***Интеграция программных систем и продуктов — это обмен данными между системами с возможной последующей их обработкой.***  
  
Смысл интеграции заключается в том, чтобы данные, которые пользователь вводит в одну систему, автоматически переносились в другую. Продукт, в который пользователь вводит данные, называется источник. А получатель данных, соответственно, приемник.  
  
Достаточно часто данные переносят в обе стороны, например, после преобразования в системе-приемнике результаты отправляются обратно в источник. А потому интеграция бывает как односторонней, так и двухсторонней.  
  
Например, если вы объединяете конфигурацию 1С: Торговля с 1С: Бухгалтерией, вам может потребоваться передать данные по всем продажам в бухгалтерию, а обратно получить сведения об оплате по этим продажам.  
  
Лично я делю процесс интеграции на такие этапы:

1. Определяем, какой продукт является источником, какой – приемником.
2. Сопоставляем объекты между источником и приемником.
3. Выбираем протокол для интеграции
4. Проводим постобработку данных (после переноса в одну из сторон)

Я всегда придерживаюсь этой последовательности при планировании работ по интеграции. Это помогает работать системно, не упустить ни одного важного момента и провести интеграцию таким образом, чтобы клиенту было удобно работать в объединенной системе.  
  
***Важно: при интеграции различных программных решений нужно хорошо понимать их функционал.***  
  
Когда-то я и сам совершал такую ошибку, и брался за интеграцию программных продуктов, которые я недостаточно хорошо знал. А потому могу сказать точно: изучать программный продукт в процессе интеграции – это не совсем корректно. При таком подходе чаще всего возникает множество ошибок и проблем, например, перенос не тех данных или сбои в работе. Рекомендую сначала хорошо изучить программный продукт, понять, что он может, каким образом в нем реализованы те или иные функции, и только потом заниматься интеграцией.  
  
В принципе, в процессе интеграции вам может потребоваться и более сложный обмен, и придется вводить, например, трех- или четырехстороннюю интеграцию. Но, по сути, эти процессы ничем не отличаются от обычного одно- или двухстороннего процесса. А потому я буду говорить об интеграции односторонней. А в конце скажу пару слов об особенностях двухсторонней. Все остальные направления вы всегда сможете выстроить по аналогии.

#### Выбираем источник и приемник

Для каждого случая интеграции данных важно четко определить, какая система будет источником, а какая – приемником.  
  
Например, у вас есть система CRM и программа 1С: Торговля. В обеих системах существует такое понятие, как контактное лицо. В принципе, вводить его вы можете и с одной, и с другой стороны. В данном случае, очевидно, что источником стоит назначить CRM, так как этого требует логика работы с любой CRM-системой.  
  
Аналогично и в других случаях. Нужно понимать, в какой системе пользователь будет вводить данные, а какая станет получателем этих данных через интеграцию. Это обязательно согласовывается с клиентом (пользователем), кроме случаев, когда источник очевиден. при этом обязательно нужно поставить в известность клиента, что данные определенного типа следует вводить именно через систему-источник.

#### Сопоставление объектов (данных)

Каждый раз при работе с данными нужно очень хорошо понимать, что именно вы выгружаете, в каком виде, а также, куда вы будете выгружать эти данные. В некоторых случаях в источнике у вас будет строковая переменная, а в приемнике – два или более объектов. В других важно просто правильно выбрать объект-приемник.  
  
Например, практически в любой CRM контактное лицо и клиент – это одно и то же. С другой стороны в 1С контактное лицо может быть клиентом, партнером, поставщиком. И очень важно понимать, куда именно записывать данные этого контактного лица. Также важно сопоставлять все данные до того, как начнется работа непосредственно с кодом. Для этого прекрасно подойдут таблицы или блок-схемы.  
  
Когда-то я так же, как и многие, пренебрегал этим этапом работы. Сейчас я знаю, что эти действия позволят избежать огромного количества ошибок. На какой бы стороне ни работал программист – на стороне программы-источника или приемника, такая табличка очень помогает в работе. Программист должен четко понимать, какие данные будут брать из источника, куда их нужно переносить, и как они будут обрабатываться.  
  
Например, при выгрузке контактного лица из CRM нужно четко сопоставить этот контакт партнеру или покупателю.  
Также очень важно понимать, какие преобразования потребуются для выгружаемых данных. Например, нужные для интеграции данные в источнике хранятся в качестве перечисления в виде текста. А в приемнике (пусть это будет 1С) аналогичное перечисление имеет ссылочный тип. Следовательно, вам потребуется преобразовать текст в ссылку, и уже ссылку передать в документ.  
  
***И здесь возникает проблема: требуются правила сопоставления.***  
  
Вы должны четко продумать и прописать правила сопоставления. Более того, об этих правилах необходимо оповестить ваших клиентов. Важно понимать, что клиент не видит логику работы обмена данными, он не понимает особенностей интеграции.  
  
Конечно, вы обязательно введете ограничение прав доступа, добавите другие варианты защиты. Но, как показывает практика, это не гарантирует от того, что пользователь совершит ошибку, из-за которой интеграция перестанет работать или будет работать не корректно. Это может быть кто-то из сотрудников, обладающий правами администратора, или приглашенный специалист, который дорабатывает, например, печатную форму документа, но при этом не осведомлен об особенностях интеграции.  
  
В результате возникают самые разные казусы. Например, вы используете в качестве ключевого слова для поиска при сопоставлении слово «дилер». Клиент по каким-то причинам меняет его в программе-источнике на слово «дилеры». Казалось бы, мелочь! Но эта мелочь приведет к тому, что поиск в 1С перестанет работать.  
  
Я решил эту проблему таким образом:

1. Обязательно оставляю клиенту подробно описанные правила сопоставления и пояснения, какие параметры и данные менять недопустимо.
2. Предусматриваю варианты оповещения об ошибке. Т.е. не только фиксирую проблему в логе ошибок, но и оповещаю пользователя о сбое каким-то образом: при помощи SMS, письмом на email, всплывающими уведомлениями в 1С. А иногда всеми этими способами сразу.

***Почему я пришел к такому варианту работы?***  
  
Интеграция – процесс сложный, и проблемы из-за человеческого фактора возникают достаточно часто, защититься от них практически не реально. Также бывают и программные сбои, особенно это касается таких сложных систем с большим числом багов, как программные продукты 1С. А для бизнеса очень важно, чтобы обмен данными проходил своевременно, а если возникла проблема также важно ее оперативно устранить.  
  
Например, в моей практике была ситуация, когда я провел интеграцию 1С и Oracle, причем, последний являлся программой-источником. Далее на стороне Oracle изменили одно из полей. В результате заказы перестали загружаться в 1С вообще, при этом сервер не выдавал уведомление об ошибке. Обнаружили проблему через неделю.  
  
С одной стороны, это явная недоработка отдела продаж моего клиента, так как неделю не получать ни одного заказа и не волноваться по этому поводу, мягко говоря, странно. С другой – отсутствие уведомления об ошибке я считаю собственной недоработкой. Конечно, в результате ошибки были исправлены, система дальше работала без сбоев, но теперь я всегда добавляю несколько вариантов уведомления об ошибке при передаче данных.  
  
Самые распространенные решения:

1. При помощи смс, электронного письма, всплывающих уведомлений в 1С информацию о сбое должен получить человек, который занимается обработкой заказов.
2. Для контроля аналогичное уведомление (чаще всего на email) отправляется руководителю отдела или директору компании.
3. Обязательно ведется лог-файл ошибок для того, чтобы специалист смог просмотреть все подробности.

В некоторых случаях также стоит добавлять уведомление о сбое другим лицам, этот вопрос решается с заказчиком индивидуально.  
  
Также стоит лог-файл ошибок вести максимально подробно и как можно дольше хранить историю. Не забывайте, что вы имеете дело с данными, которые имеются в одной базе данных, но отсутствуют в другой. И без подробного отчета вам будет очень сложно понять, что именно произошло в процессе передачи данных.

#### Обмен данными: писать самому или применять типовое решение?

Лично я предпочитаю всегда разрабатывать решение под заказчика. Здесь можно спорить, можно обсуждать различные варианты, но есть факт: типовые обмены данными всегда сильно перегружены возможностями, которые вашему клиенту не нужны. В результате процесс обмена значительно замедляется, а число возможных ошибок вырастает в разы.  
  
Кроме того, при выборе типового программного решения вы очень сильно зависите от поставщика программного обеспечения. Для любого исправления бага вам придется ждать выпуск очередной версии программы. Также придется подстраиваться при обновлениях под все изменения в работе, который внес разработчик.  
  
***А потому при выборе между самостоятельным написанием обмена данными и типовым решением, которое не на 100% подходит для данной ситуации, лучше писать обмен самому.***  
  
В некоторых случаях, когда типовое решение действительно на 100% удовлетворяет потребности клиента, а скорость работы для него не критична, я также применяю готовые продукты. Например, при выгрузке номенклатуры и фотографий на сайт я не редко использую готовый обмен данными от Битрикс. Но только для выгрузки. Для работы с заказами я применяю самописный обмен.

#### Метод подключения: REST API, SOAP или прямое подключение к базе приемника

Выбор протокола обмена данными в большинстве случаев напрямую зависит от системы, которую вы интегрируете. В большинстве случаев программисту приходится учитывать требования обеих систем, а потому выбора как такового не существует. В тех случаях, когда система может работать с несколькими протоколами, выбирайте тот, который вам удобнее. По моему опыту, для малых и средних предприятий этот вопрос не принципиален.

#### Вопросы клиентского доступа: почему не работает обмен?

Я считаю, что обо всех возможных ограничениях в доступе нужно узнать на начальном этапе интеграции. Таким образом, вы гарантированно избежите очень распространенной проблемы:  
  
Вы внедрили интеграцию, все проверили, протестировали, убедились, что система работает. После чего пользователь обнаруживает, что обмен данными не происходит.  
  
Самые распространенные ситуации:

* Ограничение доступа по IP.
* Ограничение прав пользователя.
* Ограничение по количеству обращений к источнику или приемнику

.  
В первых двух случаях ограничения обычно связаны с политикой информационной безопасности предприятия, и решаются они на административном уровне. Для пользователей, которым потребуется работа с обменом данных, системный администратор настроит перечисленные вами права. Аналогично для ограничения по IP.  
  
В случае работы с CRM-системой ограничения обычно обусловлены оплаченным пакетом услуг. Здесь достаточно оповестить клиента о наличии такого ограничения, и, при необходимости, помочь оплатить и настроить расширенный пакет.

#### 1С идентификаторы и ошибки, связанные с ними

При интеграции с 1С очень часто ошибки обмена данных возникают из-за неверного выбора УИ (уникального идентификатора). Суть проблемы заключается в том, что объекты в 1С имеют два типа УИ: один уникален внутри выбранного типа объектов. Второй используется для работы со всей базой данных.  
  
Если вы будете проводить поиск по всему справочнику с использованием идентификатора, который предназначен для работы внутри определенного типа данных, возникнет ошибка. Объект может быть вообще не найдет, либо система найдет сразу несколько разных объектов. К этой особенности 1С нужно относиться очень внимательно.  
  
***Еще одна проблема: нет возможности привязаться к уникальному идентификатору.***  
  
Например, системой-источником является сайт, и на нем не предусмотрено отдельное поле для информации о клиенте, она идет в общем тексте заказа. В этом случае придется выбрать какой-то другой вариант идентификации, например, по email.  
  
При интеграции очень важно выбрать в источнике одно из полей, которое и станет уникальным идентификатором.  
  
Я считаю хорошим тоном дублирование этого идентификатора в двух системах. Например, если я делаю выгрузку информации из CRM в 1С, то поле-идентификатор из CRM я копирую в систему 1С. В дальнейшем весь поиск и интеграция производится по этому полю быстро и просто.  
  
В принципе, это не обязательное действие. Более того, вы будете хранить даже избыточные данные, так как у вас есть нужная информация в одной из систем, но такое дублирование повышает надежность работы обеих программ и является удобным решением для интеграции и последующей обработки данных.  
  
Например, по идентификатору, который идентичен источнику, поиск будет производиться проще и быстрее, так как он не будет требовать дополнительной обработки. Кроме того, если что-то случится с базой данных одной из систем, благодаря дублирующимся идентификаторам сопоставить данные будет намного проще.

#### Формат выгрузки

Для обмена данными используются самые разные форматы. Это может быть JSON, XML, CSV, TXT, прямой доступ к базе и т.д. У меня в этом вопросе нет каких-то определенных предпочтений. Я считаю, что здесь нужно исходить из рациональных требований проекта.

#### Постобработка

Итак, обмен данными прошел успешно. Что дальше? Я считаю, что это еще не финал интеграции, так как пользователю мало того, что данные появились в системе. Обычно ему требуется, чтобы с этими данными выполнялись какие-то действия. Что именно нужно клиенту, следует уточнить у него. Но всегда надо помнить о том, что вы работаете для пользователя, для того, чтобы ему было удобно.  
  
Для интернет магазинов при интеграции чаще всего требуются:

* Оповещение менеджера о поступлении заказа, например, при помощи sms
* Уведомление пользователей о поступлении новых заказов или другой актуальной информации по email
* Звуковой сигнал и/или всплывающее окно в 1С с напоминанием о том, что появились новые запросы или заявки

Постобработка требуется, прежде всего, для того, чтобы полученные данные прошли полный жизненный цикл, а, следовательно, приняли участие в каких-то последующих бизнес-процессах. А потому после загрузки должны запускаться оповещения или какие-то определенные процессы, например, обработка заказа.  
  
Кроме действий, которые нужно выполнить в приемнике, также часто требуется после завершения успешной передачи данных выполнить определенные действия в источнике. Что именно потребуется, вам также расскажет пользователь.  
  
Например, это может быть уведомление клиента о том, что его заказ успешно прошел выгрузку и отправлен в обработку. И здесь также может быть использовано sms, электронное письмо или просто изменение статуса заказа в системе.

#### Тестирование интеграции

С моей точки зрения интеграция – это часть (иногда частный случай) внедрения программного обеспечения. И здесь, как и для любой другой работы по внедрению ПО, потребуется тестирование программистом, потом – лично консультантом, а также различные варианты тестирования вместе с пользователями.

#### Отличие односторонней и двусторонней интеграции

На самом деле, принципиальных отличий у односторонней, двусторонней или многосторонней интеграции не существует. Суть процесса остается прежней, просто в разные моменты времени приемник и источник меняются ролями. Единственное важное правило, которое я ввел для себя и вам также советую: при двухстороннем обмене необходимо хранить уникальный идентификатор для всех систем, которые участвуют в интеграции. И я считаю, что его также стоит дублировать в обеих системах.