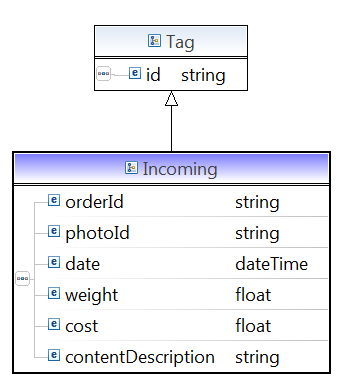
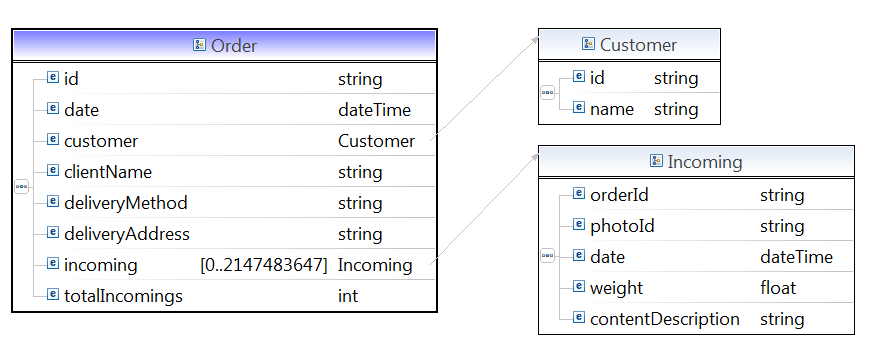
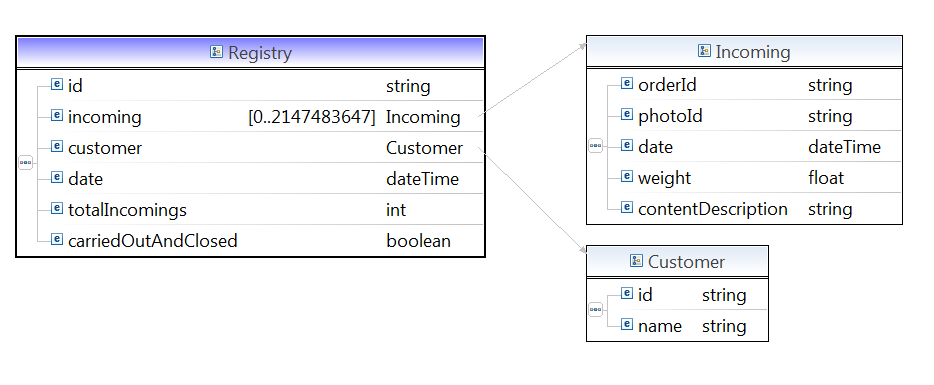
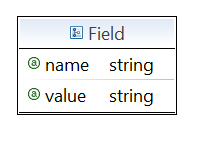
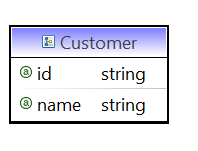
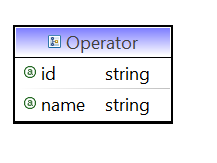
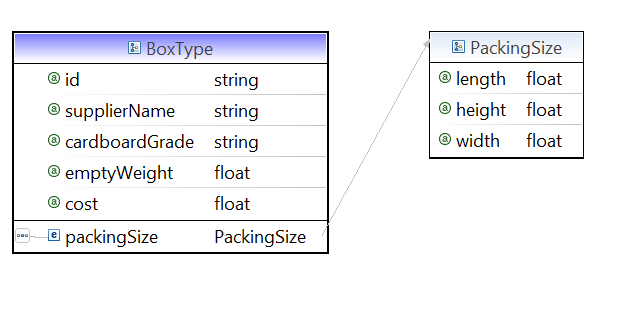
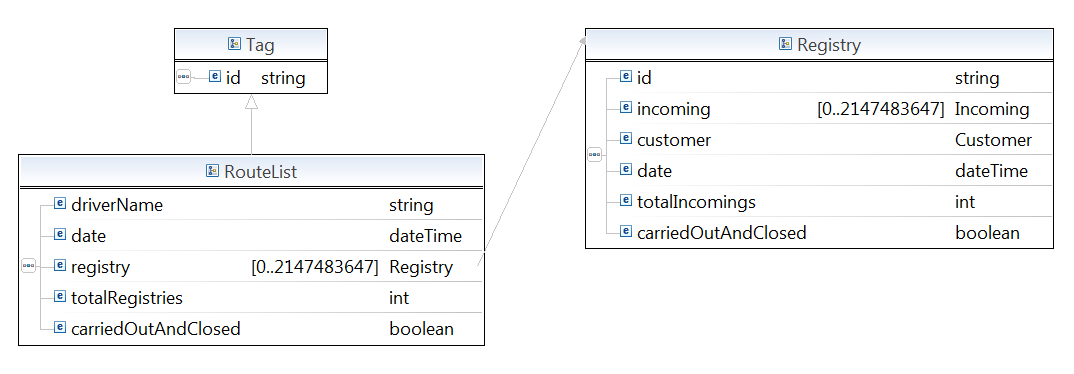
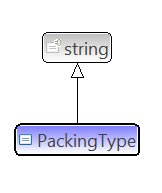
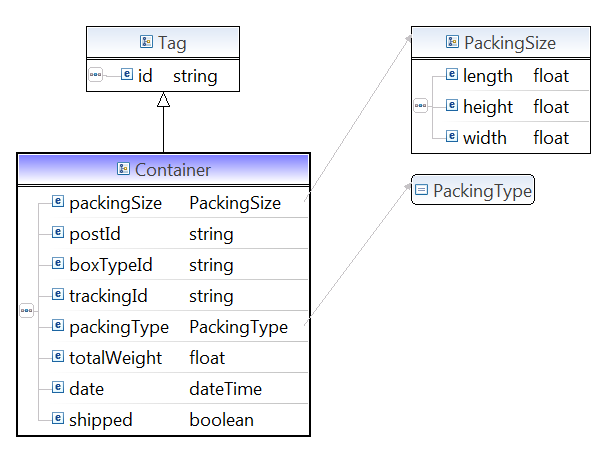
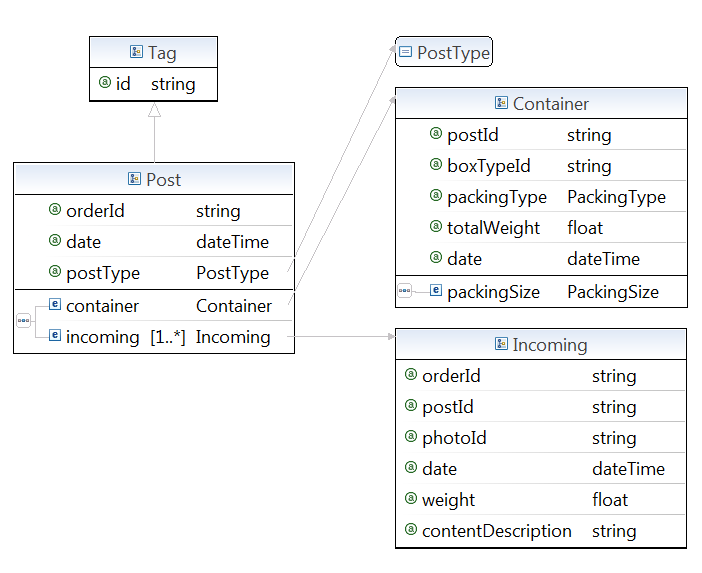
Интерфейсная спецификация упаковочной линии

# Объектная модель



# Продолжение объектной модели



# Описание объектной модели

В данной модели 7 главных сущностей:

1. Поступление (Incoming) – запечатанный пакет с маркером, который пришел от клиента (их может быть один на заказ (95%) или несколько, если заказ собирается частями или с нескольких складов). Имеет ссылку на заказ, которому оно принадлежит.
2. Реестр (Registry) – документ, к которому привязано одно или несколько поступлений от клиента;
3. Маршрутный лист (RouteList) – план водителя для забора заказов (может быть логически привязан к нескольким реестрам);
4. Заказ (Order) – содержит данные получателя, адрес доставки.
5. Отправление (Post) – целое поступление (95%) или часть поступления (исключительный случай) или несколько объединенных поступлений, предназначенных для отправления в одном контейнере. Имеет собственный код отслеживания, вес, габариты и список что там находится.
6. Контейнер (Container) – пустая коробка с идентификатором (ID контейнера) на ней, которая стоит в зоне упаковки и ждет, когда в нее поместят подходящее отправление. Контейнер может быть не коробкой, а пленкой или пакетом с произвольными размерами, которые задаются оператором непосредственно в момент упаковки вложения.
7. Тип коробки (BoxType) – справочный объект, содержащий параметры коробки: габариты (длина, ширина, высота), класс картона, вес, стоимость, название поставщика.

В 95% случаев заказ состоит из одного поступления. Поступление имеет собственный штрих код (маркер), наклеенный клиентом или с приклеенный на него бумажной накладной, распечатанной из информационной системы клиента, также содержащей маркер. После приемки и регистрации такого поступления, на упаковку оно приходит в точно таком же виде, т.е. маркер (штрих код) будет совпадать. Поэтому после регистрации (приемки) такого поступления и после того как заказ с этим поступлением проведен и закрыт, сервер должен перевести данное «поступление» в категорию «отправление», т.е. два объекта Post и Incoming будут иметь один и тот же идентификатор. Таким образом, если например код поступления равен 123, то до процедуры приемки данного поступления, функция findTag(123) вернет объект Incoming (id=123), а уже после процедуры приемки ­– объект Post (id=123).

Если несколько поступлений (Incoming) объединяются в одно отправление (Post), то объекту отправление назначается новый маркер, а между объектами поступление и отправление устанавливаются следующие взаимосвязи: поступление (Incoming) имеет ссылку на отправление (Post) через параметр PostId. Объект «отправление» хранит список поступлений, которые в него включены. Идентификаторы (маркеры) у списка поступлений и отправления в этом случае должны полностью различаться.

Если одно поступление разбивается на несколько отправлений, то создаются несколько объектов Post, которые указывают на одно и то же поступление (Incoming). Объектам Post назначаются новые маркеры, идентификаторы (маркеры) у новых отправлений и поступления в этом случае должны полностью различаться.

# Алгоритм Работы

1. Перед началом работы оператор должен просканировать свою карточку. Код карточки (логин) и системный пароль передаются в HTTP заголовке каждого запроса к почтовому сервису (HTTP based authorization). Программа проверяет правильность логина путем вызова функции GetOperator. Функция должна вернуть объект Operator с его идентификатором (это и есть код карточки) и ФИО. Имя оператора отображается в заголовке программы на всех последующих шагах программы. Если штрих код неверен, то выбросится исключение, так как HTTP-запрос завершиться отказом 401 (Unauthorized), после чего программа выведет сообщение «Неверный код».
2. В программе работает монитор активности оператора, который отслеживает время работы и время простоя, чтобы потом можно было оценить его производительность. В настройках программы параметр idleTreshold определяет максимальное время бездействия, после которого программа закрывает текущий сеанс работы и переходит на главный экран с просьбой авторизоваться.

Раз в минуту программа вызывает функцию SetOperatorActivity с параметром isActive=true – это означает, что оператор работает в программе. После выхода из программы или по автоматическому закрытию сеанса в случае бездействия, вызывается функция SetOperatorActivity с параметром isActive=false. Веб-сервис или сама система управления предприятием должна автоматически сбросить флаг активности для данного оператора, если функция SetOperatorActivity не вызывается более 2-х минут, на случай сбоя программы или разрыва соединения между программой и веб-сервисом.

1. Функция FindTag используется программой, чтобы определить, что делать дальше. Программа передает считанный сканером штрих код, в ответ на который веб-сервис должен вернуть тип объекта, которому этот код принадлежит. Объектом может быть:
   * ***Поступление*** – программа перейдет на алгоритм приемки;
   * ***Отправление*** – программа перейдет на алгоритм упаковки;
   * ***Контейнер*** – программа перейдет на алгоритм маркировки;
   * ***Маршрутный лист*** – программа запрашивает данные маршрутного листа для вывода информации на экран и запоминает его идентификатор;
   * Что-либо другое – программа выведет сообщение об ошибке.

После вызова функции FindTag, программа запрашивает одну из функций FindIncoming, FindRouteList, FindPost, FindContainer для уточнения данных по данному штрих коду.

1. Если пришло поступление, то программа:
   1. По полю orderId, которое присутствует в поступлении, загружает данные заказа (для этого используется функция GetOrder).
   2. Программа ищет реестр для данного поступления, вызывая функцию FindRegistry, куда передает идентификатор маршрутного листа (если есть) и идентификатор данного поступления. Номер реестра и принадлежность реестра к клиенту определяется по маркеру поступления. Если реестра нет, сервер должен создать новый. Если для данного клиента реестр уже создан, но еще не закрыт, сервер должен вернуть именно этот реестр. Т.е. возвращается всегда последний незакрытый реестр для данного клиента, иначе создается новый реестр.
   3. Проводиться проверка, если данный реестр уже проведен и закрыт (поле carriedOutAndClosed равно true), то программа выведет сообщение «Данный реестр уже проведен и закрыт».
   4. Проводится проверка – если данное поступление принадлежит другому клиенту, чем реестр, то программа выдает предупреждение «Поступление принадлежит другому клиенту» и возвращается на шаг 3.
   5. Проводится проверка, если данное поступление уже есть в списке поступлений реестра incomings, то программа переходит на шаг 4.10. Если данного поступления в списке нет, то программа переходит на следующий шаг.
   6. Данное поступление взвешивается на весах, значение веса записывается в поле weight.
   7. Данное поступление фотографируется, идентификатор фотографии записывается в поле photoId.
   8. Данное поступление добавляется в реестр с помощью функции AddIncomingToRegistry. В функцию передается идентификатор реестра и объект Incoming с заполненными полями веса и фотографии. Если функция выполнена успешно, то поступление считается зарегистрированным. Поступление добавляется в реестр и в заказ одновременно.
   9. Проверяется количество собранных поступлений в данном заказе:

order.incomings.size <>= order.totalIncomings

Если ожидаемое количество поступлений 1 и данное поступление уже зарегистрировано, то программа выводит сообщение «Положите поступление в тележку на упаковку», и переходит на шаг 4.10.

Если количество поступлений меньше ожидаемого, то программа выводит сообщение «Положите поступление в тележку на склад ответхранения» и переходит на шаг 4.10.

Если количество поступлений два или более и номер данное поступление является последним (или больше totalIncomings), то программа выводит сообщение «Положите поступление в тележку для сборки-объединения одного заказа из нескольких поступлений», и переходит на шаг 4.10.

* 1. Программа выводит окно с информацией о реестре и со списком зарегистрированных поступлений. В этом окне можно проверить, сколько всего должно быть поступлений, и количество уже собранных поступлений. Можно удалить поступление из списка с помощью функции DeleteIncomingFromRegistry.

В данном окне есть три кнопки:

* «Сохранить» – программа просто переходит на шаг 4.11, потому что все данные уже сохранены в системе.
* «Провести и закрыть» – программа вызовет функцию CarryOutRegistry, чтобы провести данный акт в системе, поменять его статус или выполнить другие необходимые действия. Затем программа перейдет на шаг 4.11.
* «Удалить акт» – программа вызовет функцию DeleteRegistry, после чего перейдет на шаг 4.11.

Находясь в данном окне, оператор может просканировать новый штрих код. Программа вызовет функцию FindIncoming, которая должна вернуть новое поступление. Если новый штрих код не является поступлением, или новое поступление принадлежит другому клиенту (чем реестр), то программа выдаст сообщение «Неверный код». Если же это новое поступление от того же клиента, программа переходит на шаг 4.5.

* 1. Если маршрутный лист присутствует в программе, то программа переходит в режим ожидания поступления. После сканирования оного переходит на шаг 4.  
     Если маршрутного листа нет, программа переходит на шаг 3.

1. Если пришел маршрутный лист, то программа:
   1. Проверяет, что данный маршрутный лист еще не закрыт (поле carriedOutAndClosed должно быть false), иначе выдает предупреждение и возвращается на шаг 3.
   2. Запоминает идентификатор маршрутного листа и переходит в режим ожидания поступления. После сканирования оного переходит на шаг 4.

При этом программа будет работать в рамках данного маршрутного листа, пока маршрутный лист не закроют. После чего программа вызовет функцию CarryOutRouteList, удалит маршрутный лист из памяти и перейдет на шаг 3.

Во время работы с маршрутным листом, на экране отображаются список зарегистрированных реестров. Если в момент проведения маршрутного листа количество зарегистрированных реестров меньше ожидаемого (поле totalRegistries), программа выведет предупреждение и не позволит закрыть маршрутный лист.

1. Если пришло отправление, то программа:
   1. Проводит проверку, если данное отправление уже запаковано в контейнер (поле container будет не нулевым), то программа выдаст предупреждение и вернется на шаг 3.
   2. Создает новый объект Контейнер, который связывается с отправлением (заполняется поле postId и присваивается поле container). В веб-сервисе пока ничего не вызывается.
   3. Выбирает тип упаковки, записывает выбранное значение в поле контейнера packingType. Далее:
      1. Если отправление пакуется в коробку, то программа считывает штрих код коробки, затем вызывает функцию findContainer. Контейнер должен быть пуст (поле postId будет не заполнено), иначе программа выдаст предупреждение. Размеры пустого контейнера записываются в поле packingSize, а тип коробки в boxTypeId.

Программа вызывает функцию GetBoxCount, которая должна вернуть количество пустых коробок такого же размера в зоне упаковки. Если коробок какого-то размера в зоне упаковки остается меньше заданного количества, например 10 штук (задается в настройках программы), то программа выводит предупреждение «Пополните запас коробок размера хх-хх-хх, их осталось всего 7 штук».

Штрих код записывается в поле id объекта контейнер, программа переходит на шаг 6.4.

* + 1. Если отправление пакуется в пленку или бумагу, то программа предлагает ввести размеры упаковки вручную, после чего записывает их в поле packingSize. Поле boxTypeId устанавливается при этом в null. Оператор вручную наклеивает сгенерированный ранее штрих код (см. шаг 7) на контейнер и сканирует его, программа записывает штрих код в поле id объекта контейнер и переходит на шаг 6.4.
  1. Вызывает функцию AddContainer, чтобы добавить контейнер к отправлению, после чего переходит на шаг 3.

1. Если пришел контейнер, то программа:
   1. Проверяет, что этот контейнер не пустой (поле postId будет заполнено), иначе выдает предупреждение. Проверяет, что этот контейнер еще не промаркирован (поле shipped равно false), иначе выдает предупреждение.
   2. Данный контейнер взвешивается на весах, значение веса записывается в поле totalWeight. Программа вызывает функцию UpdateContainer, чтобы обновить параметры контейнера в системе.
   3. Программа собирает всю необходимую информацию для печати этикеток и форм и печатает их. Затем программа переходит на шаг 3.

Программа печатает следующие формы: Ф-2, Ф-7, Ф-1113ен, Ф-116.

Данные для печати собираются с помощью функции GatherInfo, в которую программа передает список названий полей, которые требуется вернуть, а также идентификатор заказа. Сервис должен вернуть коллекцию объектов Field, каждый их которых содержит название и собственно данные. Поля, которые могут быть запрошены программой, указаны в приложении А спецификации.

* 1. Программа ожидает сканирования кода отслеживания данного отправления. Данный код напечатан на этикетках, которые только что были приклеены оператором. Оператору нужно просканировать данный код, чтобы программа его сопоставила с внутренним кодом и проверила, что оператор все сделал правильно. Если просканированный код совпадает с кодом отслеживания, то программа вызывает функцию MarkPostAsShipped, отмечая контейнер как отправленный, после чего переходит на шаг 3. Если код не совпадает, то программа выдает сообщение об ошибке.   
     Во время ожидания, контейнер можно перевесить, нажав на соответствующую кнопку, при этом программа перейдет на шаг 7.2.

1. В программе можно вызвать мастер подготовки маркеров и печати наклеек для резервирования пустых коробок. Задача мастера – подготовить наклейки для коробок заданного размера, связав маркеры наклеек с размерами коробок и их количеством, а также подготовить наклейки для упаковки произвольного размера (в этом случае маркеры просто резервируются в системе). При этом текущая операция прерывается, и программа отображает на экране следующие элементы:
   * Приглашение ввести штрих код типа коробки;
   * 4 кнопки для указания количества маркеров, например 5 – 50 – 500 – 5000. Указанные количества можно поменять в настройках программы.
   * Кнопку «Генерация и печать маркеров»;
   * Кнопку «Готово».
   1. Оператор может просканировать штрих код типа коробки по заранее подготовленной таблице (как на кассе супермаркета). При этом программа вызывает функцию GetBoxType, куда передает просканированный штрих код, сервер должен вернуть объект BoxType. Программа выводит параметры коробки на экран (класс картона, размеры, вес и т.п.).

Оператор может и не сканировать штрих код коробки. В этом случае программа не будет привязывать сгенерированные маркеры к типу коробки, а просто напечатает их.

* 1. По нажатию на кнопку «Генерация и печать маркеров» программа вызывает функцию GenerateTagsForContainers, которая должна вернуть список объектов Tag, в количестве указанном в поле count.

Если оператор указал тип коробки, то программа вызывает функцию addBoxContainers, чтобы связать сгенерированные маркеры с заданным типом коробки. В функцию передаются два параметра: список маркеров и идентификатор типа коробки. Сервер создает в базе данных пустые контейнеры, идентификаторы которых равняются переданным маркерам, а размер и прочие параметры коробок копируются из соответствующего типа коробки (boxTypeId).

Если тип коробки не был указан (просканирован), программа переходит к печати наклеек.

После этого программа печатает все наклейки из списка с номером маркера.

* 1. По нажатию на кнопку «Готово», программа переходит на шаг 3.

1. В программе можно вызвать мастер подготовки маркеров и печати наклеек для интернет-магазина (заказчика). Маркеры будут наклеиваться работниками магазина на новые поступления перед их отправкой на упаковочную линию. При этом текущая операция прерывается, и программа отображает на экране следующие элементы:
   * Приглашение ввести штрих код интернет-магазина;
   * 4 кнопки для указания количества маркеров, например 5 – 50 – 500 – 5000. Указанные количества можно поменять в настройках программы.
   * Кнопку «Генерация и печать маркеров»;
   * Кнопку «Готово».
   1. Сперва оператор должен просканировать штрих код магазина по заранее подготовленному заданию. При этом программа вызывает функцию GetCustomer, куда передает просканированный штрих код, а сервер должен вернуть объект Customer. Программа выводит название магазина на экран.
   2. По нажатию на кнопку «Генерация и печать маркеров» программа вызывает функцию GenerateTagsForIncomings, которая должна вернуть список объектов Tag, в количестве указанном в поле count.

После этого программа печатает все наклейки из списка с номером маркера.

* 1. По нажатию на кнопку «Готово», программа переходит на шаг 3.

1. В программе можно вызвать функцию контроля возвратов от перевозчиков. При запуске этой функции программа сначала просит выбрать перевозчика, от которого пришли возвраты.
   1. Программа простит отсканировать листы реестра, конвертирует полученные сканы в единый pdf-файл.
   2. Программа требует прочитать штрихкод реестра. С помощью полученного штрихкода, программа загружает данные реестра методом findRegistry2(<штрихкод>). Программа осуществляет проверку, если данный реестр закрыт, выводиться сообщение об ошибке и просьба просканировать другой штрихкод.
   3. Pdf-файл выкладывается в сетевое хранилище, а ссылка на файл регистрируется в системе методом fileUpload(<штрихкод>, <путь к файлу>, <тип документа>).
   4. Программа выводит окно с информацией о реестре и списком поступлений в нем. В этом окне можно проверить, сколько всего должно быть поступлений, и количество уже зарегистрированных поступлений.

В данном окне есть три кнопки:

* «Сохранить» – программа просто переходит на шаг 3, потому что все данные уже сохранены в системе.
* «Провести и закрыть» – программа вызовет функцию CarryOutRegistry2, чтобы провести данный реестр в системе, поменять его статус или выполнить другие необходимые действия. Затем программа перейдет на шаг 3.
* «Удалить» – программа вызовет функцию DeleteRegistry2, после чего перейдет на шаг 3.
  1. Находясь в данном окне, оператор может просканировать штрих код поступления из реестра. Если параметр реестра actionType равен значению Add (добавить), то программа загружает поступление с помощью функции FindIncoming и проверяет, что такое поступление еще не добавлено в данный реестр, иначе выдает ошибку. Если параметр реестра actionType равен значению Delete (удалить), то программа проверяет, чтобы данное поступление находилось в реестре.
  2. Данное поступление взвешивается на весах, значение веса записывается в поле weight.
  3. Данное поступление фотографируется, идентификатор фотографии записывается в поле photoId.
  4. Если параметр реестра actionType равен значению Add (добавить), то данное поступление добавляется в реестр с помощью функции AddIncomingToRegistry2. Если параметр реестра actionType равен значению Delete (удалить), то данное поступление удаляется из реестра с помощью функции DeleteIncomingFromRegistry2.

В функции передается идентификатор реестра и объект Incoming с заполненными полями веса и фотографии.

Если функции выполнены успешно, то поступление считается зарегистрированным, после чего программа возвращается на шаг 10.5.