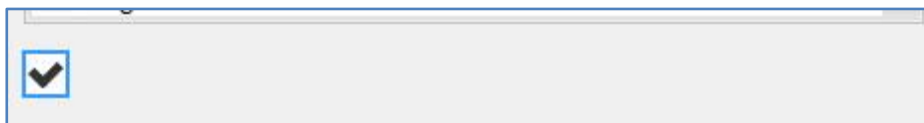
Изображение 3.4.2 – маска ввода с регулярным выражением

### 3.4.2 Int (Java-тип – java.lang.Integer)

Тип **Int** предназначен для хранения чисел и унаследован от типа Str, соответственно, его редактор практически не отличается от редактора Str, за исключением того, что он позволяет вводить только цифры и значение ограничено диапазоном значений Java-типа Integer.

### 3.4.3 Bool (Java-тип – java.jang.Boolean)

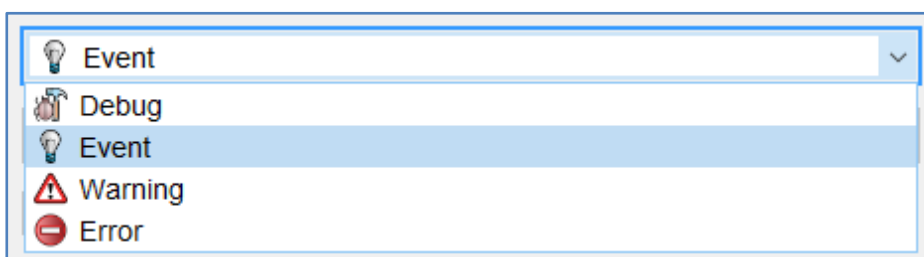
Тип **Bool** предназначен для хранения булевых значений, при этом NULL-значение интерпретируется как FALSE. Редактор представляет собой флаг, переключаемый по щелчку мыши:



Изображение 3.4.3 – редактирование свойства Bool

### 3.4.4 Enum (Java-тип – java.jang.Enum<?>)

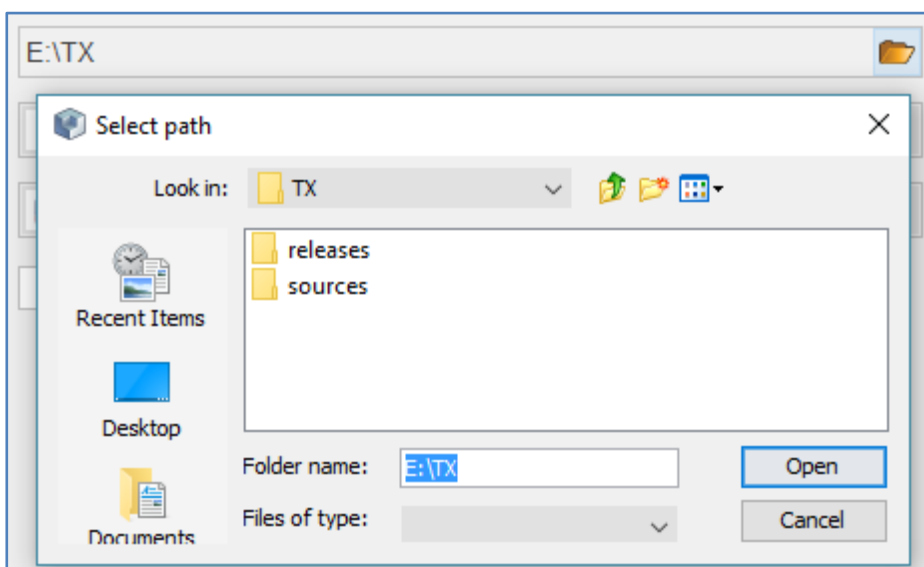
Тип **Enum** предназначен для хранения значения из набора констант. Редактор представляет собой выпадающий список со всеми значениями в данном наборе, при этом NULL-значение недопустимо.



Изображение 3.4.4 - выбор значения свойства Enum

### 3.4.5 FilePath (Java-тип – java.nio.file.Path)

Тип **FilePath** предназначен для хранения пути к файлу или директории. Редактор представляет собой не редактируемое поле ввода, в котором отображается выбранное значение и команду редактора для запуска диалога выбора нужного объекта файловой системы. По умолчанию, можно выбирать любой объект, но также возможно указание (в прикладном коде) фильтра объектов (директории, файлы или файлы с определенными расширениями):



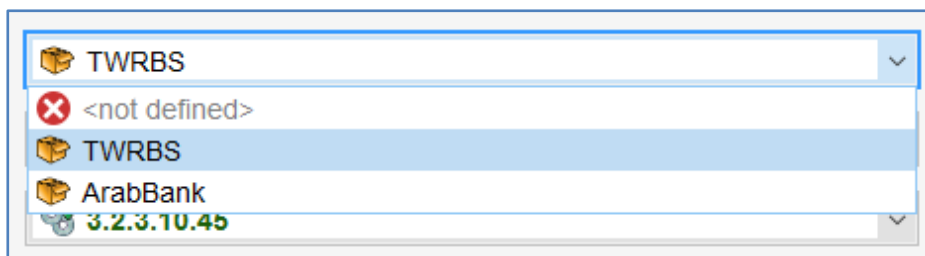
Изображение 3.4.5 – выбор каталога файловой системы в качестве значения свойства FilePath

### 3.4.6 EntityRef (Java-тип – Entity)

Тип EntityRef хранит в качестве значения ссылку на другую сущность, при создании свойства указанного типа указывается класс сущности и, если необходимо – условие отбора сущностей. Тип имеет 2 различных редактора.

#### 3.4.6.1 Стандартный редактор выбора в выпадающем списке.

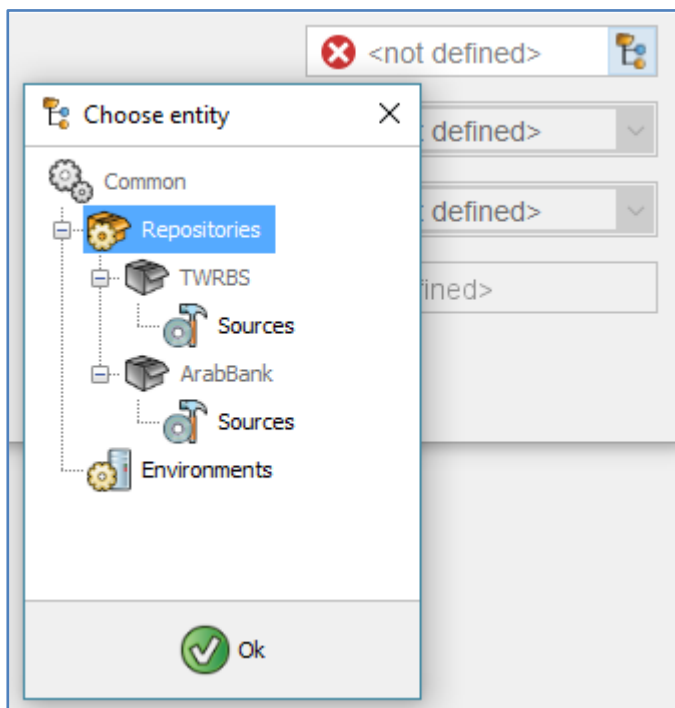
Редактор свойства представляет собой выпадающий список сущностей указанного класса и удовлетворяющих условию отбора. Первый элемент списка содержит специальное значение-маркер для сброса значения:



Изображение 3.4.6 – выбор сущностей в выпадающем списке

#### 3.4.6.2 Диалог выбора из дерева.

Редактор свойства представляет собой не редактируемое поле, отображающее выбранную сущность и команду вызова диалога с деревом объектов. Перед вызовом объекты предварительно фильтруются условием отбора, а затем выстроены в дерево, при этом необходимые узлы для построения всей структуры тоже добавляются, но недоступны для выбора.



Изображение 3.4.7 – выбор сущностей в дереве

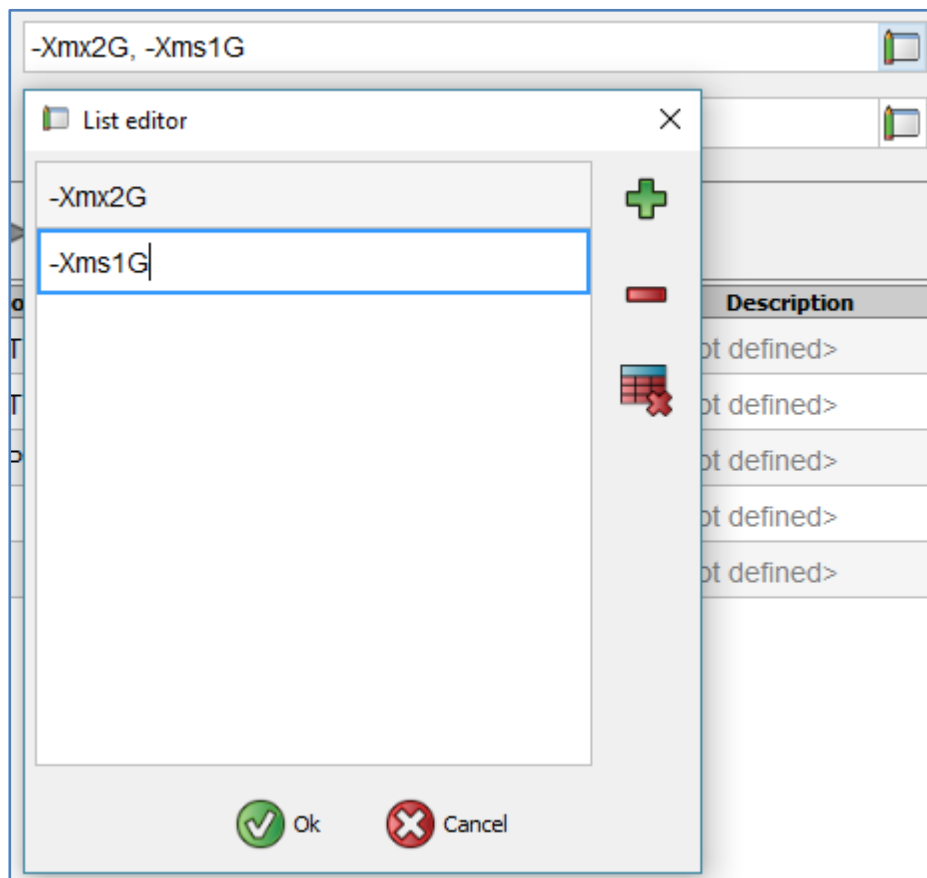
К примеру, на изображении показан диалог выбора каталогов, при этом ни корень дерева, ни сущности – владельцы каталогов «Sources» не являются каталогами, но они необходимы для построения древовидной модели, понятной пользователю.

### 3.4.7 ArrStr (Java-тип – java.util.List<String>)

Тип **ArrStr** предназначен для хранения списка строк. При этом, внешний вид и поведение редактора может кардинально меняться в зависимости от применяемой маски ввода относительно стандартного варианта.

#### 3.4.7.1 Стандартный редактор ввода строк.

Редактор представляет собой поле ввода, недоступное для непосредственной модификации, которое содержит все добавленные строки, разделенные запятыми. При выделении поля появляется кнопка очистки списка (аналогично редактору типа Str). Сам список формируется в диалоге, вызываемом командой редактора. В диалоге можно добавлять ячейки (пустые строки) и редактировать их, удалять выбранную строку или удалять весь список.

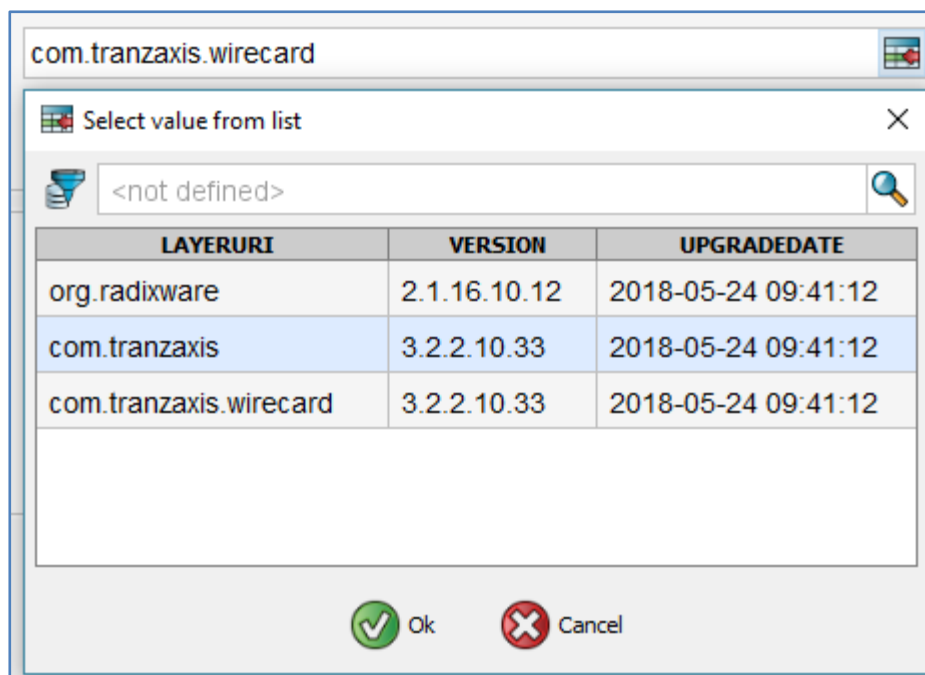


Изображение 3.4.8 – редактирование списка строк в модальном диалоге

Если свойство недоступно для редактирования, диалог будет доступен, но только в режиме просмотра, т.е. добавление, удаление и редактирование значений будет недоступным.

#### 3.4.7.2 Выбор строк из внешнего источника

Свойству может быть назначена маска **DataSetMask**, которая заменяет команду вызова диалога редактирования на вызов селектора строк из внешнего поставщика. Реализация внешнего поставщика данных может быть различной, но в общем случае он заполняет строки таблицы и предлагает выбрать одну из записей, в результате строки, обозначенные в колонках, попадут в виде списка в значение свойства. На изображении ниже представлен пример внешнего поставщика данных (результат запроса к БД):



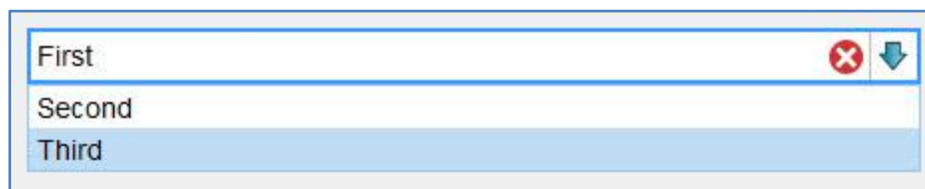
Изображение 3.4.9 – выбор значения из результата запроса в БД

Диалог выбора данных включает в себя поисковый запрос. Поиск ведется во всех ячейках таблицы, а именно – ищется вхождение введенной строки в значениях ячеек. Запускается поиск по кнопке или нажатием клавиши ENTER. В результате, таблица отображает только те записи, где указанный текст был найден. Если в результате поиска была найдена единственная запись, она автоматически выбирается.

Помимо модификации способа ввода, данная маска изменяет вид отображения строк в поле редактора. Вместо списка строк, разделенных запятыми, указывается формирующая строка с указанием места вставки элементов списка, согласно их индексам. Это обусловлено в первую очередь благодаря тому, что записи таблицы, а значит и результирующий список имеют фиксированное количество элементов.

### 3.4.7.3 Выбор значения из предустановленного списка

При установке маски **StrSetMask**, редактор свойства ведет себя как **Str**, который запрещает свободный ввод, но предлагает выбрать значение из предустановленного списка возможных значений в выпадающем списке. При этом список предустановленных значений, как и выбранное из них – хранятся в списке. Данный список можно формировать и модифицировать программно.

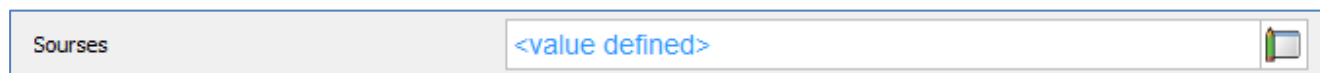


Изображение 3.4.10 – выбор значения из предустановленного списка

## 3.4.8 Map (Java-тип – java.util.Map)

Тип хранения пар ключ-значение. При этом, как ключи, так и значения могут быть любым из прикладных типов данных, описанных в данном разделе, кроме самого типа Map и AnyType (поскольку он не является сериализуемым типом, а значит значение карты будет невозможно сохранить в БД).

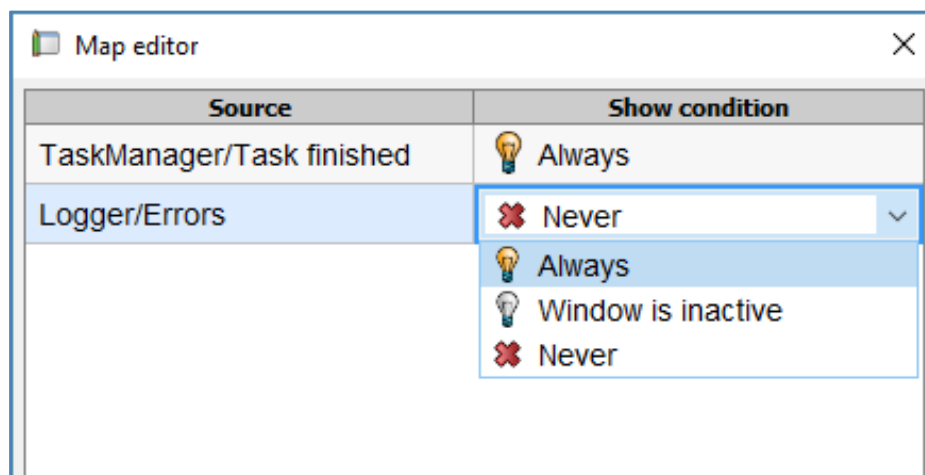
Редактор представляет собой поле ввода, недоступное для непосредственной модификации, которое содержит метку о наличии значения или его отсутствии.



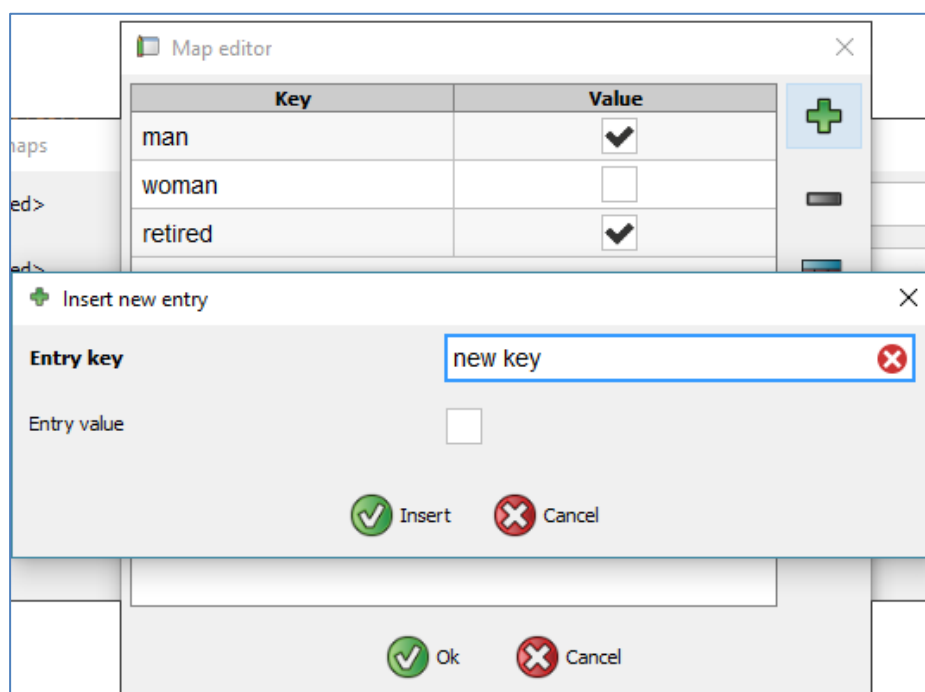
Изображение 3.4.11 – Карта имеет значения

Редактирование и просмотр карты осуществляется в модальном окне. Редактор имеет два режима работы:

- Редактирование только значений карты
- Редактирование значений, а также добавление / удаление ключей и полная очистка карты



Изображение 3.4.12 – Редактирование значений



Изображение 3.4.13 – Добавление ключа-значения

### 3.4.9 AnyType (Java-тип – java.lang.Object)

Специальный тип данных, предназначенный исключительно для визуализации в GUI произвольных объектов. В связи со своей функцией, изменение значения свойства такого типа

возможно только программным способом. Основной областью использования свойств данного типа является вывод той или иной информации об объектах.

### 3.4.9.1 Стандартная метка с иконкой

Стандартный редактор типа представляет собой метку, содержащую строковое представление объекта, а если объект реализует интерфейс **Iconified**, то и его иконку



Изображение 3.4.14 – вывод строк с иконками

### 3.4.9.2 Текстовый блок

Зачастую необходимо иметь возможность отобразить объемную текстовую информацию, для этого может использоваться нестандартный редактор свойства данного типа – текстовый блок:



Изображение 3.4.15 – вывод многострочного текста



## 3.5. Команды сущностей

Команды являются, пожалуй, самой важной частью платформы, поскольку именно в командах закладывается функциональность сущностей, раскрывается предназначение каждой из них.

Набор доступных команд определяются в классе сущности, включая в себя:

- Иконку для визуального представления в GUI
- Наименование и детальное описание
- Пользовательское условие применимости
- Параметры команды – опционально.
- Код команды, реализующий её логику.

Команды могут быть вызваны как из редактора сущности, так и из селектора сущностей. В каждом случае, команда получает контекст исполнения – список сущностей, в первом случае в списке будет только один объект, во втором – выбранные в селекторе сущности. Прикладные команды сущности определяются разработчиком в классе и могут объединяться в группы, если имеют близкий по смыслу функционал. В этом случае первая команда в группе отображается в виде кнопки, а все остальные – в качестве пунктов выпадающего меню.

### 3.5.1.1 Типы команд

Команды делятся на типы по принципу вызова для контекста:

- Одиночные – вызываются для каждого элемента контекста последовательно.
- Групповые – вызываются один раз для всех элементов контекста.

Также они подразделяются по типу:

- Прикладные – основной тип команд, реализующих основные бизнес-задачи конкретной сущности.

Только этот тип команд доступен в модуле ярлыков ([Launcher](#)).

- Административные – служебные команды, предназначенные не для повседневного использования, а для выполнения сложных манипуляций, затрагивающих к примеру, список сущностей, их родительский объект и т.д. Данный тип команд не доступен в модуле ярлыков.
- Информационные – команды, предназначенные для отображения какой-либо информации о сущности и/или на получения таковой на основе свойств сущности. Код команды не должен менять свойства сущности. Также этот тип команд не доступен в модуле ярлыков.
- Родительские – специальный тип команд, но в отличие от предыдущих типов, в качестве контекста команды всегда выступает родительская сущность. Соответственно, данная команда во-первых – доступна только в селекторе сущностей, а во-вторых – может быть исполнена вне зависимости от наличия дочерних сущностей.

### 3.6. Модули

Модуль представляет собой обособленную и самостоятельную часть приложения, реализующую определенную часть логики приложения. Каждый модуль имеет свою собственную архитектуру пользовательского интерфейса, но имеет механизм встраивания в окно приложения, предоставляя в качестве встраиваемого объекта визуальный компонент (кнопка, панель и т.д.).

Список модулей, входящих в состав платформы:

Имя модуля	Назначение
<a href="#"><u>Service Configurator</u></a> Модуль конфигурации локальных сервисов	Предоставляет возможность настраивать работу сервисов или отключать их, если сервис разрешает такую возможность.
<a href="#"><u>Explorer</u></a> Модуль проводника	Реализует навигацию по древовидной структуре сущностей приложения, интерфейс создания и настройки сущностей, вызов команд сущностей и их свойств.
<a href="#"><u>Logger</u></a> Модуль трассировки	Предназначен для отображения событий приложения.
<a href="#"><u>Task Manager</u></a> Модуль исполнения задач	Позволяет исполнять, отслеживать и контролировать задачи.
<a href="#"><u>Launcher</u></a> Модуль ярлыков команд	Позволяет создавать ярлыки на команды для быстрого доступа.
<a href="#"><u>Instance Viewer</u></a> Модуль экземпляров приложения	Позволяет обмениваться данными с экземплярами приложения, запущенных на других машинах.




### 3.6.1 Модуль конфигурации сервисов (Service Configurator)

Представляет собой селектор локальных сервисов, запущенных приложением.



Изображение 3.6.1 – список системных сервисов (сервис нотификации отключен)

Каждый сервис имеет собственный набор редактируемых свойств (настроек) и/или динамических свойств для отображения полезной информации. В зависимости от набора свойств конкретного сервиса кнопка вызова редактора принимает разный вид:

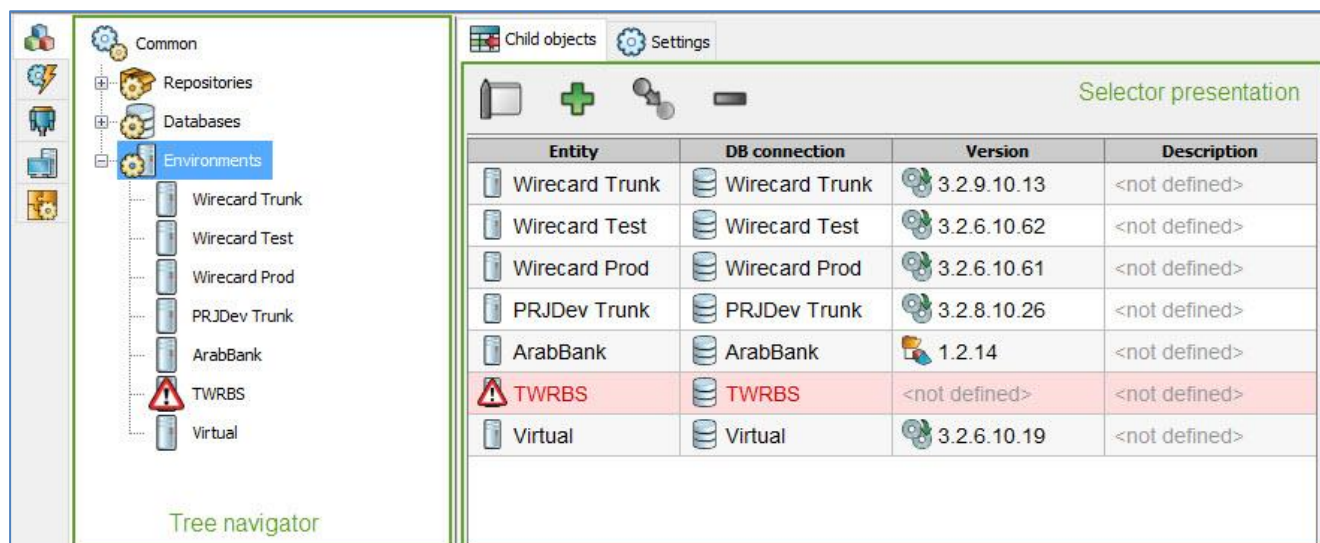
-  Сервис имеет редактируемые свойства.
-  Сервис имеет только информационные свойства.
-  Сервис не имеет свойств и редактор не доступен.

Также имеются две команды для запуска и остановки сервисов, которые поддерживают данную возможность. Большинство сервисов играют ключевую роль в работе приложения и их остановка недопустима.

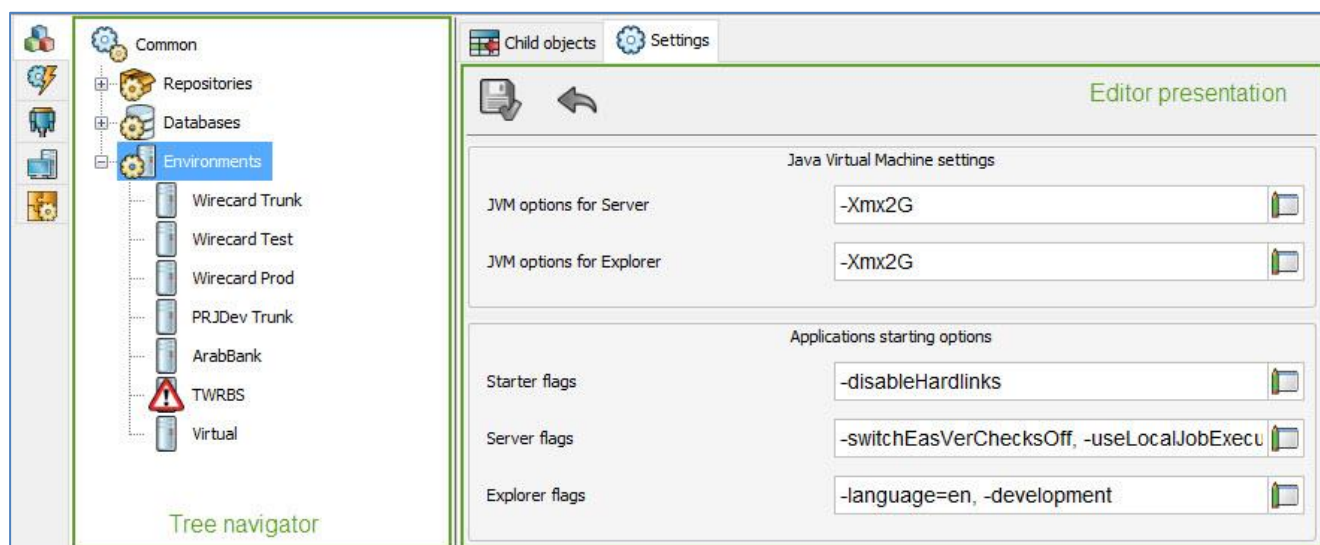
## 3.6.2 Модуль проводника (Explorer)

Визуально модуль представлен в виде панели, разделенной на 2 части:

- Навигатор древовидной структуры сущностей в левой части
- Панель отображения вкладок Редактора и/или селектора сущностей в правой



Изображение 3.6.2 – вкладка селектора



Изображение 3.6.3 – вкладка редактора

Роль проводника – обеспечить возможность найти нужную сущность и выделив его, получить доступ к его настройкам (редактор) или списку дочерних сущностей (селектор), а также запускать команды сущностей.

В правой панели проводника в общем случае располагаются вкладки редактора свойств выбранной сущности и вкладка селектора дочерних объектов. При этом, если выбранная сущность не содержит свойств, доступных для редактирования – вкладка редактора не отображается. Если выбранная сущность не является каталогом – не отображается вкладка селектора.

### 3.6.2.1 Навигатор

Навигатор представляет собой дерево сущностей, при выборе которых обновляется информация в правой части.

Сущность отображаются в виде иконки определенной в классе и её наименования в одном из трех режимов:

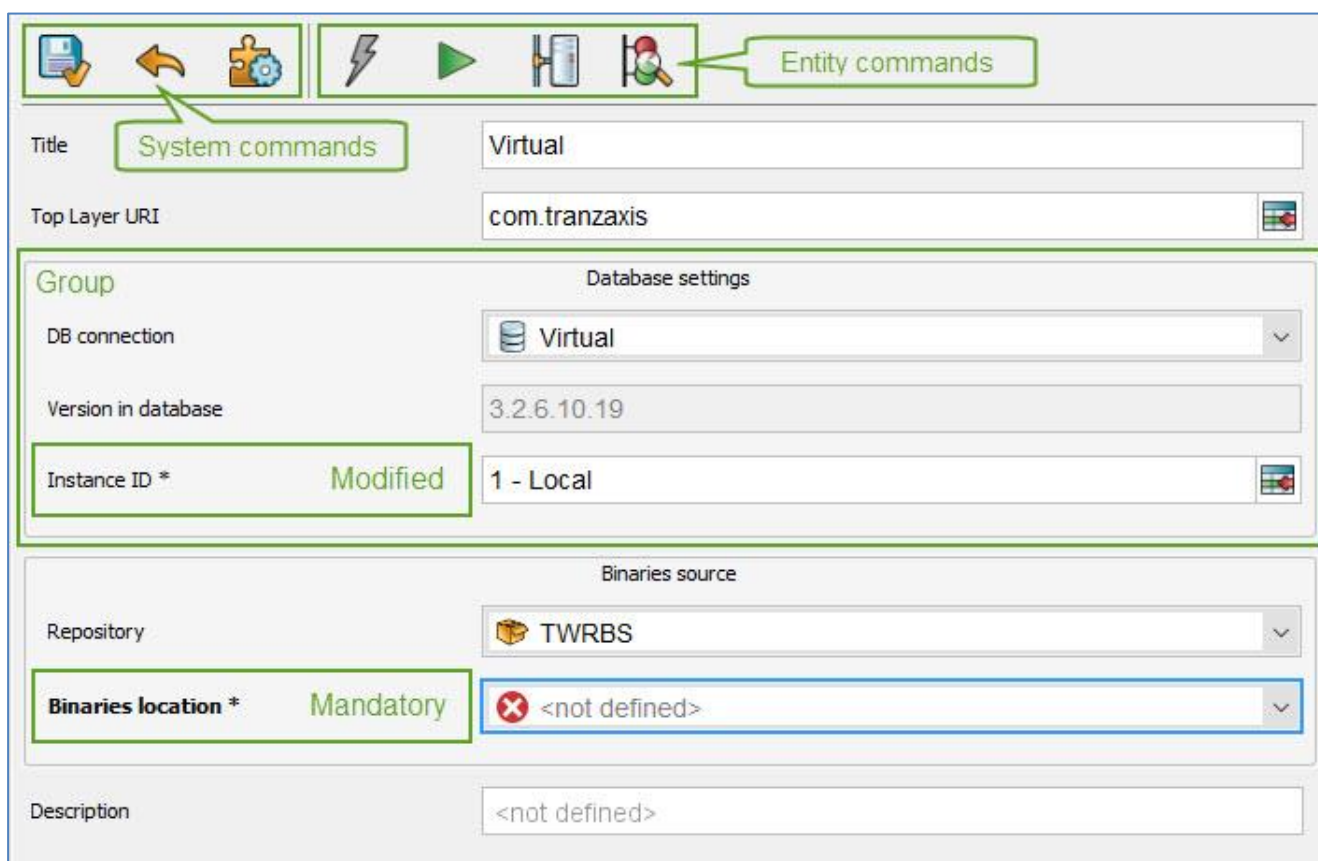
- Доступна для выбора и активирована – обычный режим отображения
- Доступна для выбора, но не активирована – отображается монохромная иконка сущности. При этом прикладной смысл данного режима определяется разработчиком.
- Не доступна для выбора и не активирована – сущность не готова к взаимодействию с пользователем в данный момент времени. В частности, каталоги находятся в этом состоянии в процессе загрузки их дочерних сущностей, как следствие – построение селектора будет преждевременным.

Если сущность некорректна – отображается комбинированная иконка сущности и знака предупреждения. Если сущность заблокирована – отображается комбинированная иконка сущности и знака блокировки.

### 3.6.2.2 Редактор

Редактор сущности представляет собой компонент состоящий из панели команд в верхней части и страницы редактора.




Общая схема редактора сущности:



Изображение 3.6.4

#### Панель команд

Панель команд содержит системные команды редактора и прикладные команды сущности. Системные команды:

	Команда	Горячая клавиша	Описание
	Сохранение изменений сущности	Ctrl+S	Команда активируется при наличии измененных хранимых свойств в модели сущности. При вызове производится сохранение всех изменившихся хранимых свойств, а также перерасчет всех динамических свойств.
	Откат изменений сущности	Ctrl+Z	Команда активируется при наличии измененных хранимых свойств в модели сущности. При вызове производится восстановление сохраненных ранее значений измененных свойств, а также перерасчет всех динамических свойств.
	Отображение дополнительных настроек сущности	-	<p>Команда добавляется в панель редактора если модель обладает свойствами, помеченными как дополнительные – как правило это свойства для более тонкой настройки поведения объекта, которую приходится выполнять не часто.</p> <p>При вызове появляется модальное окно с дополнительными настройками.</p> <div style="border: 1px solid red; padding: 5px; margin-top: 10px;"> Сохранение и откат значений производится независимо от основных настроек модели. </div>

### Страница редактора

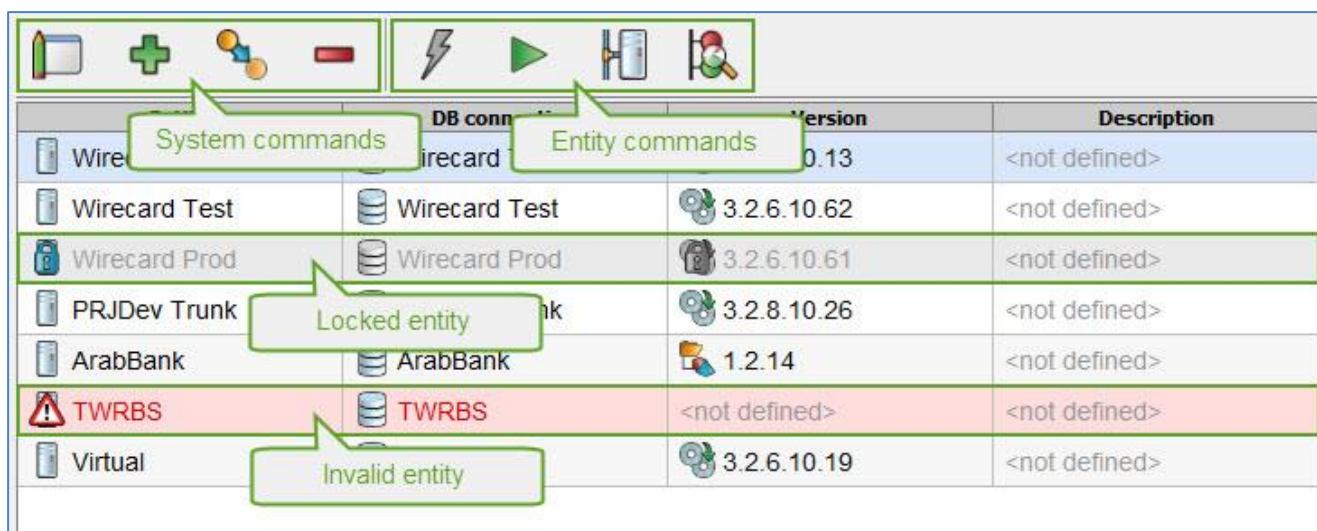
Страница редактора отображает редакторы всех имеющихся в модели свойств, если при их создании не указано запрет отображения в редакторе. Страница разбита на 2 колонки, в левой размещаются метки наименований свойств, в правой – редакторы свойств. Свойства, имеющие общий смысл могут быть объединены в группы, которые визуальнo выделяются в виде именованной панели. Метки наименования, помимо основной задачи отображения наименования свойства, отображает состояние свойства:

- К тексту метки добавляется символ «\*» - свойство изменено, но еще не сохранено.
- Текст метки отображается жирным текстом – свойство обязательно, но не имеет значения.
- При наведении курсора мыши на наименование свойства отображается расширенная подсказка для свойства.

### 3.6.2.3 Селектор

Общая схема селектора сущностей:



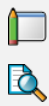





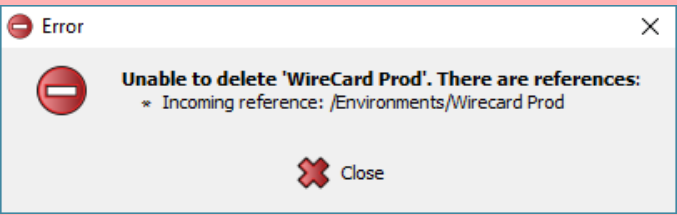
Изображение 3.6.5

Селектор сущностей представляет собой компонент состоящий из панели команд в верхней части и таблицы дочерних объектов каталога.

### Панель команд

Панель команд содержит системные команды селектора и прикладные команды сущностей. Системные команды:

	Команда	Горячая клавиша	Описание
	Редактирование свойств сущности	-	<p>Команда активна, если выбранная сущность имеет свойства, доступные в редакторе. При вызове появляется модальное окно с основными настройками сущности.</p> <div style="border: 1px solid red; padding: 5px;"> <p>Иконка команды может изменяться в случае если:</p> <ul style="list-style-type: none"> <li>• Модель не имеет редактируемых свойств</li> <li>• Сущность заблокирована</li> </ul> <p>Во всех этих случаях редактор открывается в режиме просмотра.</p> </div> <p>Кроме этого, данная команда вызывается при двойном щелчке мыши на строке селектора.</p>
	Создание новой сущности	Ctrl+Numpad+	Команда доступна, если в каталоге разрешено создание сущностей. При вызове появляется модальное окно создания новой сущности для заполнения её основных свойств.
	Копирование сущности	Ctrl+D	Команда доступна, если в каталоге разрешено создание сущностей. При вызове появляется модальное окно создания новой сущности с уже заполненными свойствами, скопированными из настроек выбранной сущности. Пользователь может изменить скопированные значения по своему усмотрению.

	Удаление сущности	Delete	<p>Команда доступна, если в каталоге разрешено создание сущностей. При вызове появляется окно запроса подтверждения на удаление выбранных сущностей. При подтверждении происходит удаление сущности из каталога и БД.</p> <p>Если сущность связана с другой в качестве значения свойства типа EntityRef – команда удаления выведет сообщение о невозможности удаления</p> 
---	-------------------	--------	--

### Таблица дочерних объектов

Таблица содержит записи для каждой дочерней сущности каталога. В колонках таблицы отображается подмножество свойств сущности, разрешенных для отображения в селекторе. Таблица допускает множественное выделение записей, к примеру, для исполнения команд, которые разрешены для группы объектов. Некорректные и заблокированные сущности подсвечиваются цветом и иконкой в первой ячейке (см. схему селектора). Таблица поддерживает механизм ручной сортировки сущностей (drag-and-drop).

Если в каталоге запрещено ручное создание сущностей, то и сортировка становится недоступной.

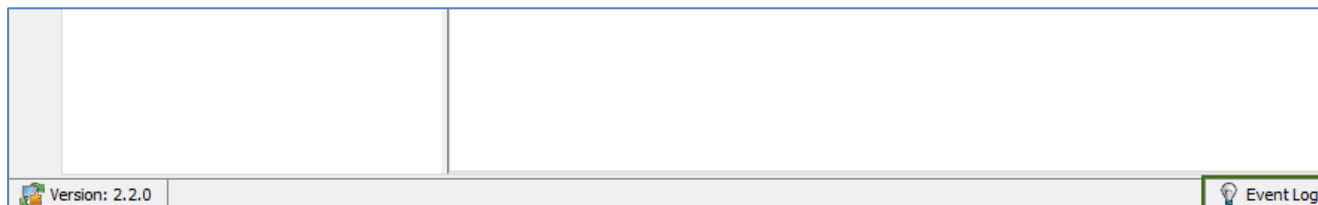


### 3.6.3 Модуль трассировки (Logger)

Модуль предназначен для просмотра событий, генерируемых приложением.

Модуль был разработан в самом начале работ над платформой и приложением. Текущий функционал модуля обеспечивает в большей степени нужды разработчика нежели конечного пользователя и в будущем будет кардинальным образом переработан.

Визуально модуль представлен в виде кнопки в левой нижней части окна приложения, содержащей иконку максимального уровня имеющихся в логе на данный момент сообщений. Нажатие на кнопку открывает панель GUI.



Изображение 3.6.6 – расположение кнопки открытия окна трассировки

Модуль имеет два канала вывода сообщений:

- Стандартные потоки вывода (STDOUT, STDERR) – сообщения текущей сессии
- Панель GUI – сообщения текущей сессии

#### 3.6.3.1 Канал вывода «Стандартные потоки вывода»

Модуль записывает сообщения о событиях приложения в поток вывода STDOUT события с уровнем Debug, Event и Warning. Сообщения об ошибках помещаются в поток STDERR. При помощи сервиса управления трассировкой ([Logger Management Service](#)) можно программно устанавливать необходимость вывода сообщений того или иного уровня.

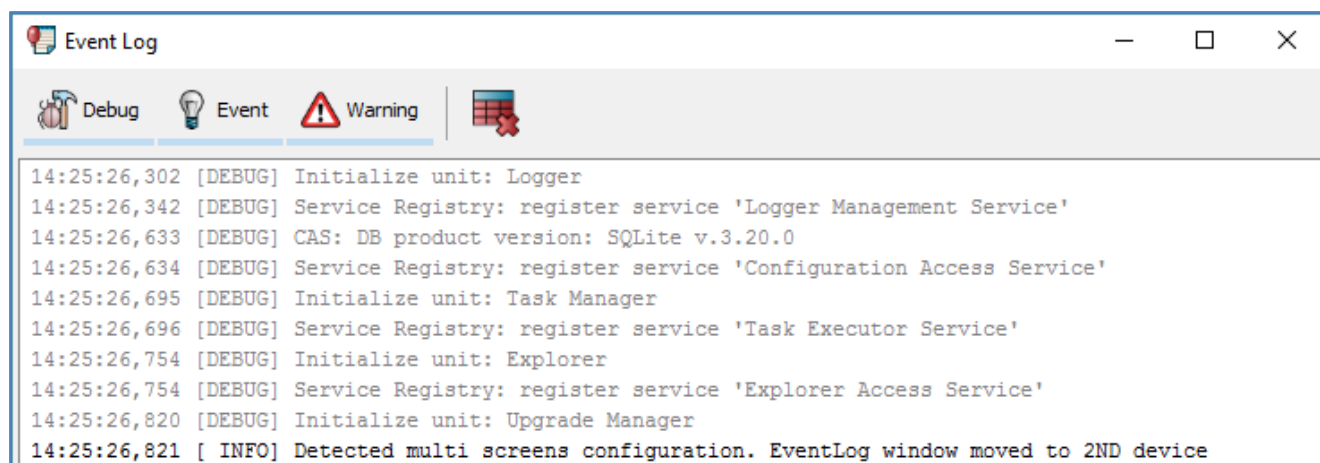
```
16:57:07,519 [DEBUG] Perform command [update]. Context: [trunk]
16:57:07,654 [DEBUG] Task 'Update working copy: "E:\TX\sources\svn2.compassplus.ru.twrbs.trunk\trunk"' state changed: Pending -> Started
16:57:09,673 [DEBUG] Task 'Update working copy: "E:\TX\sources\svn2.compassplus.ru.twrbs.trunk\trunk"' state changed: Started -> Cancelled
16:57:09,690 [WARN] UPDATE [E:\TX\sources\svn2.compassplus.ru.twrbs.trunk\trunk] canceled
16:57:39,724 [DEBUG] Selected path: /Repositories/ArabBank/Sources
16:57:44,362 [DEBUG] Perform command [build]. Context: [trunk]
16:58:29,128 [INFO] BUILD SOURCE [E:\TX\sources\svn.compassplus.ru.pacb.trunk.tranzaxis.arabank.custdev\trunk] finished. Total time: 43 seconds
16:58:29,531 [DEBUG] Property 'trunk@built' has been changed: '1095 / There are errors' -> '1095 / There are errors'
16:58:29,613 [DEBUG] CAS: Altered catalog OFFSHOOT entry: #126 {built=2[4]1095[1]1}
16:58:29,618 [DEBUG] Task 'Build working copy: "E:\TX\sources\svn.compassplus.ru.pacb.trunk.tranzaxis.arabank.custdev\trunk"' state changed: Started -> Failed
16:58:29,631 [DEBUG] Task 'Build product modules' state changed: Started -> Failed
16:58:29,633 [ERROR] BUILD SOURCE [E:\TX\sources\svn.compassplus.ru.pacb.trunk.tranzaxis.arabank.custdev\trunk] failed. Total time: 43 seconds
Build failed. Total errors: 12, total warnings: 10
[ArabBank.CustDev::WF.Issuance.Instant::Issuance_procType]
- Web: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Web: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
- Desktop: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Desktop: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
[ArabBank.CustDev::WF.Issuance.InternetShoppingCard::Issuance_procType]
- Web: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Web: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
- Desktop: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Desktop: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
[ArabBank.CustDev::WF.Issuance.Manual::Issuance_procType]
- Web: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Web: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
- Desktop: The type col6KIGNQEL7NEWFDQMOEIO4POSTQ is ambiguous
- Desktop: The type col6SKMJSZ2NFVONGCP26PG6QPEM is ambiguous
```

Изображение 3.6.7 – консольный вывод

#### 3.6.3.2 Канал вывода «Панель GUI»

Панель графического интерфейса содержит органы кнопки управления (переключатели видимости для каждого уровня трассировки, кнопку очистки лога) и окно вывода событий.





Переключатели вывода сообщений только скрывают сообщения определенного уровня, не отключая при этом запись сообщений в окно.



Изображение 3.6.8 – вывод логов в панель

### 3.6.3.3 Уровни трассировки

Каждому сообщению присваивается уровень:

-  *Debug* – отладочные сообщения, содержат информацию о внутренних процессах приложения.
-  *Event* – события, предназначенные для информирования пользователя приложения.
-  *Warning* – предупреждение об ошибках, не влияющих на работоспособность приложения
-  *Error* – ошибки или исключения, которые могут нарушить работоспособность приложения или целостность данных.

### 3.6.4 Модуль исполнения задач (Task Manager)

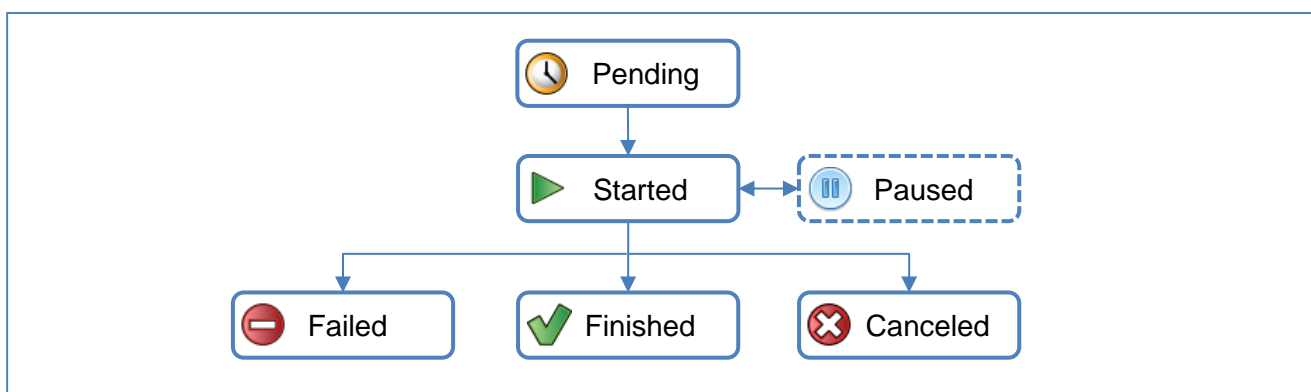
Если в приложении возникает необходимость в выполнении длительных операций (ввод-вывод, порождение процессов и т.д.) возможна упаковка таких операций в задачу, с последующей передачей её на исполнение модулю исполнения задач при помощи встроенного сервиса исполнения задач (Task Execution Service). Задачи могут исполняться в одном из двух пулов потоков (очередях):

- Очередь фонового исполнения. Размер – 10 потоков (фиксированная), при превышении указанного размера, задачи очереди, не распределенные по потокам, находятся в состоянии ожидания, до момента освобождения любого из потоков.
- Очередь оперативного исполнения. Размер – неограничен, потоки выделяются по необходимости и по завершении задачи уничтожаются.







**Задача** – операция приложения, длительность которой не позволяет запускать её синхронно либо прикладной смысл операции требует запуск её в асинхронном режиме. Задачи делятся на 2 типа:

- Обычная задача – независима от других задач.
- Групповая задача – содержит последовательность вложенных задач, которые должны быть выполнены последовательно. При этом успешным завершением групповой задачи будет считаться успешное завершение всех её вложенных задач. Контроль над исполнением (отмена, приостановка) вложенных задач невозможен.

#### 3.6.4.1 Жизненный цикл задачи



Изображение 3.6.9 – схема жизненного цикла задачи

-  **Pending** – задача ожидает очереди на исполнение (входит в состав групповой задачи или пул потоков заполнен).
-  **Started** – задача в стадии исполнения.
-  **Paused** – задача приостановлена по команде пользователя. Не каждая задача может поддерживать данную функцию – об этом должен позаботиться разработчик приложения.
-  **Finished** – задача выполнила свою функцию и завершилась без ошибок.
-  **Failed** – в процессе исполнения задачи возникла ошибка, в следствие чего дальнейшее исполнение задачи стало невозможным.
-  **Canceled** – задача отменена по команде пользователя.

### 3.6.4.2 Планирование задач

Сервис исполнения задач принимает задачи, позволяя выбрать один из трех вариантов исполнения модулем:

- Исполнить задачу в очереди фонового исполнения – стандартный способ исполнения, задача отображается в мониторе и доступна для контроля пользователем (см. ниже).
- Исполнить задачу в очереди оперативного исполнения – при этом задача будет отображаться в модальном диалоге. Пользователь может как контролировать исполнение задачи, так и переместить её в фоновый режим.
- Исполнить задачу в очереди оперативного исполнения, но незаметно для пользователя – в этом случае задача выполняется без возможности контроля со стороны пользователя. Таким способом, к примеру, выполняются задачи загрузки каталогов.

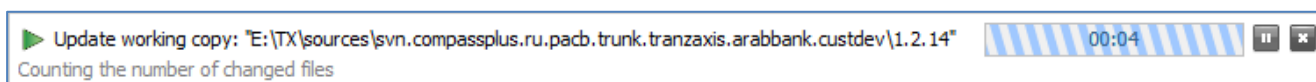
### 3.6.4.3 Виджеты задач

Задачи отображаются в диалоге или мониторе в виде виджетов, которые различаются в зависимости от типа задачи.

Виджет одиночной задачи содержит:

- иконку статуса жизненного цикла (слева)
- наименование задачи (в верхней части)
- пояснение прогресса исполнения (в нижней части)
- прогресс (процент выполнения или время с момента старта)
- кнопки управления (остановка и, если возможно, приостановка)

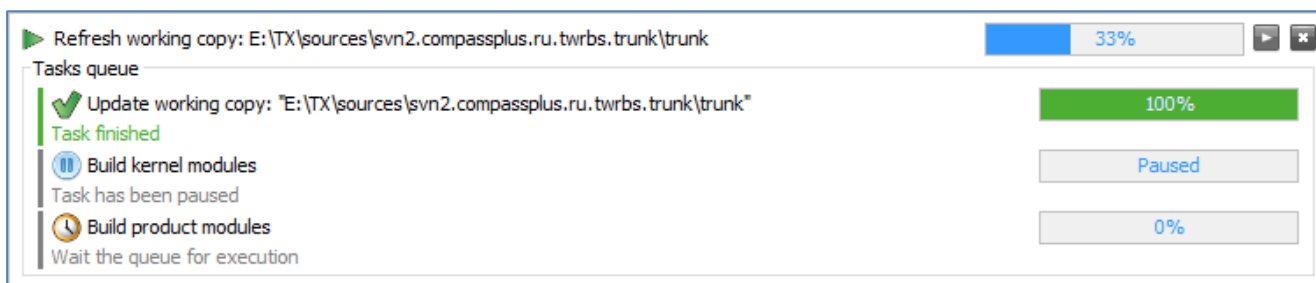
Кнопка остановки задачи при первом нажатии приводит к остановке задачи, повторное нажатие удаляет задачу из очереди.



Изображение 3.6.10 – виджет одиночной задачи

Виджет групповой задачи содержит:

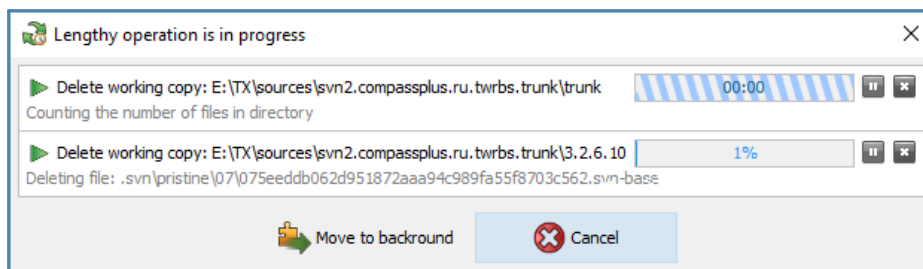
- Все перечисленные элементы одиночной задачи применительно к головной задаче.
- Список виджетов вложенных задач, без кнопок управления.



Изображение 3.6.11 – виджет групповой задачи

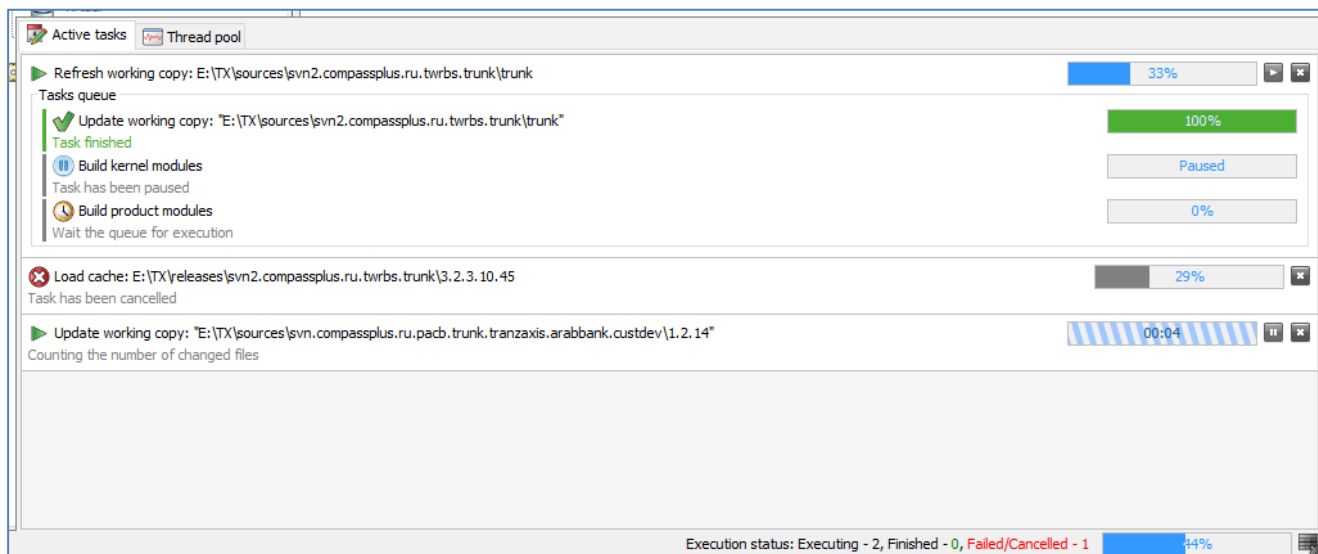
### 3.6.4.4 Контроль задач

Задачи, исполняемые в оперативной очереди, отображаются в модальном окне. При необходимости их можно переместить в фоновое исполнение.



Изображение 3.6.12 – модальное окно

Задачи, исполняемые в фоновой очереди, помещаются в монитор, представляющий собой панель статуса и открываемое по щелчку на панели окно, отображающее текущее состояние очереди.



Изображение 3.6.13 – контроль задач в мониторе

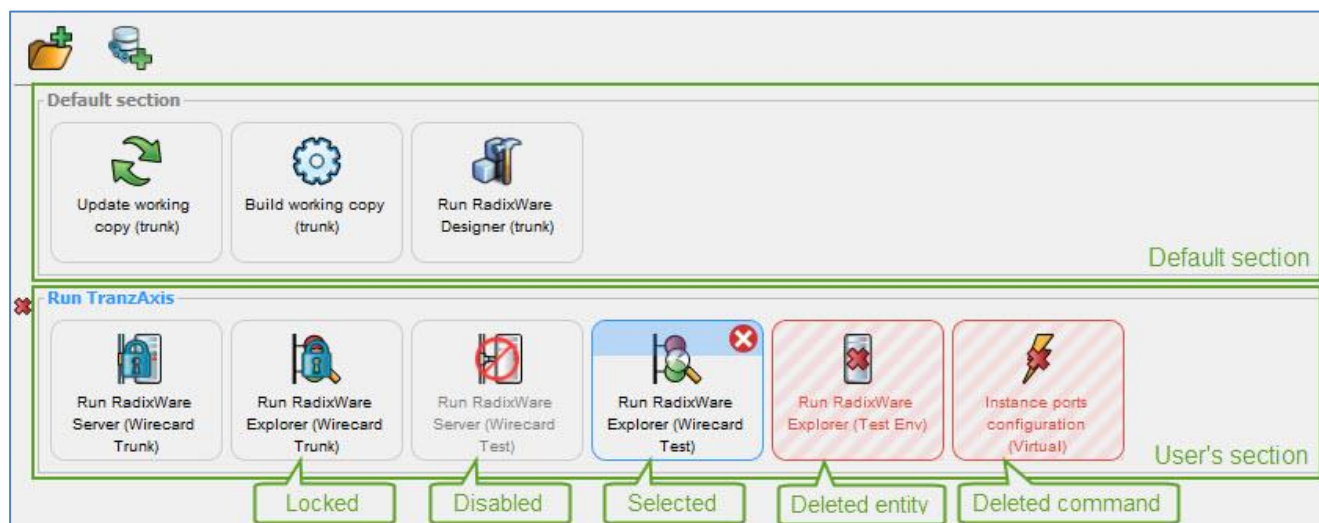
Статус исполнения отображает краткую информацию о задачах, находящихся в данный момент в очереди:

- Количество исполняющихся, завершенных и прерванных (пользователем или в результате ошибки) задач.
- Общий прогресс очереди задач.

Очередь задач автоматически очищается, если все задачи очереди выполнены успешно, в противном случае задачи остаются для детального просмотра результатов пользователем в мониторе. В этом случае пользователь может вручную удалить сбойные или отмененные задачи.

### 3.6.5 Модуль ярлыков команд (Launcher)

Модуль используется для более быстрого доступа к часто используемым командам сущностей. Модуль представлен в виде панели, которая содержит в себе секции с ярлыками.



Изображение 3.6.14 – различные статусы ярлыков

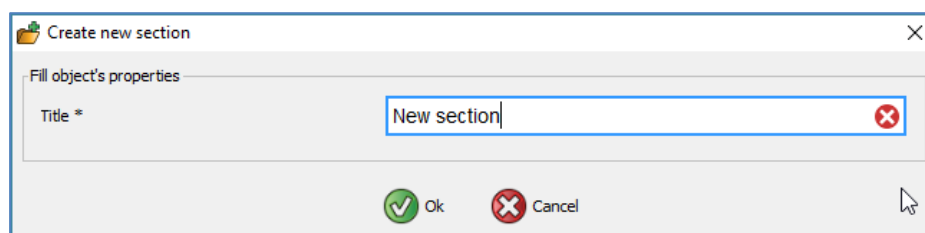
Ярлыки, не привязанные к какой-либо секции, располагаются в специальной «секции по умолчанию», которая отображается только если в ней находится хотя бы один ярлык. Данная секция создается автоматически при первом запуске модуля, она не может быть удалена или переименована.

При наведении курсора мыши на ярлык, он выделяется, при этом в заголовке отображается кнопка удаления. Ярлыки визуальнo отображают следующие состояния:

- **Активный** – команда доступна.
- **Отключен** – команда недоступна в данный момент согласно её условиям применимости для данной сущности.
- **Заблокирован** – команда не может быть выполнена, поскольку сущность в данный момент заблокирована.
- **Не найдена сущность** – Возможно если после создания ярлыка сущность, на которую он ссылается, была удалена.
- **Не найдена команда** – Возможно если после создания ярлыка команда, на которую он ссылается, была удалена из сущности или переименована.

#### 3.6.5.1 Создание секций

Создание секции может быть вызвано командой на панели модуля или из диалога создания ярлыка.

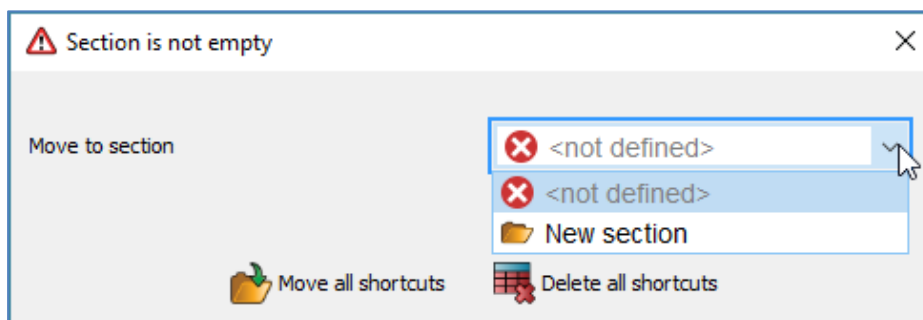


Изображение 3.6.15 – создание секции

### 3.6.5.2 Удаление секции

Пользовательская секция может быть удалена при нажатии кнопки удаления в левом верхнем углу секции. При этом проверяется наличие в ней ярлыков и если секция не пуста предлагается 2 варианта:

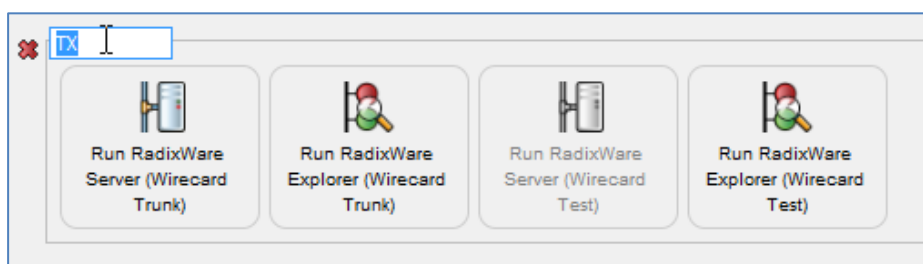
- Переместить все ярлыки в новую секцию (если не задано – ярлыки переместятся в «секцию по умолчанию»).
- Удалить все ярлыки вместе с секцией.



Изображение 3.6.16 – размещение ярлыков при удалении секции

### 3.6.5.3 Переименование секции

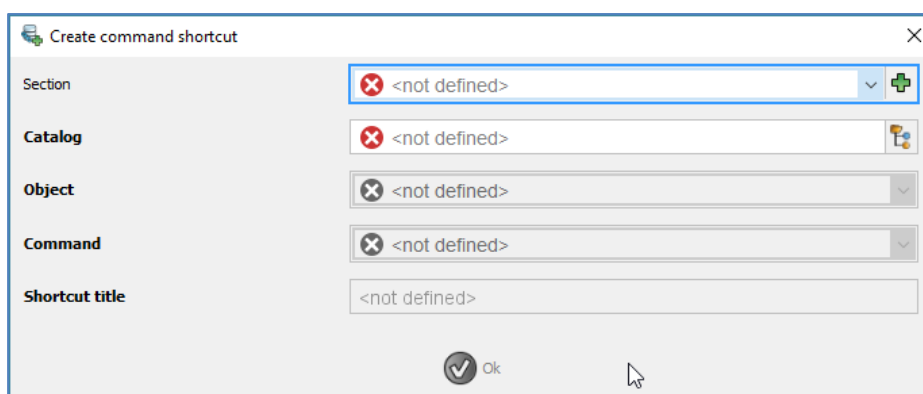
Переименование уже имеющейся пользовательской секции производится в текстовом редакторе, который вызывается двойным щелчком мыши на имени секции. После ввода нового имени следует нажать ENTER для применения изменения, в противном случае изменение будет отменено.



Изображение 3.6.17 – переименование секции

### 3.6.5.4 Создание ярлыка

Диалог создания нового ярлыка выглядит следующим образом:



Изображение 3.6.18 – создание ярлыка

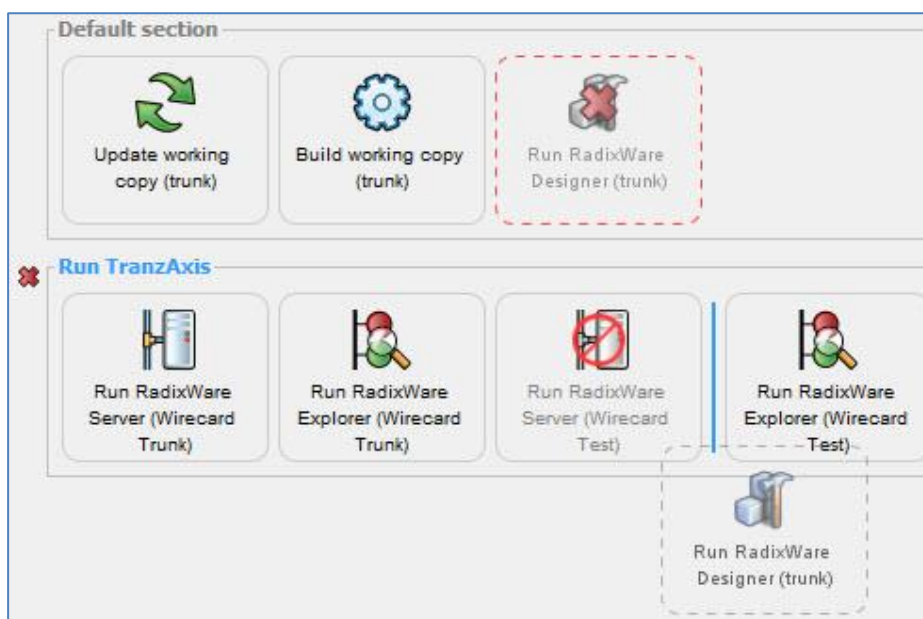
Для создания ярлыка требуется последовательно заполнить следующие свойства:



- **Секция** – Ссылка на секцию. Для выбора доступны имеющиеся пользовательские секции. Если значение не задано, ярлык будет размещен в «секции по умолчанию». При необходимости можно создать секцию командой в редакторе, в этом случае она автоматически будет выбрана в качестве значения.
- **Каталог** – Ссылка на каталог. Для выбора доступны каталоги, имеющие дочерние объекты, хранящиеся в БД, которые в свою очередь имеют команды.
- **Объект** – Ссылка на сущность в выбранном каталоге.
- **Команда** – Ссылка на одну из команд выбранной сущности
- **Заголовок ярлыка** – Имя, отображаемое в ярлыке, формируется автоматически на основе выбранной сущности и имени команды, но может быть скорректировано пользователем.

### 3.6.5.5 Ручная перегруппировка ярлыков

Ярлыки могут быть перераспределены пользователем вручную. Для этого реализован механизм Drag and Drop. Режим включается при нажатии левой кнопки мыши на ярлыке и смещении на небольшое расстояние. Выбранный ярлык помечается специальным образом (см. изображение) и появляется курсор, который перемещается вслед за мышью, отображая будущую позицию ярлыка после отпускания кнопки мыши. Таким образом переместить ярлык можно в другую позицию как в текущей его секции, так и вставить в другую.

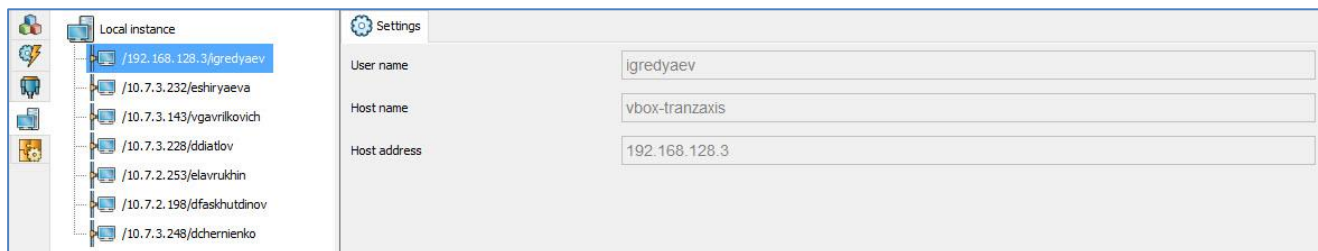


Изображение 3.6.19 – процесс перетаскивания ярлыка между секциями



### 3.6.6 Модуль инстанций приложения (Instance Viewer).

Представляет собой дерево сущностей внешних инстанций приложения (запущенных на других машинах локальной сети) и подключенных в данный момент к текущей инстанции приложения.



Изображение 3.6.20 – список подключенных инстанций

Локальная инстанция не имеет свойств.

Внешняя инстанция имеет следующие свойства:

- **Имя пользователя** – имя пользователя операционной системы, запустивший инстанцию приложения на удаленной машине.
- **Имя хоста** – сетевое имя удаленной машины.
- **Адрес хоста** – IP адрес удаленной машины.

### 3.7. Сервисы

**Сервисы** представляют собой механизм программного доступа к функциям приложения или его модулей. При старте приложения создается реестр сервисов, который в свою очередь последовательно загружает все системные сервисы приложения, запускает их и регистрирует в модуле конфигурации сервисов.

Основные сервисы, входящие в состав платформы:

Имя сервиса	Назначение
<a href="#"><u>Configuration Access Service</u></a> Сервис доступа к настройкам	Реализует функции хранения и доступа к данным всех сущностей, такие как поиск сущностей определенного класса, поиск сущностей по уникальному идентификатору или составному (наименование и ID владельца), обслуживание структуры хранилища (синхронизация структуры таблиц с моделями сущностей) и т.д.
<a href="#"><u>Logger Management Service</u></a> Сервис управления трассировкой	Изменение настроек отображения событий трассировки. Реализуется модулем <a href="#"><u>Logger</u></a> .
<a href="#"><u>Explorer Access Service</u></a> Сервис доступа к дереву сущностей (возможно, будет упразднен)	Поиск объектов в дереве по уникальному идентификатору или принадлежащих определенному классу. Реализуется модулем <a href="#"><u>Explorer</u></a> , а также необходим для механизма работы редакторов свойств типа EntityRef.
<a href="#"><u>Task Execution Service</u></a> Сервис исполнения задач	Прием задач на исполнение в пуле потоков. Реализуется модулем <a href="#"><u>Task Manager</u></a> .
<a href="#"><u>Notification Service</u></a> Сервис генерации уведомлений	Уведомление пользователя о событиях путем формирования системных уведомлений. В частности, используется модулем <a href="#"><u>Task Manager</u></a> для уведомления о результате исполнения запущенной задачи и модулем <a href="#"><u>Logger</u></a> при возникновении ошибок.
<a href="#"><u>Instance Communication Service</u></a> Сервис взаимодействия инстанций	Предназначен для поиска и установления соединений между инстанциями приложения в локальной сети. После установления соединения инстанции могут взаимодействовать посредством опубликованных внешних сервисов.

### 3.7.1 Сервис доступа к настройкам

Все настройки приложения, включающие в себя значения хранимых свойств сущностей, хранятся в базе данных SQLite, создаваемой автоматически в директории профиля пользователя операционной системы – файл «manager.db».

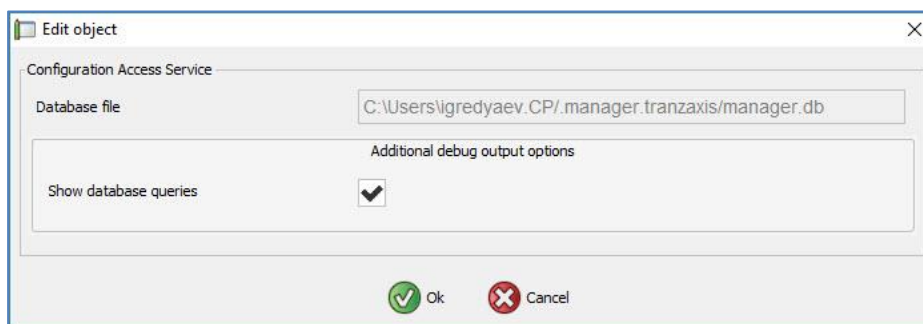
Каждый тип сущности представлен в виде отдельной таблицы, а свойства сущности – её колонки. Колонки таблиц содержат строковые представления значений сущностей.

При создании первой сущности определенного класса сервис автоматически создает необходимую структуру таблицы, включая связанные по внешнему ключу. Если в процессе изменения класса сущности (новая версия ПО) набор её свойств меняется, сервис также автоматически корректирует структуру таблиц, удаляя или добавляя колонки.

Таблицы связываются при помощи внешних ключей, обеспечивая таким образом работу типа данных EntityRef с одной стороны и контроль используемых объектов с другой. При этом связи таблиц подразделяются на два типа:

- Внешние – если иная сущность имеет поле типа EntityRef значение которой указывает на данную сущность.
- Внутренние – если есть иная сущность, которая имеет ссылку на данную сущность в качестве владельца.

#### 3.7.1.1 Диалог настройки сервиса



Изображение 3.7.1

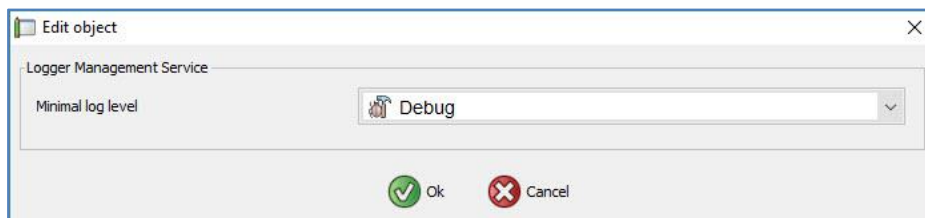
Сервис имеет следующие свойства:

- **Файл базы данных** – содержит путь к файлу БД.
- **Отображать запросы базы данных** – Все операции изменения данных в БД будут выводиться в лог с уровнем “Debug”. Кроме этого, при старте приложения в лог будет выведен дамп структуры таблиц с указанием состава колонок, их типов и описания ссылок, если колонка таблицы представляет собой внешний ключ.

## 3.7.2 Сервис управления трассировкой

Сервис используется для глобальной настройки уровня сообщений, выводимых в имеющиеся каналы вывода модуля [Logger](#), который устанавливается при старте приложения и в момент смены данной настройки. При необходимости, пользователь может скорректировать необходимость вывода событий того или иного уровня в панель GUI.

### 3.7.2.1 Диалог настройки сервиса



Изображение 3.7.2




Сервис имеет следующие свойства:

- **Минимальный уровень событий лога** – все сообщения с уровнем ниже указанного не будут выводиться в стандартный поток вывода. Окно сообщений также будет настроено по умолчанию в соответствии с выставленным значением.




### 3.7.3 Сервис нотификации

Сервис предназначен для доставки уведомлений пользователю о событиях, на которые следует обратить внимание. Сервис предоставляет программный интерфейс для регистрации источника уведомления с последующим использованием данного источника при генерации событий. При регистрации источника также задается условие, при котором данное уведомление должно быть показано и оба значения (источник и условие показа) добавляются в настройку сервиса, соответственно, пользователь впоследствии может самостоятельно изменить правило показа сообщений от каждого конкретного источника по своему усмотрению. Кроме этого, данный сервис не является критическим для работы системы и может быть остановлен.

Доступные условия от отображения уведомлений:

-  Уведомление показывается всегда.
-  Уведомление показывается только когда окно приложение свернуто или не является активным в данный момент.
-  Уведомления отключены.

Уведомления подразделяются на три типа по серьезности события

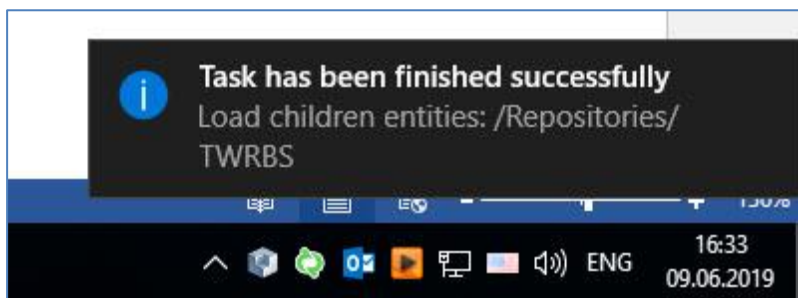
-  Информация для пользователя.
-  Предупреждение о проблеме, тем не менее не являющейся критичной для работы системы и/или её компонентов.
-  Ошибка, которая может привести к нестабильной работе системы.

Модуль имеет один канал вывода сообщений:

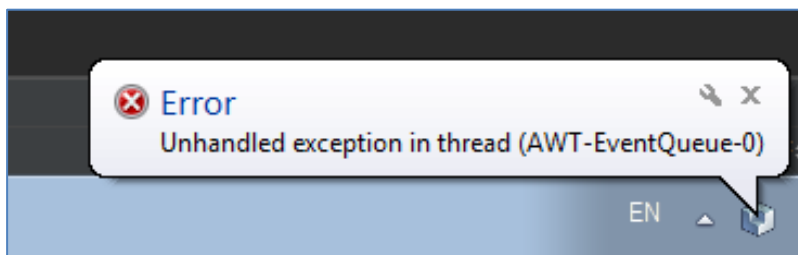
- Сообщения операционной системы

#### 3.7.3.1 Канал «Сообщения операционной системы»

Канал предназначен для оперативного уведомления пользователя о событии, поскольку по истечении некоторого времени (зависит от операционной системы) уведомление пропадает. Хотя внешний вид сообщения зависит от установленной операционной системы, оно всегда содержит заголовок и тело сообщения:

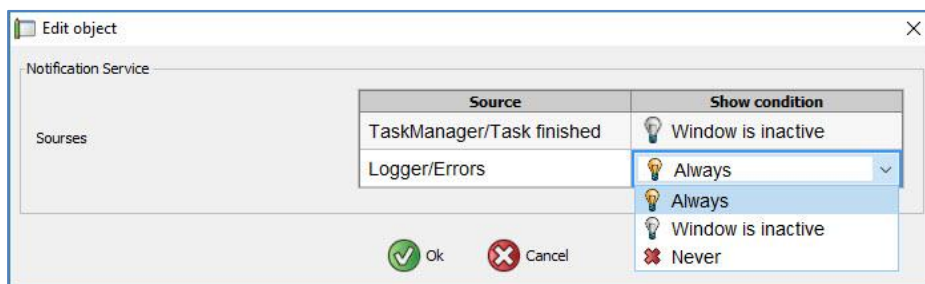


Изображение 3.7.3 – вид уведомления в Windows 10



Изображение 3.7.4 – вид уведомления в Windows 7

### 3.7.3.2 Диалог настройки сервиса



Изображение 3.7.5

Сервис имеет следующие свойства:

- **Источники** – карта источников сообщений и условий показа каждого из них.

### 3.7.4 Сервис исполнения задач

Сервис предоставляет собой программный интерфейс для передачи экземпляров задач на исполнение пулу потоков. При получении задачи, сервис планирует запуск задачи и уведомляет модуль исполнения задач, который в свою очередь создает виджет для данной задачи и размещает его в соответствующем месте GUI (всплывающее окно либо в мониторе).

### 3.7.5 Сервис взаимодействия инстанций

Сервис представляет собой систему компонент, реализующих взаимодействие приложений, запущенных на различных машинах в локальной сети и выполняющих следующие задачи:

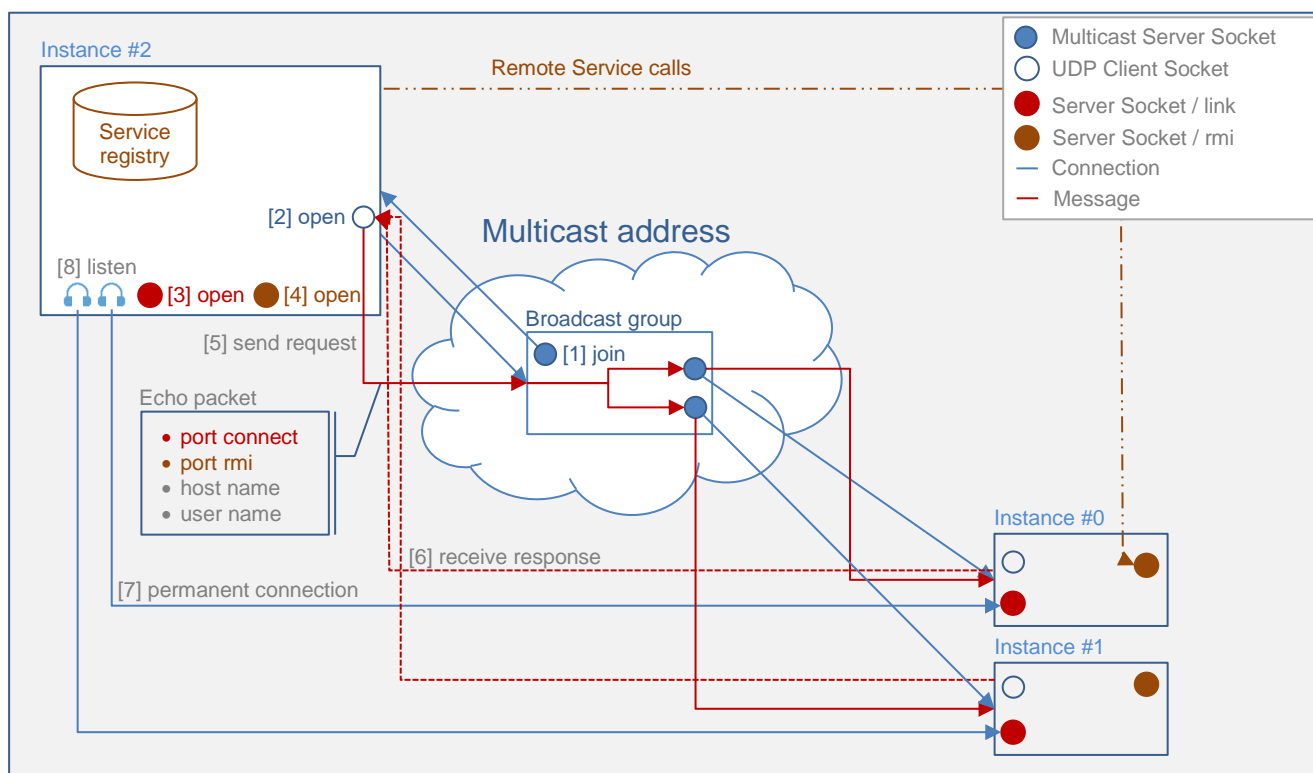
- Установка соединения с внешними инстанциями и контроль отключения.
- Запуск собственных сетевых сервисов.
- Предоставление программного интерфейса к подключенным внешним инстанциям и их сетевым сервисам.

При старте сервиса запускаются два основных компонента сервиса:

- Реестр RMI (Remote Method Invocation) для регистрации собственных сетевых сервисов и предоставления доступа к ним внешним инстанциям.
- Сервер сообщений, обеспечивающий обмен сообщениями между инстанциями, содержащими адреса доступа к реестру RMI.

#### 3.7.5.1 Подключение и отключение инстанций

Сервер сообщений использует технологию многоадресного вещания (Multicast) для доставки сообщений всем инстанциям приложения. Общая схема сетевого взаимодействия инстанций выглядит следующим образом:



Изображение 3.7.6 – схема обнаружения инстанций и взаимоподключение

На этапе [1] сервер открывает серверный порт на адресе из специально выделенного для многоадресного вещания диапазона и присоединяется к группе вещания. Впоследствии, на этот порт будет приходить сообщения от подключенных к группе вещания инстанций. Затем сервер открывает клиентский сокет для посылки UDP сообщений в группу вещания [2], клиентский сокет для постоянного соединения с другими инстанциями [3] и серверный сокет для приема подключений к реестру RMI [4] (порты [3] и [4] выбираются случайно из доступных на машине).

Далее сервер готовит UDP пакет, содержащий следующие данные

- Номер порта для постоянного соединения [3].



- Номер порта для доступа к RMI [4].
- Имя хоста
- Имя пользователя

Пакет отправляется в группу вещания [5] откуда пересылается всем присоединившимся участникам группы (в том числе и отправителю, где данный пакет игнорируется). Инстанция-получатель, получив пакет запускает поток для обработки пакета, который выполняет действия:

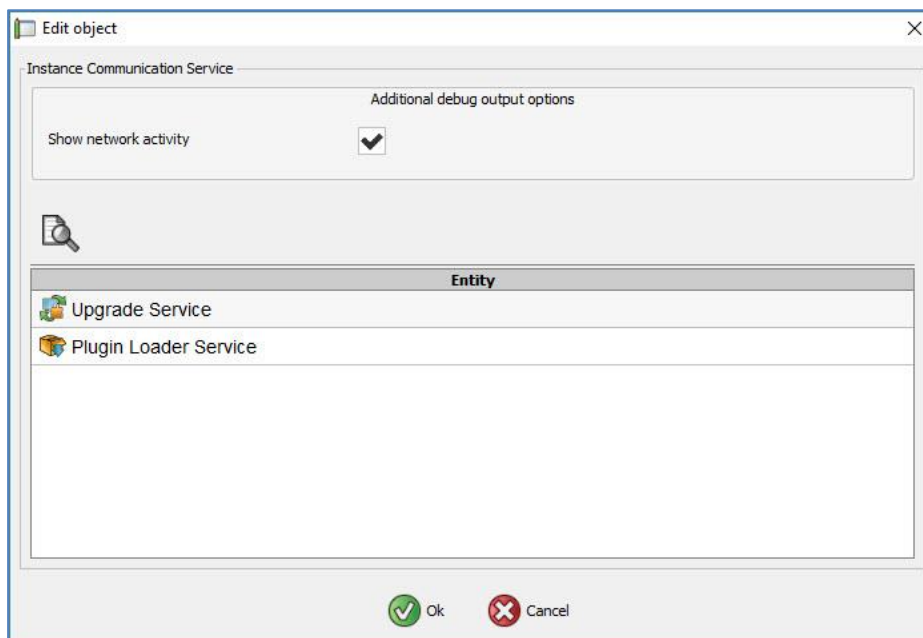
- Соединяется с портом инстанции-отправителя [3], указанным в пакете.
- Добавляет инстанцию в список для отображения в GUI.
- Формирует аналогичный пакет со своими параметрами подключения и отправляет в сокет отправителя (т.е. в обход группы вещания) [6].

Инстанция-отправитель, получив ответный пакет также устанавливает соединение с инстанцией, отправившей ответ [7] и запускает поток, слушающий данное соединение [8], а поскольку в этом соединении данные не передаются, фактически этот поток используется для контроля состояния подключения и при обрыве происходит отключение инстанции.

### 3.7.5.2 Доступ к сервисам

При запуске сервис последовательно создает экземпляры каждого сетевого сервиса и размещает их в реестре RMI (в качестве ключа используется имя интерфейса сервиса). При необходимости вызова сетевого сервиса инстанция сначала обращается к серверу для получения удаленных инстанций, затем получает прокси-объект для сервиса по его имени и вызывает методы, которые фактически исполняются на внешней инстанции.

### 3.7.5.3 Диалог настройки сервиса



Изображение 3.7.7 – список сетевых сервисов

Сервис имеет следующие свойства:

- **Отображать сетевые запросы** – Если флаг включен – выводятся получаемые и отправляемые сетевые пакеты, а также информация о подключении/отключении внешних инстанций.

Диалог настройки сервиса включает вложенный селектор сетевых сервисов, запущенных в данной инстанции с их собственными настройками (в будущем).

## 4. Приложение TranzAxis Manager

Приложение предназначено для оптимизации работы по сопровождению множества разнородных конфигураций продуктов на базе платформы RadixWare, включая:

- различные сервера БД и/или схемы БД с установленным TranzAxis для тестирования разработок как самого продукта, так и проектных разработок отдельно взятого клиента.
- различные сервера SVN с загруженными исходниками и релизами.
- разнородные конфигурации запуска приложения RadixWare Designer для разработки модулей продукта.
- разнородные конфигурации приложения RadixWare Starter для запуска RadixWare Server и/или RadixWare Explorer

В рамках задачи оптимизации работы пользователя (или разработчика) TranzAxis приложение позволяет выполнять следующие основные действия:

- Задачу редактирования конфигурационных файлов заменить на более интуитивную настройку объектов в GUI.
- Автоматизировать большинство операций по сопровождению рабочих копий версий продукта (загрузка из SVN, компиляция, удаление).
- Загружать предварительно скомпилированные релизы продукта, размещенные в SVN.
- Помощь пользователю при настройке объектов и дальнейшем их использовании в рабочих процессах.
- Локализация всех вышеперечисленных действий в одном приложении.

## 4.1. Компоненты приложения

Приложение построено по модульно-сервисной структуре и большинство его компонент предоставляется его платформой ([Платформа приложения](#)), соответственно, с описанием большинства модулей и сервисов можно ознакомиться в соответствующих разделах описания платформы приложения:

- [Модули платформы](#)
- [Сервисы платформы](#)

Кроме этого, приложение реализует собственные компоненты:

- [Модули приложения](#)
- [Подключаемые компоненты \(плагины\)](#)
- [Сущности, команды и задачи](#)
- [Сервисы приложения](#)
- [Сетевые сервисы](#)

## 4.2. Модули приложения

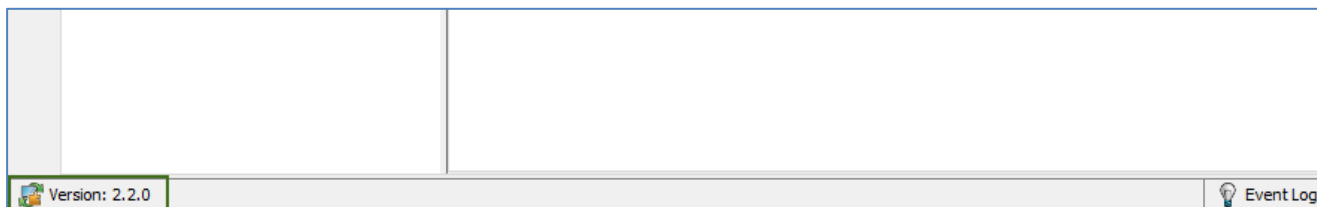
Приложение реализует следующие модули:

Имя модуля	Назначение
<a href="#"><u>Upgrade Management</u></a> Модуль обновления приложения	Обеспечивает возможность обновления приложения. Реализует сетевой сервис обновления подключаемых компонентов ( <a href="#"><u>Upgrade Service</u></a> ).
<a href="#"><u>Plugin Management</u></a> Модуль подключаемых компонентов приложения	Предоставляет подключать / отключать / удалять сторонние компоненты приложения. Реализует сетевой сервис обновления подключаемых компонентов ( <a href="#"><u>Plugin Loader Service</u></a> ).

## 4.2.1 Модуль обновления приложения

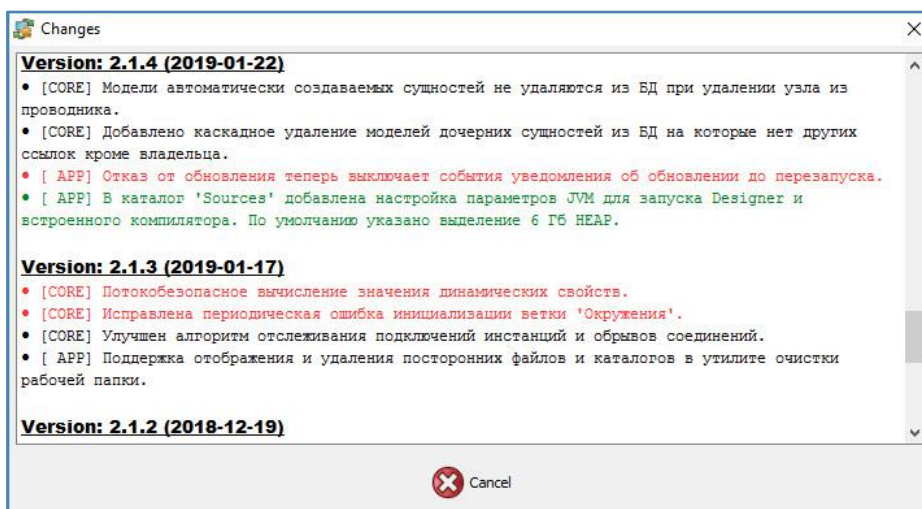
Модуль обновления приложения реализует поиск обновлений приложения среди инстанций приложения, объединенными в сеть сервисом взаимодействия инстанций ([Instance Communication Service](#)) и их установку.

Визуально модуль представлен в виде кнопки в левой нижней части окна приложения, содержащей текущую версию приложения:



Изображение 4.2.1

Нажатие на кнопку открывает диалог отображения истории изменений приложения вплоть до текущей версии



Изображение 4.2.2

Описание версий включает в себя список изменений, классифицируемых по области изменения

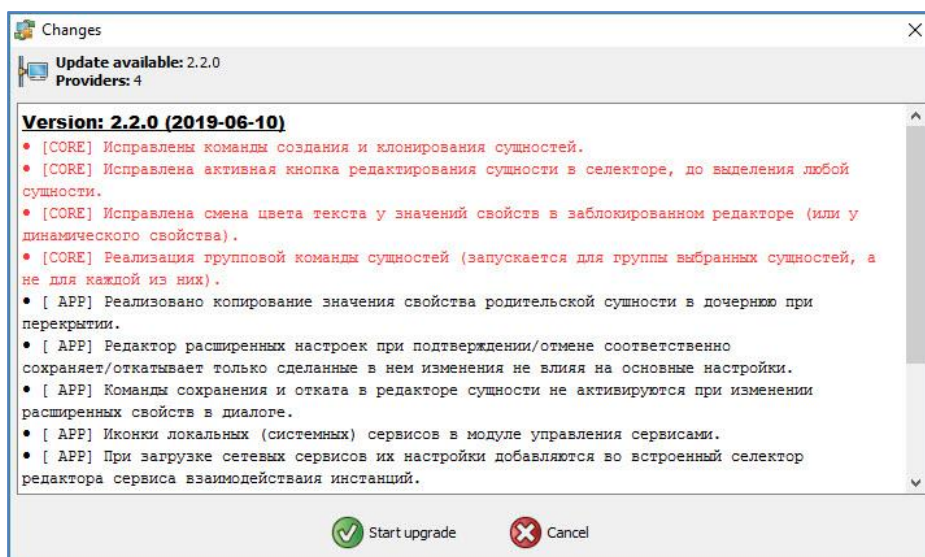
- [CORE] – изменения в платформе приложения
- [APP] – изменения в приложении
- [API] – изменения, затрагивающие публичные интерфейсы классов, доступные разработчику.

и типу изменений

- **Bugfix** – исправление ошибок
- **Change** – изменение, улучшение, оптимизация или рефакторинг если это влияет на внешний вид приложения или его поведение с точки зрения пользователя.
- **Feature** – новые возможности приложения и/или платформы

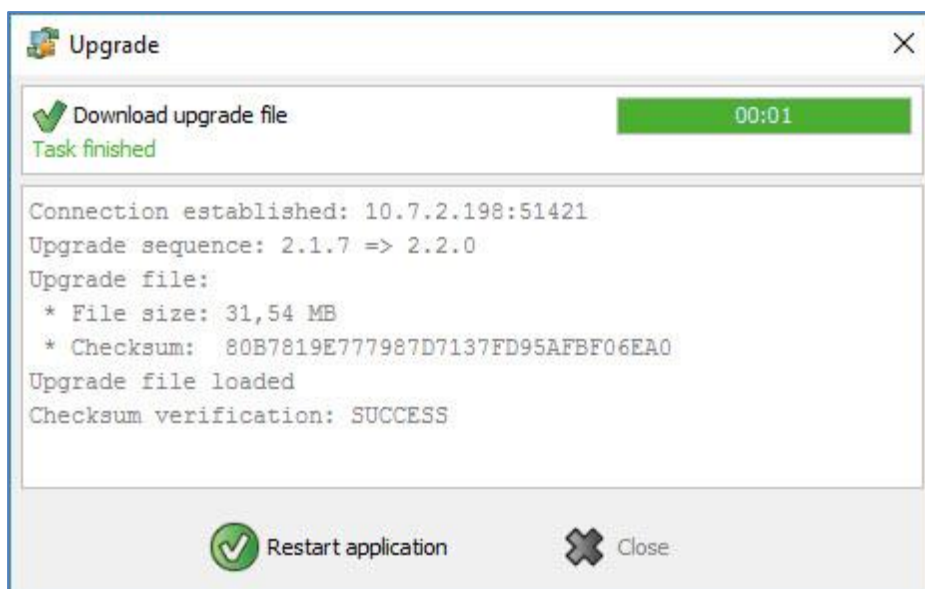
### 4.2.1.1 Процесс обновления приложения

Модуль отслеживает момент подключения внешних инстанций, используя сервис обновления ([Upgrade Service](#)), запрашивает инстанцию о наличии обновлении. В случае возможности обновления модуль формирует список провайдеров (внешний инстанций) с более новой версией, сортирует его по убыванию номера версии и отображает диалог с информацией об изменениях.



Изображение 4.2.3

После нажатия кнопки «Начать обновление» происходит передача файла обновления и проверка контрольной суммы скачанного файла.



Изображение 4.2.4

По окончании процесса загрузки обновления пользователь должен перезагрузить приложение, нажав кнопку «Перезагрузить приложение». В момент перезагрузки происходит перезапись файла приложения и удаление файла обновления.

## 4.2.2 Модуль подключаемых компонентов

Модуль обеспечивает загрузку подключаемых компонентов и встраивание их в общую логику приложения. Также модуль реализует сетевой сервис обновления подключаемых компонентов ([Plugin Loader Service](#)) при помощи которого осуществляется передача обновлений компонентов приложения между инстанциями приложения, объединенными в сеть сервисом взаимодействия инстанций ([Instance Communication Service](#)).

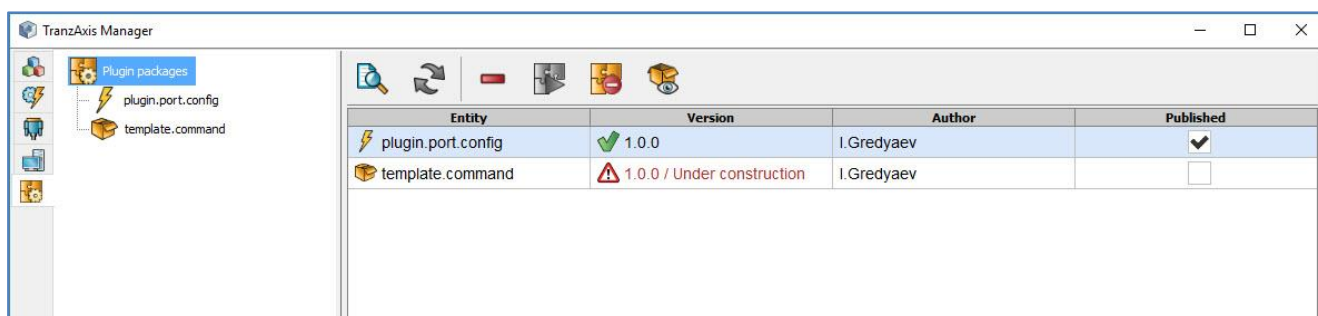
При старте приложения модуль автоматически создает подкаталог «plugins» в том же каталоге файловой системы где расположен исполняемый файл приложения. Этот каталог используется для загрузки пакетов подключаемых компонентов.

Программа предусматривает защиту от «несанкционированного» удаления файлов в данном каталоге в обход стандартного механизма удаления и подчищает ссылки в БД автоматически.

### Понятия

- **Плагин** – реализация подключаемого компонента.
- **Пакет** – JAR файл, содержащий один или несколько плагинов.
- **Каталог пакетов** – корневой элемент проводника модуля

Модуль представляет собой дерево сущностей-представлений загруженных пакетов, объединенных в каталог и содержащих информацию как о самих пакетах, так и о плагинах, входящих в пакет.



Изображение 4.2.5

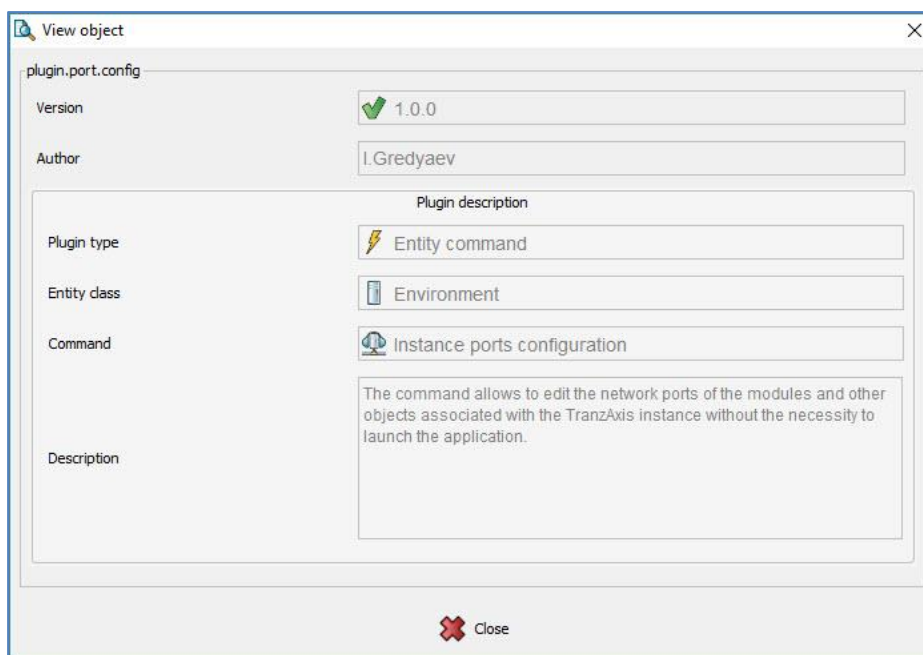
### 4.2.2.1 Каталог пакетов

Поскольку каталог пакетов содержит дочерние элементы типа «Пакет», при выделении его в дереве проводника отображается селектор установленных пакетов. Каталог имеет две собственные команды.



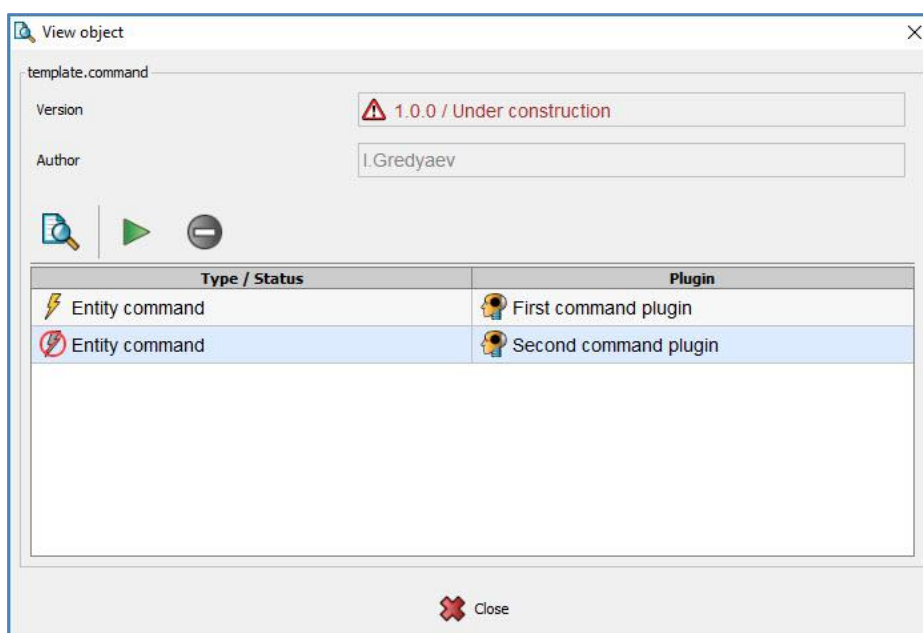
#### Команда отображения информации о пакете

Фактически является стандартной командой просмотра выделенного в селекторе объекта за одним исключением – внешний вид редактора завит от количества плагинов в пакете. Если в пакете присутствует только один плагин – в редакторе отображаются свойства пакета (версия, автор) и информация о самом плагине (производится инъекция свойств сущности типа «Плагин» в модель пакета).



Изображение 4.2.6 – сведения о пакете, содержащем единственный плагин

Если пакет содержит более одного плагина, редактор отображает только информацию о пакете (версия, автор), а плагины располагаются во вложенном селекторе.



Изображение 4.2.7 – сведения о пакете, содержащем несколько плагинов



### Команда обновления / загрузки плагинов

Команда активна, если среди внешних экземпляров приложения имеются новые плагины или обновления для уже установленных. Общее количество обновлений также отображается на кнопке команды в панели. Более подробно процедура обновления описана в главе «[Загрузка плагинов](#)».

## 4.2.2.2 Пакет

Сущность является представлением загруженного JAR файла с плагинами.



## Свойства пакета

- **Версия** – текущая версия пакета. Если пакет находится в стадии разработки, помимо номера версии указывается строка «**Under construction**».
- **Автор** – имя автора пакета, указанный при сборке пакета.
- **[Селектор] Опубликован** – статус доступности пакета для внешних инстанций.

## Команды пакета



### Команда удаления пакета

Команда производит выключение всех плагинов данного пакета, затем удаляет пакет файл JAR пакета.



### Команда включения всех плагинов пакета

Команда активна, если пакет содержит хотя бы один не включенный плагин. Только после включения плагин становится доступным в приложении. Конкретное местонахождение плагина напрямую зависит от его типа, подробнее об этом можно ознакомиться в главе «[Типы плагинов](#)».



### Команда выключения всех плагинов пакета

Команда активна, если пакет содержит хотя бы один не выключенный плагин. После выключения плагин становится недоступным в приложении.



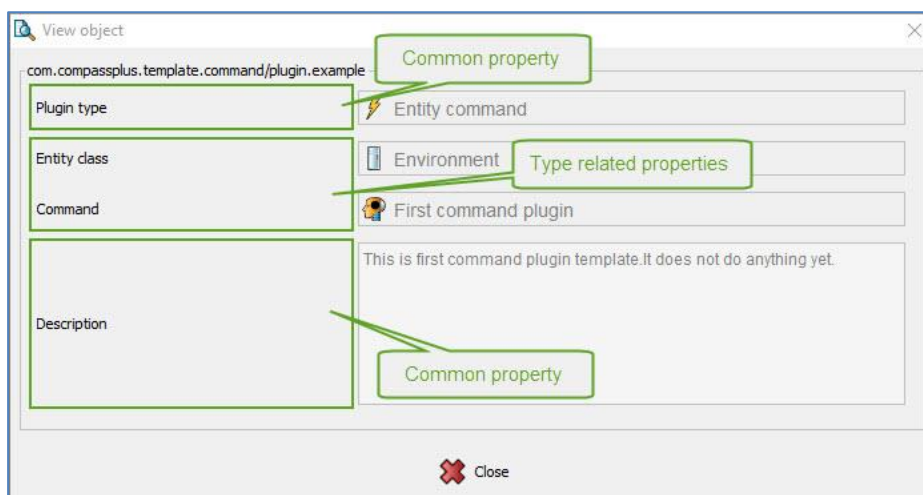
### Команда публикации пакета

Команда активна, если пакет не находится в стадии разработки. Опубликованный пакет становится доступным всем внешним инстанциям приложения.

## 4.2.2.3 Плагин

Сущность является представлением плагина, входящего в состав пакета и предназначена для просмотра информации о типе плагина и о его функциональном предназначении. Поскольку типов плагинов в общем случае несколько, данная информация должна объединять:

- Общие свойства плагинов
- Свойства конкретного типа плагина



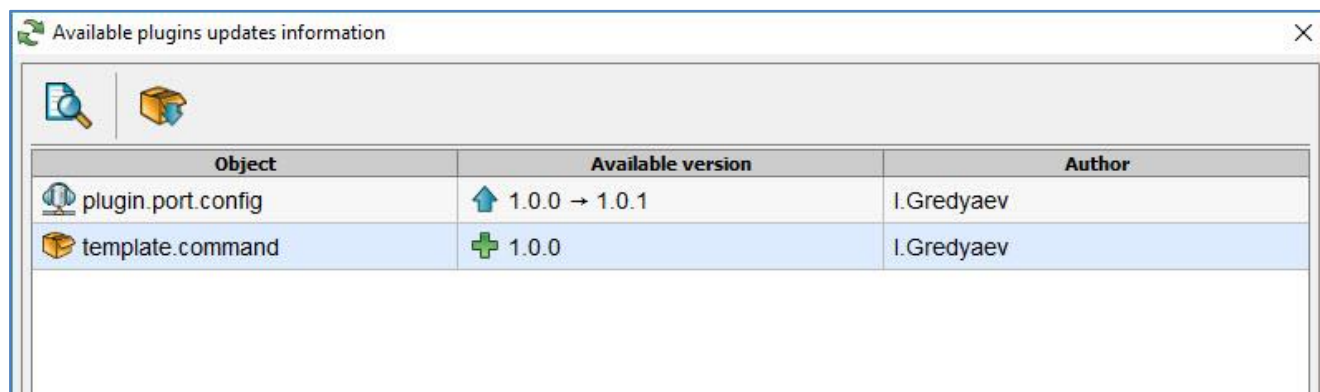
Изображение 4.2.8

## Общие свойства плагина

- **Тип плагина** – Название типа плагина и его графическое изображение.
- **Описание** – Текстовое описание функции плагина, указанное разработчиком при сборке плагина.

### 4.2.2.4 Загрузка плагинов

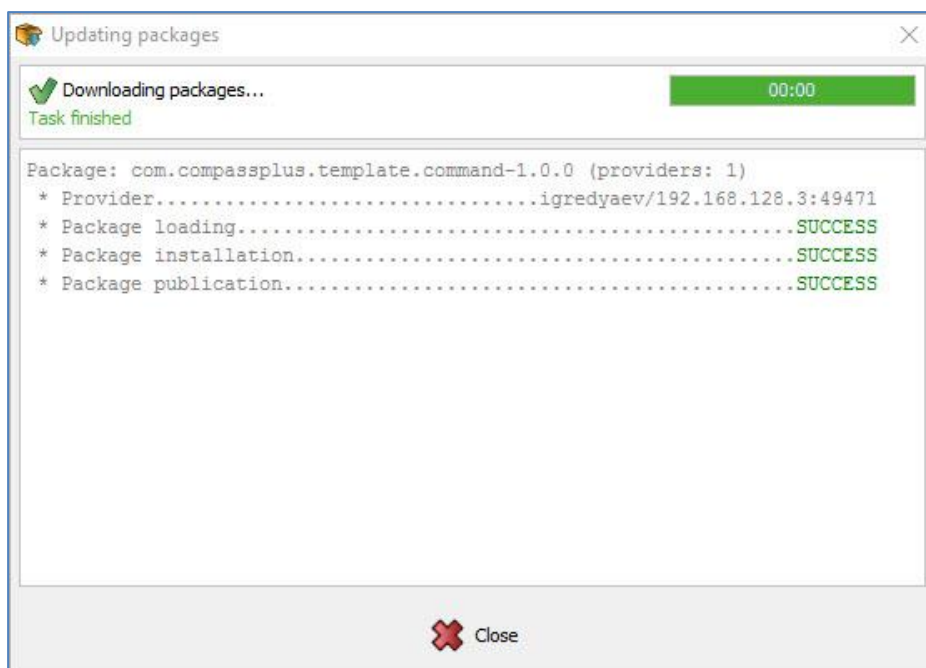
Вызов команды обновления и загрузки плагинов появляется окно со списком имеющихся новых плагинов и/или обновлений уже установленных.



Изображение 4.2.9

Список обновлений в селекторе динамический – как только пропадает последняя инстанция-поставщик обновления, элемент удаляется из селектора, а если удаляется последний элемент – окно закрывается. Тоже самое происходит и в процессе обновления, поскольку список локальных пакетов меняется, то и список обновлений должен быть пересчитан. Для загрузки пакетов следует выбрать один или несколько элементов в селекторе и вызвать команду загрузки из панели. Процесс загрузки каждого пакета происходит


- Закачка файла JAR с внешней инстанции-поставщика и проверка контрольной суммы скачанного файла.
- Удаление устаревшего пакета (если происходит обновление).
- Установка нового пакета (в случае обновления включенные ранее плагины пакета будут снова включены).
- Публикация пакета (новый пакет публикуется всегда, обновленный – только если старый был опубликован).



Изображение 4.2.10

### 4.3. Подключаемые компоненты

Приложение поддерживает следующие типы подключаемых компонентов (плагинов):

	Тип плагина	Описание
	Команда сущности	Плагин расширяет стандартные функции сущностей, имеющих в поставке приложения.

### 4.3.1 Команда сущности

Плагин, являясь командой при включении добавляется в список команд сущности, при этом размещается не в основной панели команд, а в модальном окне выбора плагина для запуска. Кроме этого, новая команда сущности становится доступной в модуле ярлыков ([Launcher](#)). Информацию о типе сущности, расширяемой плагином можно получить в редакторе пакета или плагина.

#### Свойства типа плагина

Entity class	Environment
Command	First command plugin

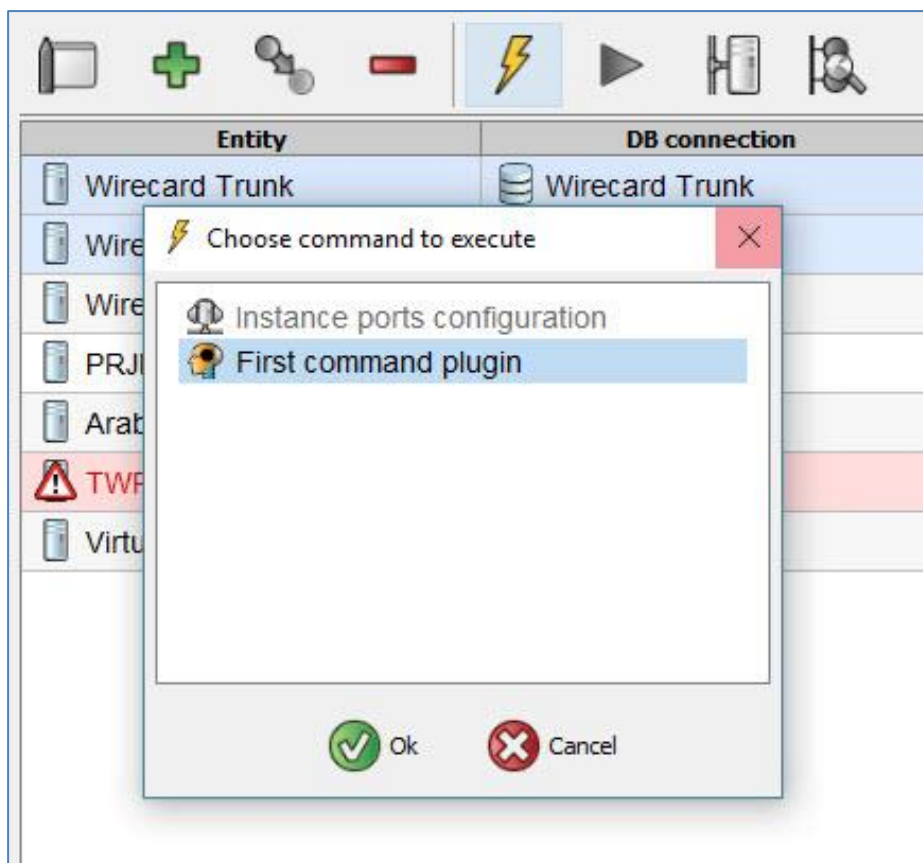
Изображение 4.3.1 – описание доступное только данному типу плагина

- **Класс сущности** – Название типа сущности, в которую будет добавлена данная команда.
- **Команда** – Название команды, которое будет отображаться в GUI.



#### Команда выбора плагина для запуска

Команда активна, если для текущего контекста (список выбранных в селекторе сущностей или сущность, редактор которой отображается в проводнике) есть хотя бы один доступный плагин (согласно их условиям применимости). При вызове команды отображается диалог выбора плагина



Изображение 4.3.2 – выбор доступного плагина для запуска

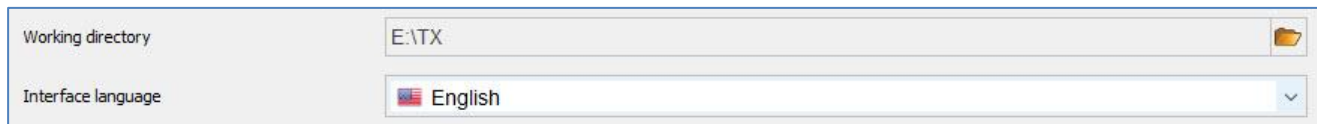
## 4.4. Сущности, команды и задачи

В рамках данного раздела будут описаны основные объекты конфигурации приложения, отображаемые в модуле проводника ([Explorer](#)), т.е. перечень сущностей, напрямую задействованных в решении прикладной задачи приложения.

## 4.4.1 Сущность «Общие настройки»

Корневая сущность в дереве проводника содержит общие настройки приложения и инструменты (выполненные в виде команд данной сущности). Сущность создается приложением автоматически при старте.

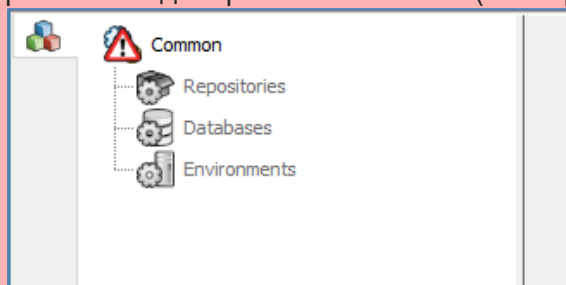
### 4.4.1.1 Свойства сущности



Изображение 4.4.1

- **Рабочая директория** – папка на диске, где приложение будет создавать файлы в процессе работы (логи, рабочие копии исходников, кэш и т.д.)
- **Язык интерфейса** – язык отображения текстовых меток объектов пользовательского интерфейса. В настоящее время поддержан русский и английский язык. По умолчанию язык выбирается в соответствии с настройками операционной системы, при изменении и после перезагрузки приложения будет установлен указанный язык.

При первом запуске приложение контролирует корректность сущности и не позволяет работать с дочерними объектами (блокирует их).

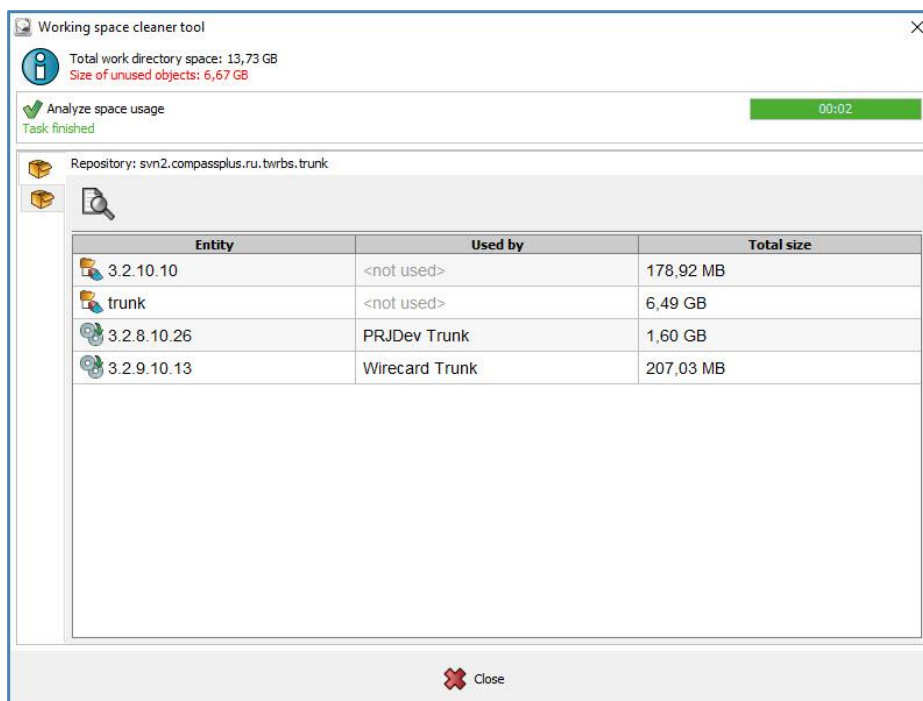


Для продолжения работы требуется заполнить все обязательные поля настроек.








#### 4.4.1.2 Команда «Утилита очистки рабочего каталога»

Запускает задачу анализа структуры рабочего каталога и выводит диалоговое окно, представляющее в графическом виде данную структуру в виде записей, а также предоставляет возможность удалить некоторые элементы из рабочего каталога. В процессе поиска и классификации объектов утилита также сопоставляет найденные элементы репозиториям SVN и при выводе объектов в GUI, каждый репозиторий с его объектами размещает на отдельной вкладке. Те объекты, которые не удалось сопоставить ни одному репозиторию размещаются на отдельной вкладке.



Изображение 4.4.2




Утилита поддерживает обнаружение и отображение следующих элементов:

	Тип объекта	Описание и правило удаления
	Рабочая копия с исходниками продукта	Объект порождается сущностью «Версия» при загрузке исходников из SVN. Удаление возможно только в случае, если соответствующий объект "версия" не используется.
	Дамп памяти	Объект порождается при аварийном завершении приложения RW Designer.
	Кэш релиза продукта	Объект порождается сущностью «Релиз» во время загрузки бинарных из SVN при запуске RadixWare приложения.
	Посторонний каталог	При поиске объектов утилита может встретить каталог, не связанный с известными ей сущностями. Также в случае недоступности SVN репозитория несвязанные сущности типа «Релиз» отображаются аналогично.
	Посторонний файл	Тоже самое что и «Посторонний каталог».



### 4.4.1.3 Параграфы

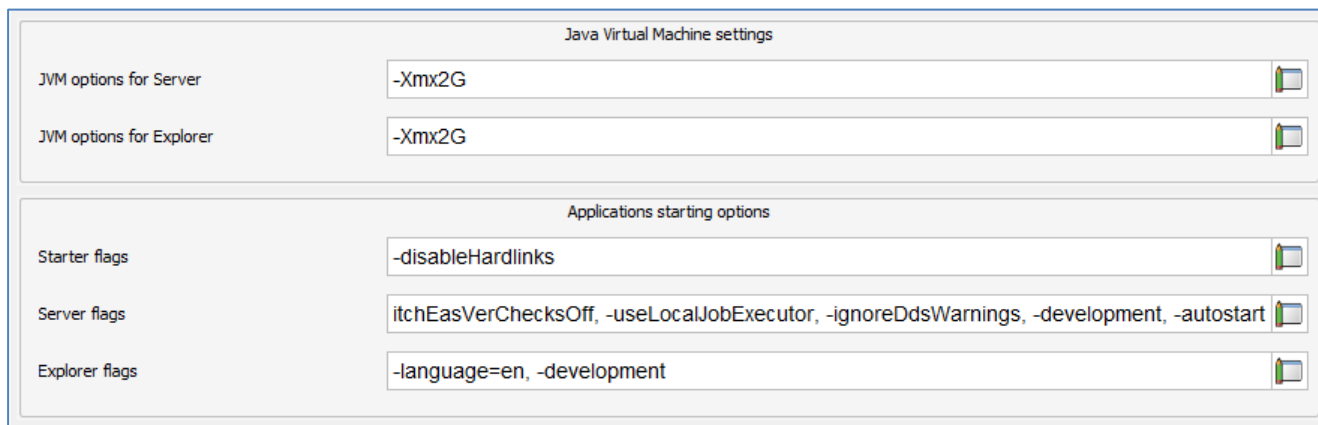
В состав приложения включены следующие параграфы:

	Параграф	Описание
	Каталог репозиториев	Пользователь может создавать и использовать различные репозитории SVN имеющих соответствующую структуру каталогов ( <a href="#">Репозиторий SVN</a> ). Каталог не имеет настроек.
	Каталог баз данных	Список баз данных ( <a href="#">База данных</a> ), используемые продуктовым приложением. Каталог не имеет настроек.
	Каталог настроек окружений	Список сущностей ( <a href="#">Окружение</a> ), используемые продуктовым приложением, совмещающие в себе все необходимые настройки для запуска продуктового приложения. Настройки каталога описаны в разделе ()

## 4.4.2 Параграф «Каталог настроек окружений»

Параграф является каталогом сущностей «[Окружение](#)», создаваемые пользователем приложения, а также содержит специфические настройки запуска серверной и клиентской частей продуктового приложения, наследуемые окружениями (но могут быть перекрыты, см. раздел «[Перекрытие свойств](#)»).

### 4.4.2.1 Настройки сущности



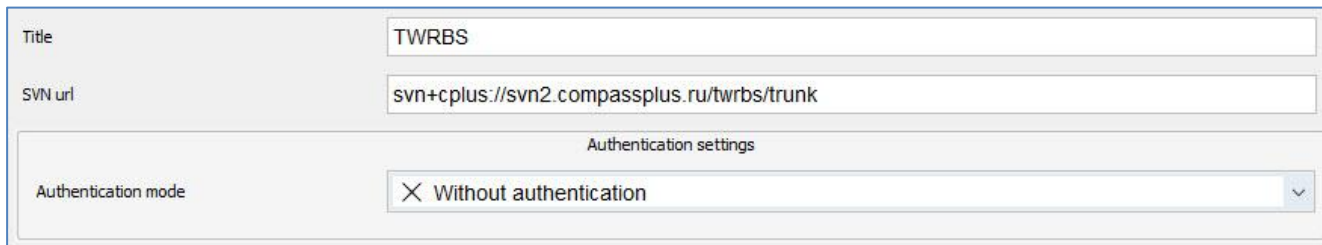
Изображение 4.4.3

- **Опции JVM для Server** – список настроек (флагов) виртуальной машины Java при запуске RadixWare Server. По умолчанию задана опция «-Xmx2G» - максимальный размер пула выделенной памяти равен 2 Гб.
- **Опции JVM для Explorer** – список настроек (флагов) виртуальной машины Java при запуске RadixWare Explorer.
- **Флаги Starter** – список настроек (флагов) стартера RadixWare. По умолчанию задана опция «-disableHardlinks» (*забыл, что это такое*).
- **Флаги Server** – список настроек (флагов) приложения RadixWare Server. По умолчанию заданы опции:
  - «-switchEasVerChecksOff» – сервер не будет проверять соответствие источника бинарных файлов (SVN репозиторий) сервера и проводника.
  - «-useLocalJobExecutor» – при выполнении запланированных задач будет использоваться локальный модуль исполнения задач, а не передаваться в том числе и на модули других инстанций TranzAxis.
  - «-ignoreDdsWarnings» – игнорировать ошибки проверки структур объектов БД.
  - «-development» – включение дополнительных возможностей для разработчика.
  - «-autostart» – после запуска сервер попытается автоматически запустить инстанцию TranzAxis.
- **Флаги Explorer** – список настроек (флагов) приложения RadixWare Explorer. По умолчанию заданы опции:
  - «-language=en» – использовать английский язык интерфейса по умолчанию.
  - «-development» – включение дополнительных возможностей для разработчика.

### 4.4.3 Сущность «Репозиторий SVN»


В данном каталоге пользователем вручную создаются ссылки на репозитории SVN, содержащие структуру каталогов, создаваемую приложением RadixWare Manager. Сущности данного класса используются приложением для выбора источника бинарных файлов продукта для его запуска.

#### 4.4.3.1 Свойства сущности



The screenshot shows a web form for configuring an SVN repository. It has three main sections: 'Title' with a text input containing 'TWRBS'; 'SVN url' with a text input containing 'svn+cplus://svn2.compassplus.ru/twrbs/trunk'; and 'Authentication settings' which contains a dropdown menu for 'Authentication mode' currently set to 'X Without authentication'.

Изображение 4.4.4

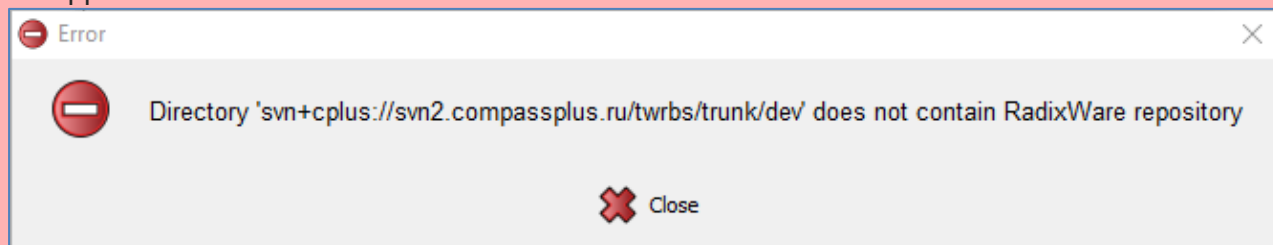
- **Наименование** – название сущности.
- **SVN url** – ссылка на каталог SVN, имеющий структуру каталогов и файлов, создаваемую приложением RadixWare Manager при развертывании продуктового приложения.
- **[Редактор] Режим аутентификации** – в настоящее время поддерживаются следующие виды аутентификации:
  - ✕ «Без аутентификации» – если возможен анонимный доступ к SVN или в операционной системе уже сконфигурирован туннель SSH с настройками доступа к нему
  -  «Логин / пароль» – если SVN настроен на простой доступ по логину/паролю пользователя.
- **[Редактор] Имя пользователя** – имя пользователя SVN. Доступен для ввода при выборе режима аутентификации «Логин / пароль».
- **[Редактор] Пароль** – пароль пользователя SVN. Доступен для ввода при выборе режима аутентификации «Логин / пароль».



#### 4.4.3.2 Команда «Загрузка/выгрузка репозитория»

Команда предназначена для загрузки структуры репозитория и создания вложенных объектов. Команда выполняется вручную пользователем или автоматически при старте приложения (если репозиторий был загружен перед закрытием приложения). В случае ручного запуска все ошибки процесса загрузки отображаются в окне предупреждения, в противном случае – записываются в лог приложения.

Перед загрузкой репозитория приложение проверяет корректность пути по наличию в корне файла "config/repository.xml". Если данный файл не обнаружен, путь к репозиторию считается некорректным.





Для ускорения запуска приложения можно выгрузить репозиторий, все имеющиеся в нем сущности, используемые где-либо в приложении, будут доступны, но без возможности выбрать что-то другое.

Пример:

Скомпилированы две ветви разработки, одна из них выбрана в окружении запуска. При загруженном репозитории, можно выбрать вторую ветвь, при выгруженном – доступна только первая, поскольку приложение «не видит» вторую.

При загрузке репозитория происходит подключение к серверу SVN, обнаружение поддерживаемых объектов, создание и вставка в дерево проводника дочерних сущностей:

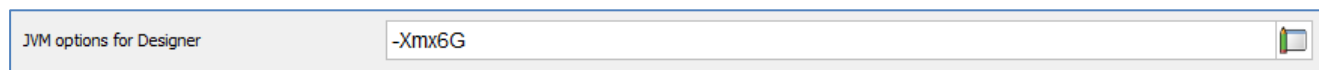
	Сущность	Описание
	Ветвь разработки	Представление ветви, содержащей исходный код версии продуктового приложения ( <a href="#">Версия</a> ). Отображается в подкаталоге « <a href="#">Исходники</a> »
	Релиз продуктового приложения	Полностью готовая (скомпилированная) версия продуктового приложения ( <a href="#">Релиз</a> ). Не отображается в дереве, поскольку не имеет значимых свойств или команд.

При невозможности загрузки репозитория по причине недоступности сервера, приложение загружает его в **offline** режиме, в котором будут доступны только загруженные ранее ветви разработки.

#### 4.4.4 Каталог «Исходники»

Используется для отображения имеющихся веток разработки продуктового приложения.

##### Настройки каталога



Изображение 4.4.5

- **Опции JVN для Designer** – список настроек (флагов) виртуальной машины Java при запуске интегрированной среды разработки RadixWare Designer. По умолчанию задана опция «-Xmx6G» - максимальный размер пула выделенной памяти равен 6 Гб.








Также эти настройки учитываются при запуске компиляции рабочей копии при помощи команд:

- «[Компиляция рабочей копии](#)»
- «[Пересборка рабочей копии](#)»






## 4.4.5 Сущность «Версия»

Сущность является моделью рабочей копии с исходниками продукта. Данный вид сущности используется разработчиком продукта для доработки функциональности и/или запуска продукта с целью тестирования. Сущности данного класса создаются программно при загрузке репозитория SVN.

### 4.4.5.1 Свойства сущности

Object	Working copy status	Working copy revision	Build status
 trunk	 Not loaded	<not defined>	<not defined>
 1.2.14	 Loaded	1186 / 21-05-2019 19:59	 1186 / Success
 1.2.13	 Not loaded	<not defined>	<not defined>

Изображение 4.4.6

- **Версия** – номер версии ветви разработки.
- **[Селектор] Состояние рабочей копии** – содержит иконку состояния и текстовое пояснение:
  -  Рабочая копия отсутствует в рабочем каталоге.
  -  Рабочая копия загружена не полностью.
  -  Рабочая копия некорректна (повреждена или отсутствует структура каталога «.svn»).
  -  Рабочая копия загружена успешно.
  -  В процессе загрузки / обновления рабочей копии обнаружены конфликты.
- **Ревизия рабочей копии** – содержит номер и время ревизии загруженной рабочей копии.
- **Собранная ревизия** – содержит иконку-признак успешности последней компиляции рабочей копии и номер ревизии, на которой компиляция запускалась.



#### 4.4.5.2 Команда «Удаление рабочей копии с диска»

Команда доступна, если рабочая копия полностью или частично присутствует на диске в рабочем каталоге, независимо от её состояния. Команда может быть выполнена для группы сущностей. Команда порождает задачу удаления директории рабочей копии, исполняемая в оперативной очереди менеджера задач, на время исполнения которой сущность блокируется.

По окончании исполнения задачи обновляются значения свойств сущности «Состояние рабочей копии» и «Ревизия рабочей копии». Если задача удаления выполнена успешно, свойство «Собранная ревизия» очищается, а сущность удаляется из БД.

При удалении сама сущность не удаляется из дерева проводника



#### 4.4.5.3 Команда «Пересборка рабочей копии»

Команда доступна если состояние рабочей копии не установлено в значение «Некорректна». Параметры команды аналогичны «[Команде компиляции рабочей копии](#)». Команда может быть выполнена для группы сущностей. Команда порождает групповую задачу, включающую в себя последовательность вложенных задач:

- Обновление рабочей копии (если рабочей копии на диске нет – «svn checkout», иначе – «svn update»).
- Очистка (если задано) и компиляция ядра продукта.
- Очистка (если задано) и компиляция модулей продукта.

исполняемая в фоновой очереди менеджера задач, на время исполнения которой сущность блокируется. По окончании исполнения задачи обновляются значения свойств сущности «Состояние рабочей копии» и «Ревизия рабочей копии».



#### 4.4.5.4 Команда «Обновление рабочей копии»

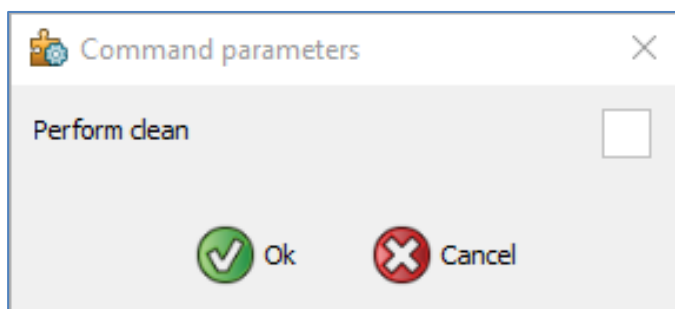
Команда доступна если состояние рабочей копии не установлено в значение «Некорректна». Команда может быть выполнена для группы сущностей. Команда порождает задачу загрузки («svn checkout») или обновления («svn update») рабочей копии с сервера в директорию рабочего каталога, исполняемая в фоновой очереди менеджера задач, на время исполнения которой сущность блокируется. Если предыдущая загрузка/обновление рабочей копии было прервано, выполняется автоматическая очистка («svn cleanup»), а затем процесс продолжается. По окончании исполнения задачи обновляются значения свойств сущности «Состояние рабочей копии» и «Ревизия рабочей копии».



#### 4.4.5.5 Команда «Компиляция рабочей копии»

Команда доступна, только если рабочая копия загружена успешно. Команда может быть выполнена для группы сущностей.

##### Параметры



Изображение 4.4.7

- **Выполнить очистку** – если установлено, перед компиляцией ядра и модулей продукта, ранее созданные бинарные файлы будут удалены.

Команда порождает групповую задачу, включающую в себя последовательность вложенных задач:

- Очистка (если задано) и компиляция ядра продукта.
- Очистка (если задано) и компиляция модулей продукта.

Задача «Компиляция ядра продукта» порождает отдельный Java-процесс, в котором выполняется очистка и компиляция файлов ядра продукта при помощи Apache Ant.

Задача «Компиляция модулей продукта» порождает отдельный Java-процесс, в CLASSPATH которого загружаются все необходимые классы продукта и платформы RadixWare, после чего запускается оригинальный компилятор RadixWare. Сообщения об этапах компиляции пробрасываются в головной процесс приложения и отображаются в виджете задачи.

По окончании исполнения задачи обновляется значение свойства сущности «Собранная ревизия».



#### 4.4.5.6 Команда «Запуск RadixWare Designer»

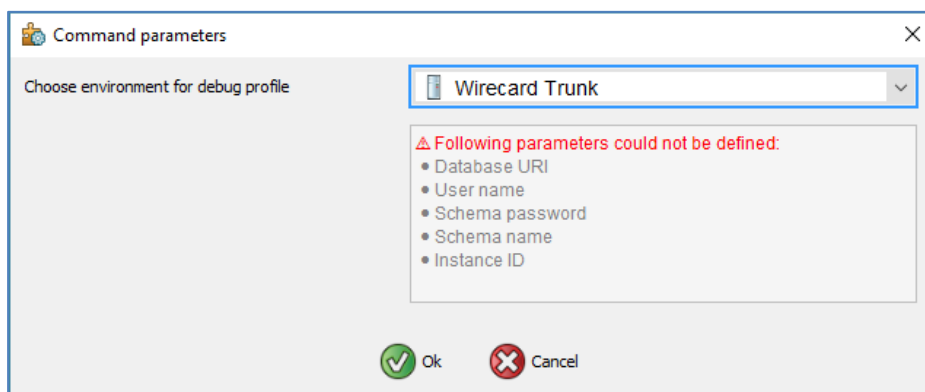
Команда доступна, только если рабочая копия загружена успешно. Команда может быть выполнена только для единичной сущности. Команда запускает интегрированную среду разработки (IDE) RW Designer, передавая ему указатель на рабочую копию, которую следует загрузить.



#### 4.4.5.7 Команда «Запуск RadixWare Designer» (с профилем отладки)

Команда аналогична предыдущей, но позволяет заполнить профиль отладчика на основе выбранной сущности типа «[Окружение](#)», которая содержит в себе все необходимые сведения для запуска продуктового приложения, которые необходимы для запуска приложения под отладчиком. Фактически, запуск продуктового приложения под отладчиком не отличается от запуска такового используя настройки окружения, за исключением возможности проводить отладку кода серверной и клиентской частей приложения.

#### Параметры



Изображение 4.4.8

- **Выбрать окружение для профиля отладки** – В выпадающем списке можно выбрать подходящее окружение, настройки которого будут использованы для запуска продуктового приложения. При этом, в диалоге параметров проверяется выбранное окружение на предмет наличия всех необходимых данных, а при отсутствии чего-либо появляется блок с предупреждением.



#### 4.4.6 Сущность «Релиз»

Сущность является указателем на имеющуюся в SVN собранную версию продуктового приложения, каких-либо настроек и команд не имеет, а потому не отображается в дереве проводника.

Сущности данного класса попадают в БД только в момент установки на них внешней ссылки (в качестве значения свойства типа EntityRef) и удаляются, как только удаляется последняя ссылка.

## 4.4.7 Сущность «База данных»

Является представлением ссылки на схему базы данных Oracle, которая используется при необходимости запустить свой собственный сервер продукта (для целей тестирования).

### 4.4.7.1 Свойства сущности

Title	Wirecard Trunk
Database URI	10.7.1.55:1521/TERMINALPAB
Schema name	TX_WIRECARD_TRUNK
Schema password	TX_WIRECARD_TRUNK
Description	<not defined>

Изображение 4.4.9

- **Наименование** – название сущности.
- **[Редактор] URI подключения** – указание адреса слушателя БД в форматах
  - [IP адрес]:[порт]/[SID]
  - [имя хоста]:[порт]/[SID]
- **Имя схемы** – наименование схемы БД Oracle.
- **[Редактор] Пароль схемы** – пароль пользователя (схемы) БД Oracle.
- **Описание** – пользовательское описание к сущности.

Нужно внимательно относиться к значению поля «Пароль схемы», поскольку по окончании редактирования сущности будет произведена попытка подключения к слушателю БД. Если в результате пароль окажется неверным – есть риск блокировки пользователя БД после нескольких неудачных попыток.



### 4.4.7.2 Команда «Монитор доступности слушателя БД»

Команда не предназначена для взаимодействия с пользователем, а выполняет сугубо информационную роль. В зависимости от результата проверки порта / подключения, иконка команды меняет вид:



Проверка подключения невозможна, когда адрес слушателя не задан.



Слушатель БД недоступен по указанному адресу или попытка подключения провалилась.



Слушатель БД доступен по указанному адресу, подключение успешно.

## 4.4.8 Сущность «Окружение»

Окружение – объект, объединяющий в себе настройки для запуска продукта, а также источники бинарных файлов продукта, необходимых для его запуска.

### 4.4.8.1 Свойства сущности

Многие свойства сущности являются зависимыми друг от друга и скрываются при условии, если заполнение не имеет смысла.

Title	Wirecard Test
Top Layer URI	com.tranzaxis.wirecard
Database settings	
DB connection	Wirecard Test
Version in database	3.2.6.10.63
Instance ID	66 - igredyaev
Binaries source	
Repository	TWRBS
Binaries location	Release
Released version	3.2.6.10.63
Description	<not defined>

- **Наименование** – наименование сущности.
- **[Редактор] URI верхнего слоя** – наименование слоя, который будет подгружен при старте продуктового приложения.

Значение может быть введено вручную или выбрать из БД, если поле «Подключение к БД» заполнено.

- **Подключение к БД** – ссылка на сущность «[База данных](#)». Необходимо заполнять только если планируется запускать сервер продукта.
- **[Редактор] Версия в базе данных** – свойство отображает версию слоя продукта (URI верхнего слоя) в базе данных. Редактор свойства отображается, если указана ссылка на базу данных. Значение выводится, если задан URI верхнего слоя, и он имеется в выбранной базе данных.
- **[Редактор] ID инстанции** – идентификатор инстанции продукта, которую следует запустить при старте RadixWare Server. Редактор свойства отображается, если указана ссылка на базу данных.

Если свойство не задано, запуск RadixWare Server не возможен.

- **[Редактор] Репозиторий** – Ссылка на сущность «[Репозиторий](#)», откуда будет выбираться источник бинарных файлов.
- **[Редактор] Источник бинарных файлов** – тип дочернего каталога сущности «[Репозиторий](#)», откуда будет выбираться источник бинарных файлов. В зависимости от выбранного значения ниже будет отображаться один из редакторов:
  - **[Редактор] Скомпилированные исходники** – Ссылка на сущность «[Версия](#)»  
При старте продукта будут использоваться бинарные файлы, полученные при

компиляции рабочей копии с исходниками продукта. Для выбора доступны только загруженные и скомпилированные рабочие копии.

Если известна версия верхнего слоя в базе данных, в выпадающем списке наиболее подходящая версия подсвечивается **зеленым цветом** и выбирается по умолчанию.

- **[Редактор] Релизная версия** – Ссылка на сущность «[Релиз](#)». При старте продукта будут использоваться бинарные файлы, предварительно загруженные из каталога релиза в SVN. Свойство может быть заполнено пользователем вручную или автоматически синхронизироваться с версией в базе данных если данный режим активирован командой редактора «[Автоматический выбор релиза](#)».

Если известна версия верхнего слоя в базе данных, в выпадающем списке релиз этой версии подсвечивается **зеленым цветом** и выбирается по умолчанию.




Релизы, версия которых отличается от версии в базе только последней цифрой в номере подсвечивается **желтым цветом**.

- **[Селектор] Версия** – свойство отображает выбранный источник бинарных файлов.
- **Описание** – пользовательское описание сущности.



#### 4.4.8.2 Команда редактора «Автоматический выбор релиза»

Команда предназначена для активации автоматической синхронизации значения поля «Релизная версия» в соответствии с версией приложения в базе данных. Кнопка команды может находиться в одном из трех состояний:

-  Режим синхронизации доступен, но не активирован, пользователь самостоятельно выбирает релиз.
-  Режим синхронизации активирован, пользователь не может выбирать релиз, но он автоматически синхронизируется при старте приложения и запуске RadixWare сервера или проводника.
-  Режим синхронизации не доступен (не задана ссылка на базу данных или невозможно извлечь информацию о версии приложения).



#### 4.4.8.3 Команда «Запуск RadixWare Server и Explorer»

Команда запуска одновременно сервера и проводника RadixWare. Команда доступна, если указаны:

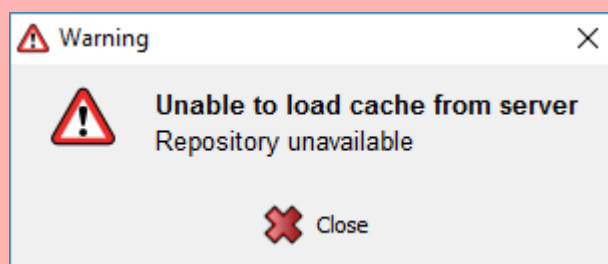
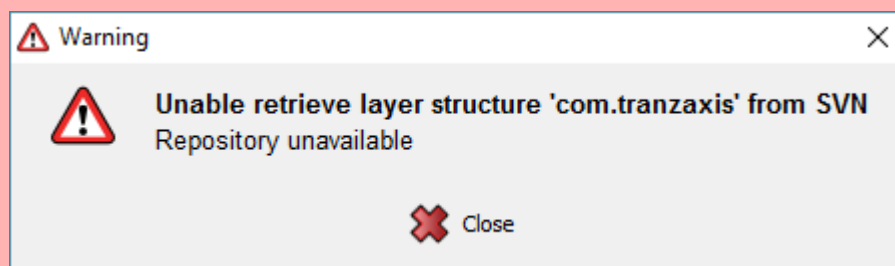
- **Uri верхнего слоя**
- **Источник бинарных файлов**
- **Ссылка на базу данных** – требуется для запуска RadixWare сервера.
- **Идентификатор инстанции** – требуется для запуска RadixWare сервера.

Если в качестве источника бинарных файлов выбрана сущность «[Релиз](#)», перед запуском запускается задача проверки наличия локальной копии файлов релиза. В случае их полного или частичного отсутствия задача догружает недостающие файлы из репозитория SVN.

Проверка включает в себя:

- Определение иерархии необходимых слоев, при отсутствии их в локальной копии задача обращается к репозиторию SVN.
- Рекурсивную проверку наличия всех файлов «layer.xml», «module.xml» и «directory.xml» которые являются индексом всех необходимых файлов.

- Рекурсивную проверку наличия всех бинарных файлов, указанных в индексе.
- При отсутствии доступа к репозиторию SVN задача отображает окно ошибки и завершается.



После успешного завершения проверки локальной копии задача запускает отдельные Java-процессы, в которых исполняются RW Server и RW Explorer и две задачи, которые ожидают завершения этих процессов. Прерывание задач приведет к завершению процессов.



#### 4.4.8.4 Команда «Запуск RadixWare Server»

Команда запуска сервера RadixWare. Команда доступна, если указаны:

- **Uri верхнего слоя**
- **Источник бинарных файлов**
- **Ссылка на базу данных**
- **Идентификатор инстанции**

Аналогично предыдущей команде запускается задача проверки локальной копии бинарных файлов, а затем запускается отдельный Java-процесс RW Server.



#### 4.4.8.5 Команда «Запуск RadixWare Explorer»

Команда запуска проводника RadixWare. Команда доступна, если указаны:

- **Uri верхнего слоя**
- **Источник бинарных файлов**

Аналогично первой команде запускается задача проверки локальной копии бинарных файлов, а затем запускается отдельный Java-процесс RW Explorer.

## 4.5. Сервисы приложения

В данный момент приложение не имеет собственных (не входящих в состав платформы) системных сервисов.

## 4.6. Сетевые сервисы

Поддержка **сетевых сервисов** – функция системного сервиса взаимодействия инстанций ([Instance Communication Service](#)), подробнее про это можно ознакомиться в разделе описания данного системного сервиса.

Приложение реализует следующие сетевые сервисы:

Имя сервиса	Назначение
<a href="#">Upgrade Service</a> Сервис обновления приложения	Обеспечивает предоставление информации о версии инстанции приложения, списка изменений между текущей версии и новой, а также отвечает за передачу файла обновленного приложения.
<a href="#">Plugin Loader Service</a> Сервис обновления подключаемых компонентов	Обеспечивает предоставление информации об имеющихся публичных плагинах или их обновлениях, сведений о составе пакетов обновления, а также отвечает за передачу файла пакета обновления.

### 4.6.1 Сервис обновления приложения

Интерфейс сервиса обновления приложения реализует следующие функции для удаленного вызова

- Предоставление о версии приложения – внешняя инстанция использует данный запрос для обнаружения более новой версии.
- Формирование списка изменений между двумя версиями - внешняя инстанция использует данный запрос для отображения диалога изменений.
- Создание и предоставление сериализуемого файлового потока, содержащего исполняемый файл приложения – используется внешней инстанцией для загрузки потока байт и записи их в локальный файл обновления.
- Предоставление контрольной суммы исполняемого файла приложения – внешняя инстанция использует его для проверки загруженного файла обновления.



## 4.6.2 Сервис обновления плагинов

Интерфейс сервиса обновления плагинов реализует следующие функции для удаленного вызова

- Предоставление информации об опубликованных пакетах плагинов для внешних инстанций.
- Уведомление подключенных инстанций о публикации пакета.
- Создание и предоставление сериализуемого файлового потока, содержащего JAR файл пакета – используется внешней инстанцией для загрузки потока байт и записи их в локальный файл пакета.
- Предоставление контрольной суммы файла пакета – внешняя инстанция использует его для проверки загруженного файла.