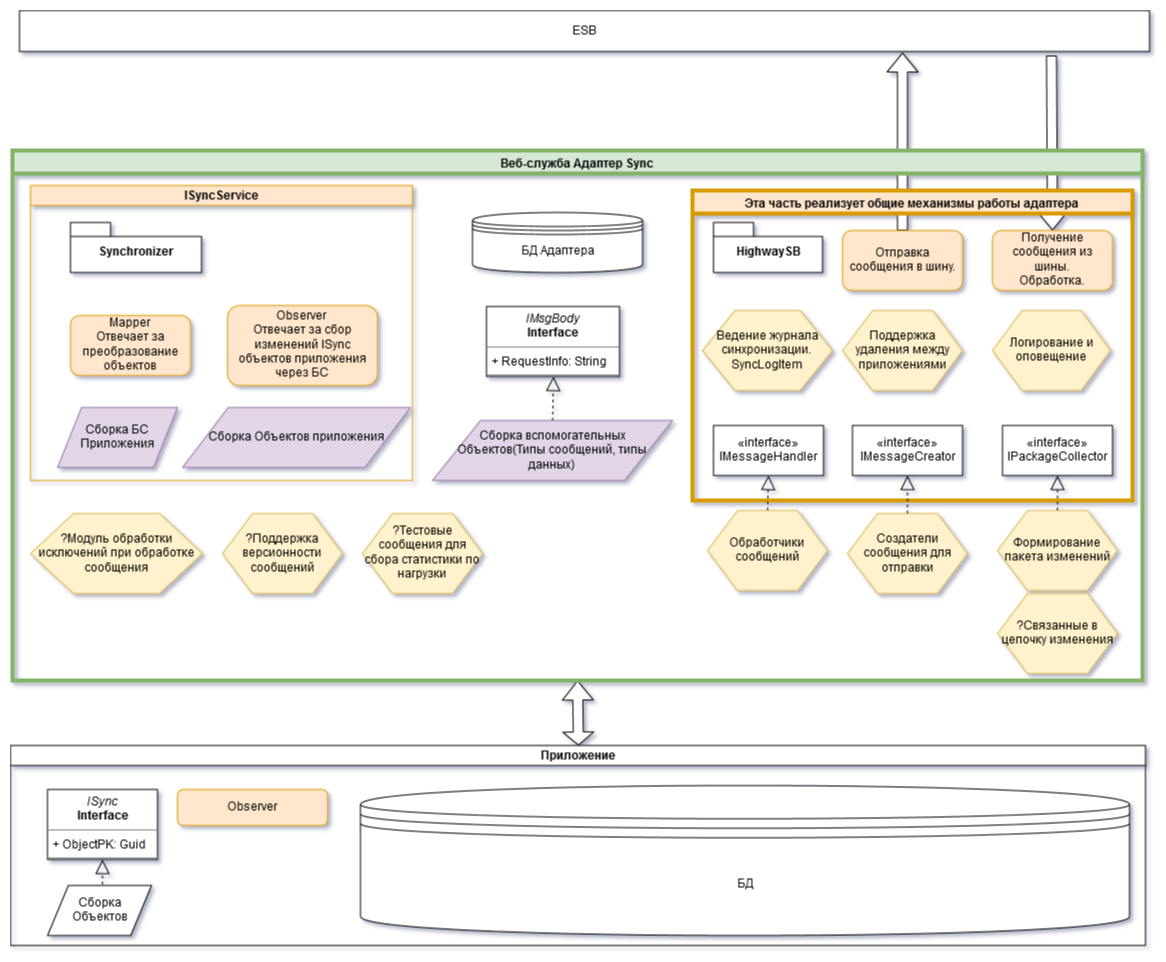
# Общие положения

1. ISyncService поддерживает изменения только из приложения. При изменениях скриптами в БД вручную создаются наборы изменений\сообщений для шины.
2. Не реализуем дополнительные модули преобразования сообщений такие как шифрование, аутентификация, защита от дублирования.
3. Мониторинг адаптера осуществляется за счет средств мониторинга шины.
4. Для каждого взаимодействия между приложениями создаём свой набор типов сообщений с префиксами направления взаимодействия, вида **ToПриемник\_fromИсточник\_НазваниеЗапроса**
5. Если не делаем обработчик исключением, то ошибки разбираются вручную администратором.

Все что помечено “?” – неутверждённые/спорные компоненты, можем отказаться от реализации или упростить её.

## Схема адаптера



Веб-служба Адаптер Sync – единая веб-служба, включающая в себя следующие компоненты:

**Сборка вспомогательных Объектов (Типы сообщений, типы данных)** – классы данных и сообщений, которые передаются через шину. Для каждого приложения поддерживается свой набор типов сообщений. Каждое сообщение начинается с префикса от источника к приемнику.

**Observer** – записывает факт-изменения объектов ISync приложения в спец. таблицу ICS\_SyncEntity, может содержать набор настроек, когда формировать факт-изменение.

**Mapper** – реализует правила преобразования объектов.

**БД Приложения** – адаптер умеет взаимодействовать напрямую с БД приложения, например, для обработки запросов на получения объектов по ИД.

**БД Адаптера**

* содержит таблицы для записи сообщений об изменении объектов ICS\_SyncEntity, ICS\_SyncSetting, ICS\_SyncSubSetting, ICS\_SyncSystem, ICS\_SyncType
* SyncLogItem – таблица для записи входящих и исходящих изменений
* SyncArchiveItem – совпадающая по структуре с SyncLogItem таблица, хранит архивные сообщения
* Log – таблица для ведения лога работы адаптера

**Отправка сообщения в шину**

* Фабрика сообщений MessageCreator
* [Отправляет сообщение в шину](#_Формирование_сообщения_для)
* Создаёт из фактов-изменений пакет для сообщения
* Проверяет логику, что в пакете одновременно не содержатся изменения на добавление и удаление сущности
* Поддержка логирования. Рассылка оповещения об ошибке
* ?Включает связанные изменения в цепочку

**Получение сообщения из шины**

* Содержит обработчики различных типов сообщений из шины IMessageHandler
* Работает через SOAP по Callback от шины
* [Принимает и обрабатывает сообщения от шины](#_Обработка_сообщения_из)
* Поддержка логирования и развернутых сообщений об ошибках при обработке сообщения. Рассылка оповещения об ошибке
* [Должен уметь принимать решение об удалении объектов, основываясь на ссылках на объект у всех приложений, участвующих в синхронизации](#_Обработка_удаления_связанных)
* ?Поддержка версионности сообщений – создание нескольких наборов обработчиков и мапперов для поддержки разной версии сообщений, чтобы не приходилось останавливать шину при изменении сообщения.

**?Модуль обработки исключений** – содержит дополнительную логику для обработки сообщений, которые не удалась обработать стандартным образом. Например, поиск потерявшегося мастера.

**?Тестовые сообщения и сбор статистики по нагрузке** – создается набор тестовых сообщений, которые отправляются с определённой периодичностью для выяснения состояния работы компонентов и их загруженности.

## Принцип работы ISyncService и сборка пакета изменений

Для работы синхронизации объекты приложения должны реализовать ISync интерфейс, который содержит свойство ObjectPK. ObjectPK – это единый ключ объекта между разными приложениями.

Observer записывает факт об изменении объекта в спец. таблицу ICS\_SyncEntity. Это происходит за счет отработки логики в БС ISyncBS, сами изменения не записываются. Для новых создаваемых ISync объектов проставляется значение ObjectPK.

Для создания пакетов изменений анализируются факты изменения объектов ICS\_SyncEntity за заданные даты.

1. Перебираем все факты-изменения из ICS\_SyncEntity по синхронизируемым типам (порядок перебора типов должен быть выстроен с учётом зависимостей) за заданные даты.
2. В рамках одного типа группируем факты-изменения по ключу.
3. Анализируем факты-изменения, если встретилось, что объект был создан и удалён, то такое изменение не будет включено в сообщение. ?Логика анализа зависимостей, проверка, что указанные мастера объекта созданы, построение цепочки пакетов с зависимостями.
4. По факту-изменению вычитывается соответствующий объект в его текущем состоянии (значение всех полей и мастеров будет на текущую дату).
5. При использовании маппера объект преобразовывается в соответствующий вспомогательный xml-объект (экземпляр AttributeDefinition), создается xml-запись изменения (экземпляр ItemDefinition), которая записывается в сообщение нужного типа.
6. Количество записей об изменении xml-объекта (ItemDefinition) одного типа в одном сообщении не должно превышать 1000, если превысило, то сохраняем пакет как готовое сообщение для шины и формируем второй пакет с сообщением для того же типа.
7. Если обработали все факты-изменения конкретного типа, то сохраняем пакет как готовое сообщение для шины и формируем сообщение для другого типа.

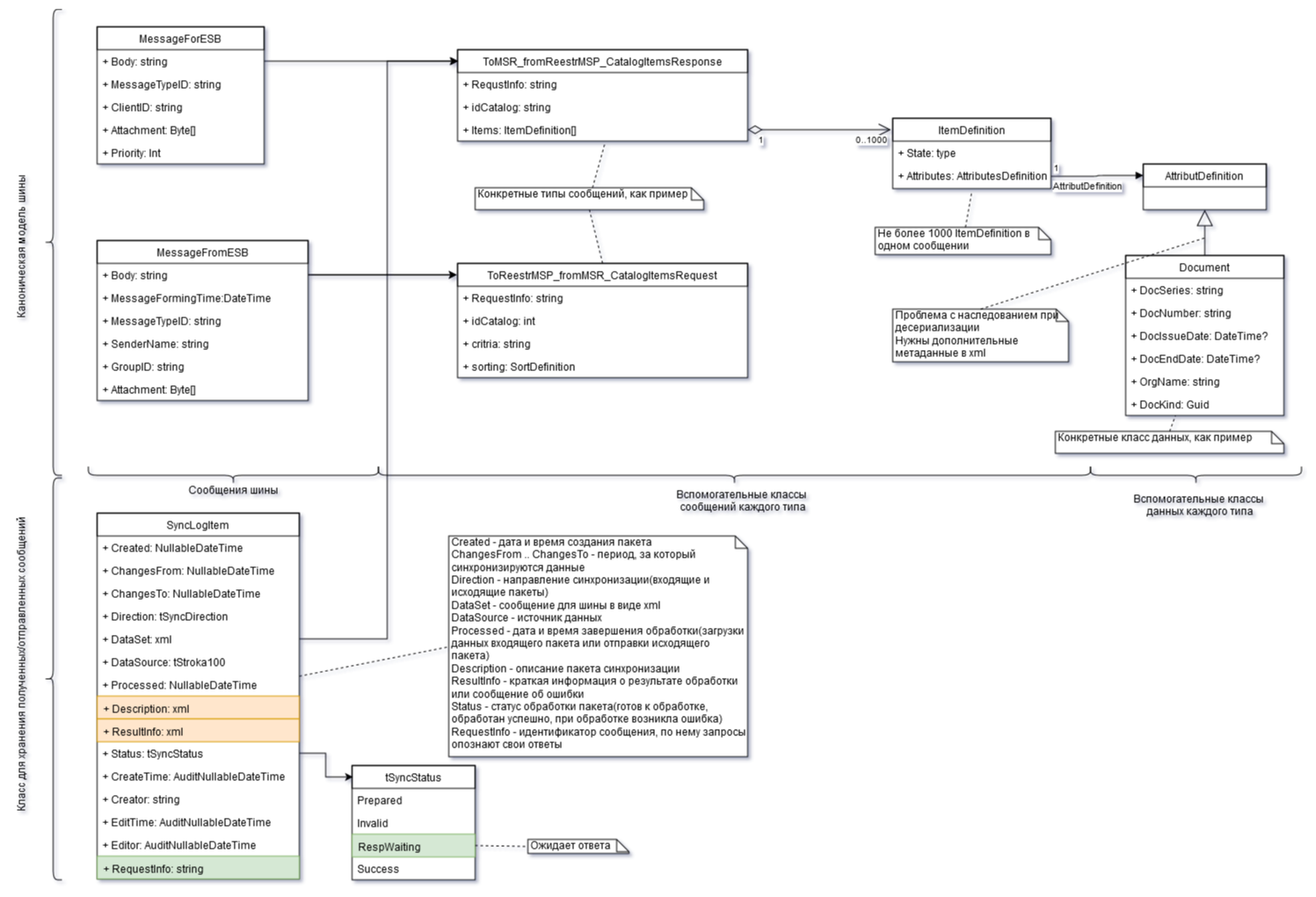
Для изменения мастеров используются сабмапперы, настройки их использования прописываются для каждого объекта.

Детейлам соответствуют отдельные мапперы, где агрегатор выступает в роли мастера.

Для преобразования из одного объекта в два используются два разных маппера с одним источником и разными приемниками.

Таким образом, если для объекта было произведено изменение, то он весь передается в сообщение в его последнем состоянии. Отсюда возникает проблема получения изменения данных за конкретный период в прошлом (что влечёт за собой использования технологического аудита и проблемы с маппингом). Принято решение: если возникнет запрос на получения изменений за старый период, то высылать пакет с изменениями, а не конкретные изменения. Для хранения пакетов используется таблица SyncArchiveItem, куда джобом будут переписываться устаревшие (настраиваемая дата устаревания) сообщения из SyncLogItem.

## Структура сообщений



### Формирование сообщения для шины

После создания пакета формируется сообщение для шины MessageForESB, создаётся SyncLogItem со статусом Prepared и в DataSet сохраняется сообщение.

По таймеру срабатывает BackgroundWorker, который ищет в таблице SyncLogItem сообщения, готовые к отправке (Direction = Out и статус Prepared), и отправляет их в шину. Если шина приняла сообщение (метод SendMessageToESB отработал без исключений), то статус SyncLogItem меняется на Success или на RespWaiting, если сообщение подразумевает получение ответа.

В конце своей обработки BackgroundWorker проверяет были ли получены ответы на сообщения со статусом RespWaiting, если период ожидание превысил настраиваемый параметр , то отправляется оповещение.

Если отправка была неуспешной, то в лог производится запись с ошибкой отправки. Если сообщение не может отправиться в течение настраиваемого времени, то высылается оповещение и статус сообщения меняется на Invalid. Отправка не будет работать пока в очереди первое сообщение Invalid.

### Обработка сообщения из шины

При получении сообщения MessageFromESB создаётся запись SyncLogItem со статусом Prepared, в DataSet записывается полученное сообщение, на этом обработчик сообщения шины AcceptMessage заканчивает работу.

По таймеру запускается BackgroundWorker, который получает все входящие (Direction = In) SyncLogItem со статусом Prepared и обрабатывает полученное сообщение (?блокировка доступа к бд на время вычитки сообщений). После успешной обработки сообщения статус SyncLogItem переводится в Success.

Если обработка прошла неуспешно, тогда сообщению присваивается статус Invalid и обработка останавливается (обработка будет пропускаться пока в очереди первое сообщение Invalid). Производится запись в лог с ошибкой обработки и отправка оповещения.

При получении сообщения смотрим на указанный RequestInfo. Если есть SyncLogItem, со статусом RespWaiting с этим RequestInfo, то обновляем его статус на Success.

## Обработка удаления связанных записей

Такой сценарий возможен, только если объект был подтянут из ЕАИСА напрямую. Для остальных случаев в БД ЕАИСа создается дубль записи, и поддерживается связность объектов между приложениями 1 к 1.

Если приходит сообщение на повторное создание (надо как-то различать их от других) объекта, то в таблицы (ObjectPK, Кратность) связностей увеличивается кратность связи. В таблице содержатся только записи с кратностью 2 и более.

При сообщении об удалении объекта, который хранится в таблице соответствий, кратность связи будет уменьшаться.

Если кратность = 1, то удалится запись о соответствии. При следующем сообщении об удалении удалится сам объект.

В таблице хранятся записи только с кратностью 2 и выше.