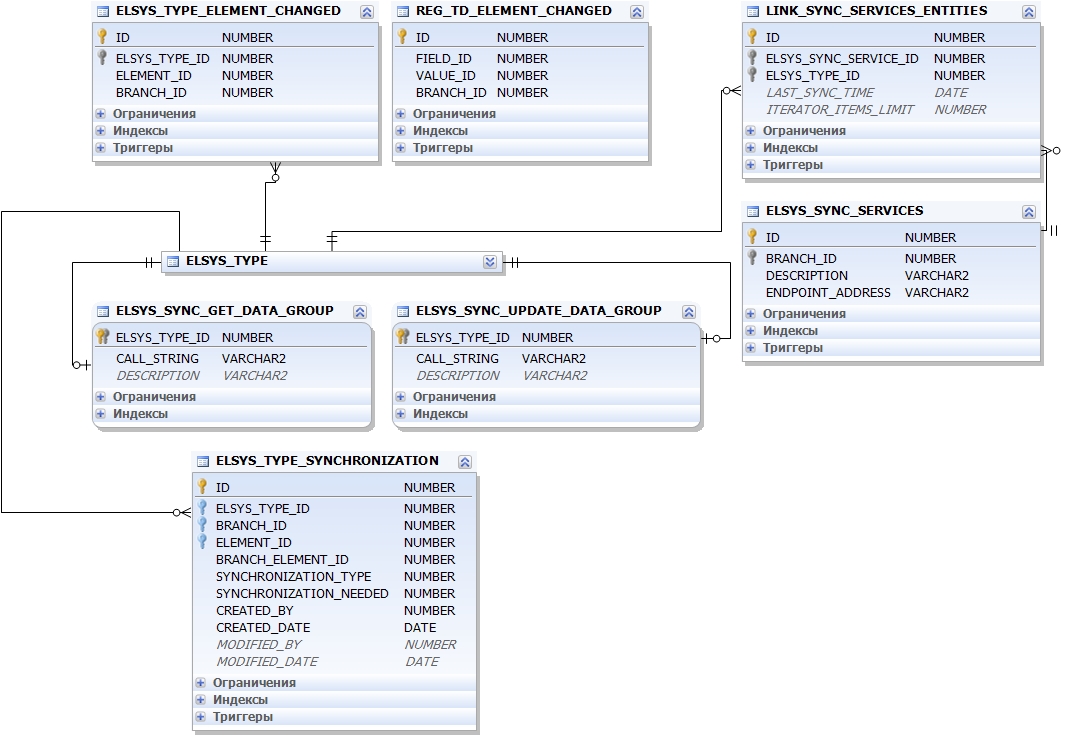
Таблицы:



Набор таблиц, используемых при синхронизации, можно разбить на четыре группы:

1. Таблицы с логами изменений элементов сущностей
   1. ELSYS\_TYPE\_ELEMENT\_CHANGED – лог изменений элементов сущностей.
   2. REG\_TD\_ELEMENT\_CHANGED – лог изменений времязависимых полей.
2. Таблицы с настройками сервисов и синхронизируемых ими сущностей
   1. ELSYS\_SYNC\_SERVICES – список сервисов-исполнителей с их параметрами и привязкой к БД филиала.
   2. LINK\_SYNC\_SERVICES\_ENTITIES – список сущностей синхронизируемых каждым сервисом.
3. Таблицы с определением сущностей-исключений из общего правила синхронизации и функциями, которые для нее вызываются.
   1. ELSYS\_SYNC\_GET\_DATA\_GROUP – список сущностей по которым реализован отдельный механизм формирования XML с данными. В поле CALL\_STRING лежит строка вызова функции.
   2. ELSYS\_SYNC\_UPDATE\_DATA\_GROUP – список сущностей по которым реализован отдельный механизм разбора XML с данными и их синхронизации. В поле CALL\_STRING лежит строка вызова функции.
4. Таблица с определением связки элементов на ЦО с элементом на филиале ELSYS\_TYPE\_SYNCHRONIZATION.

Синхронизация состоит из двух узлов: инициатора и исполнителя. Чаще всего сущности синхронизируются с ЦО на филиал. В этом случае инициатором является база ЦО, а исполнителем – база филиала. Реже наоборот. В зависимости от направления синхронизации есть некоторые отличия в алгоритмах.

Кратко синхронизацию можно разбить на следующие этапы:

1. Определение списка сущностей для синхронизации.
2. Для каждой из сущностей выполнить обработку перед синхронизаций. Чаще всего это занесение в таблицу с связями элементов ЦО-филиал данных по новым элементам сущности (ELSYS\_TYPE\_SYNCHRONIZATION).
3. Формирование фильтра синхронизируемых данных
4. Формирование XML данных
5. Передача XML данных на исполнителя и последующая синхронизация
6. Определение списка сущностей для синхронизации.

На каждую таблицу синхронизируемой сущности пишется триггер вида

CREATE OR REPLACE TRIGGER "MY\_DICTIONARY\_CHN\_LOG"

AFTER INSERT OR UPDATE OR DELETE ON MY\_DICTIONARY

FOR EACH ROW

BEGIN

IF INSERTING OR UPDATING THEN

INSERT INTO ELSYS\_TYPE\_ELEMENT\_CHANGED(ID, BRANCH\_ID, ELSYS\_TYPE\_ID, ELEMENT\_ID)

SELECT TYPE\_ELEMENT\_CHANGED\_SQGEN.NEXTVAL AS ID,

ES.BRANCH\_ID,

[MY\_DICTIONARY\_ELSYS\_TYPE\_ID] AS ELSYS\_TYPE\_ID,

:NEW.ID AS ELEMENT\_ID

FROM LINK\_SYNC\_SERVICES\_ENTITIES L,

ELSYS\_SYNC\_SERVICES ES

WHERE L.ELSYS\_TYPE\_ID = [MY\_DICTIONARY\_ELSYS\_TYPE\_ID]

AND ES.ID = L.ELSYS\_SYNC\_SERVICE\_ID;

END IF;

IF UPDATING OR DELETING THEN

INSERT INTO ELSYS\_TYPE\_ELEMENT\_CHANGED(ID, BRANCH\_ID, ELSYS\_TYPE\_ID, ELEMENT\_ID)

SELECT TYPE\_ELEMENT\_CHANGED\_SQGEN.NEXTVAL AS ID,

ES.BRANCH\_ID,

[MY\_DICTIONARY\_ELSYS\_TYPE\_ID] AS ELSYS\_TYPE\_ID,

:OLD.ID AS ELEMENT\_ID

FROM LINK\_SYNC\_SERVICES\_ENTITIES L,

ELSYS\_SYNC\_SERVICES ES

WHERE L.ELSYS\_TYPE\_ID = [MY\_DICTIONARY\_ELSYS\_TYPE\_ID]

AND ES.ID = L.ELSYS\_SYNC\_SERVICE\_ID;

END IF;

END MY\_DICTIONARY \_CHN\_LOG;

Триггер можно создать как самостоятельно, так и воспользоваться процедурой CREATE\_TRIGGERS\_CHANGE\_LOG из пакета DEV. Этот триггер заносит данные по измененным элементам сущности в таблицу логов ELSYS\_TYPE\_ELEMENT\_CHANGED. Также ведется лог изменения времязависимых полей если такие есть у сущности (он лежит в таблице REG\_TD\_ELEMENT\_CHANGED). Для определения списка сущностей необходимых для синхронизации система просматривает эти две таблицы на наличие записей. Если запись есть, то сущность синхронизируется на филиал (поле BRANCH\_ID в каждой из таблиц).

1. Обработка перед синхронизацией.

Для синхронизации, когда инициатором является ЦО а исполнителем филиал, как правило, для всех сущностей перед синхронизацией необходимо занести связку вновь созданных элементов сущности в таблицу ELSYS\_TYPE\_SYNCHRONIZATION. Это делается для того чтобы при формировании XML данных мы могли заменить все PK и FK инициатора на PK и FK исполнителя. При синхронизации с филиала на ЦО эти данные заносятся исполнителем, т.е. обработкой вызываемой на ЦО. Данные в таблице ELSYS\_TYPE\_SYNCHRONIZATION необходимы только на базе ЦО. На базе филиала их быть не должно.

1. Формирование фильтра синхронизируемых данных

Список элементов, необходимых для синхронизации, формируется в процедуре CALC\_ENTITY\_EDITED\_FILTER в пакете EL\_DIC\_SYNC\_GET\_DATA. За основу берутся данные в таблицах ELSYS\_TYPE\_ELEMENT\_CHANGED и REG\_TD\_ELEMENT\_CHANGED. Данные из этих таблиц по нужному филиалу и по необходимой сущности заносятся в временную таблицу TMP\_SYNC\_ELEM\_CHANGED с значением поля RECORD\_TYPE равным 0 (ELSYS\_TYPE\_ELEMENT\_CHANGED) и 1 (REG\_TD\_ELEMENT\_CHANGED). В таблице ELSYS\_TYPE\_ELEMENT\_CHANGED в поле ELEMENT\_ID лежат идентификаторы элементов которые были изменены. Несколько сложнее определение элементов по логу изменения времязависимых полей. В таблице REG\_TD\_ELEMENT\_CHANGED в поле FIELD\_ID лежит идентификатор времязависимого поля из таблицы ELSYS\_TABLEFIELDSTIMED. В поле VALUE\_ID лежит идентификатор записи из таблицы ELSYS\_TABLEFIELDSTIMED. TABLENAME. Если справочник имеет структуру с детальными таблицами, или любую другую из-за которой а поле REG\_TD\_ELEMENT\_CHANGED.VALUE\_ID лежит не идентификатор элемента синхронизируемого справочника, то это поле нужно преобразовать. Для этого необходимо добавить свою обработку в процедуру EL\_DIC\_SYNC\_GET\_DATA.CONVERT\_TD\_FILTER\_VALUES. Последним этапом является занесение данных в таблицу TMP\_SYNC\_FILTER\_LIST. В этой таблице лежат идентификаторы TMP\_SYNC\_ELEM\_CHANGED.VALUE\_ID. В дальнейшем именно из этой таблицы будут браться элементы для синхронизации.

Для каждой сущности для каждого сервиса можно задать максимальное количество элементов для синхронизации за итерацию. Это значение задается в поле LINK\_SYNC\_SERVICES\_ENTITIES.ITERATOR\_ITEMS\_LIMIT. Если оно равно 0, то все данные будут переданы за одну итерацию. Если указано иное значение, то весь набор синхронизируемых данных будет разбит на партии из максимум ITERATOR\_ITEMS\_LIMIT элементов и будет синхронизирован несколькими итерациями. Фильтр для текущей итерации будет занесен в переменную EL\_DIC\_SYNC.V\_ENTITY\_FILTER\_CUR. Каждая последующая итерация очищает таблицы ELSYS\_TYPE\_ELEMENT\_CHANGED и REG\_TD\_ELEMENT\_CHANGED с элементами, которые были синхронизированы в предыдущей.

1. Формирование XML данных

Есть два способа формирования XML данных и их разбора и последующей синхронизации.

1. Задать структуру справочника и запустить стандартный механизм синхронизации.
2. Самостоятельно описать формирование XML и синхронизацию сущности.

Первый способ наиболее прост, так как от разработчика требуется минимум работы. Он применяется для справочников с четкой зависимостью таблиц друг от друга.

Второй способ более трудоемок и применяется для тех справочников, где зависимости не всегда четко просматриваются (как пример справочник «Налоги Сборы Отчисления», где почти каждый из элементов имеет свою уникальную структуру) или когда оптимизация времени синхронизации критически важна (например, таблица с проводками).

Для настройки синхронизации первым способом необходимо добавить настройку сущности в переменную EL\_DIC\_SYNC.V\_SYNC\_ENTITIES\_DATA. Она имеет следующий вид:

CREATE OR REPLACE TYPE "R\_SYNC\_ENTITIES" AS OBJECT

(

ENTITYTYPE\_ID NUMBER(12),

MAIN\_TABLE\_NAME VARCHAR2(30),

INITIATOR\_DATA\_CONVERT NUMBER(1),

SYNC\_ENTITIES\_STRUC T\_SYNC\_ENTITIES\_STRUC

)

В поле ENTITYTYPE\_ID лежит идентификатор сущности.

В поле MAIN\_TABLE\_NAME лежит название основной таблицы сущности. С нее начинается построение дерева для определения зависимостей между таблицами сущности.

В поле INITIATOR\_DATA\_CONVERT лежит признак того кем будут преобразованы PK и FK – инициатором или исполнителем (зависит от направления синхронизации с ЦО или на ЦО).

В поле SYNC\_ENTITIES\_STRUC лежит массив настроек структуры таблиц сущности. Он имеет следующий вид:

CREATE OR REPLACE TYPE "R\_SYNC\_ENTITIES\_STRUC" AS OBJECT

(

PRIORITY NUMBER(12),

TABLE\_NAME VARCHAR2(30),

PRIMARY\_FIELD\_NAME VARCHAR2(30),

MASTER\_TABLE\_NAME VARCHAR2(30),

MASTER\_FIELD\_NAME VARCHAR2(30),

DETAIL\_FIELD\_NAME VARCHAR2(30),

BASIC\_FILTER VARCHAR2(500),

INITIATOR\_FILTER VARCHAR2(500),

GENERATE\_ID\_STR VARCHAR2(500),

IS\_DELETING NUMBER(1),

FIELDS\_ENTITIES T\_SYNC\_FIELDS\_ENTITIES,

FIELDS\_EXCEPTIONS T\_STRLIST,

FIELDS\_UPDATE\_EXCEPTIONS T\_STRLIST,

FIELDS\_INSERT\_EXCEPTIONS T\_STRLIST

)

В поле PRIORITY лежит приоритет/порядок данной таблицы в древовидной структуре справочника.

В поле TABLE\_NAME лежит название таблицы.

В поле PRIMARY\_FIELD\_NAME лежит название поля PK в таблице.

В поле MASTER\_TABLE\_NAME лежит название Master-таблицы.

В поле MASTER\_FIELD\_NAME лежит название PK поля Master-таблицы.

В поле DETAIL\_FIELD\_NAME лежит название поля в котором лежит ссылка на Master-таблицу.

В поле BASIC\_FILTER лежит основной фильтр для получения данных из таблицы. Чаще всего применяется к получению данных из таблицы DIC\_CONTACT с фильтрацией по полю ELSYS\_TYPE\_ID и REG\_TIMEDEPENDENT\_NUMBER с фильтрацией по полю FIELD\_ID.

!!!! Важно !!!! При задании любого фильтра можно использовать алиас для таблицы равный [символ D + PRIORITY]. Например, если у справочника есть таблица с приоритетом 3 и известно что FIELD\_ID должен быть равен 4070 то строку фильтра можно задать следующим образом BASIC\_FILTER => 'D3.FIELD\_ID = 4070'.

В поле INITIATOR\_FILTER лежит фильтр на получение данных инициатором. Принцип его формирование аналогичен полю BASIC\_FILTER.

В поле GENERATE\_ID\_STR лежит строка генерации PK для таблицы. Используется при синхронизации с филиала на ЦО.

В поле IS\_DELETING лежит признак того удаляются ли записи из этой таблицы. Как правило, все детальные таблицы имеют признак 1 (удаляются) а главные таблицы не подлежат удалению.

В поле FIELDS\_ENTITIES лежит список полей PK и FK и идентификаторы сущности для преобразования их в значения исполнителя. Это поле имеет следующую структуру:

CREATE OR REPLACE TYPE "R\_SYNC\_FIELDS\_ENTITIES" AS OBJECT

(

FIELD\_NAME VARCHAR2(30),

ENTITYTYPE\_ID NUMBER(12)

)

Здесь поле FIELD\_NAME – название поля в таблице и поле ENTITYTYPE\_ID – идентификатор сущности для преобразования.

В поле FIELDS\_EXCEPTIONS лежит список полей, по которым не формируется XML данные и которые не синхронизируются.

В поле FIELDS\_UPDATE\_EXCEPTIONS лежит список полей, которые не изменяются при синхронизации. Эти поля задаются только для новых элементов.

В поле FIELDS\_INSERT\_EXCEPTIONS лежит список полей, которые не задаются при синхронизации новых элементов. Эти поля изменяются только в уже существующих элементах.

Для настройки синхронизации вторым способом необходимо сделать следующее:

1. Реализовать функции для формирования XML вида

FUNCTION GET\_MY\_ENTITY(P\_BRANCH\_ID IN DIC\_BRANCH.ID%TYPE) RETURN XMLTYPE;

и функцию разбора XML и синхронизации вида

FUNCTION UPDATE\_MY\_ENTITY(P\_IN\_XML\_DATA IN XMLTYPE,

P\_ERROR\_MSG OUT VARCHAR2) RETURN INTEGER;

При формировании XML необходимо завязываться на фильтр который лежит в переменной EL\_DIC\_SYNC.V\_ENTITY\_FILTER\_CUR.

1. Добавить данные в таблицу ELSYS\_SYNC\_GET\_DATA\_GROUP c ELSYS\_TYPE\_ID равным идентификатору нашей сущности и CALL\_STRING подобным строке 'BEGIN :RESULT := GET\_MY\_ENTITY(:L\_BRANCH\_ID); END;'
2. Добавить данные в таблицу ELSYS\_SYNC\_UPDATE\_DATA\_GROUP c ELSYS\_TYPE\_ID равным идентификатору нашей сущности и CALL\_STRING подобным строке 'BEGIN :RESULT := UPDATE\_MY\_ENTITY(:L\_IN\_XML\_DATA, :L\_ERROR\_MSG); END;'
3. При необходимости добавить секцию в процедуру EL\_DIC\_SYNC\_GET\_DATA.CONVERT\_TD\_FILTER\_VALUES по новой сущности.

Вспомогательные данные/процедуры/функции:

1. Получение идентификатора элемента на филиале по идентификатору элемента на ЦО

FUNCTION GET\_ENTITY\_BRANCH\_ELEM\_ID(P\_ELSYS\_TYPE\_LIST IN T\_IDLIST,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE,

P\_ELEMENT\_ID IN NUMBER,

P\_SILANCE\_MODE IN NUMBER DEFAULT 1) RETURN NUMBER;

FUNCTION GET\_ENTITY\_BRANCH\_ELEM\_ID(P\_ELSYS\_TYPE\_ID IN NUMBER,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE,

P\_ELEMENT\_ID IN NUMBER,

P\_SILANCE\_MODE IN NUMBER DEFAULT 1) RETURN NUMBER;

Реализована функция с двумя вариантами параметров. 1-й – сущности передаются списком, 2-й – данные получаются для какой-то одной сущности.

Набор параметров:

P\_ELSYS\_TYPE\_LIST/P\_ELSYS\_TYPE\_ID – список сущностей/сущность

P\_BRANCH\_ID – идентификатор филиала

P\_ELEMENT\_ID – идентификатор элемента на ЦО

P\_SILANCE\_MODE – признак генерации исключения при отсутствии соответствия для данного элемента на филиале. По-умолчанию 1 – исключение не генерируется. Для того чтобы генерировалось исключение необходимо передавать 0. Генерируется исключение EL\_CONSTANT.EXP\_SYNC\_LINK\_MISSING с кодом EL\_CONSTANT. EXP\_SYNC\_LINK\_MISSING\_CODE.

1. Получение идентификатор элемента на ЦО по идентификатору элемента на филиале

FUNCTION GET\_ENTITY\_ELEM\_ID(P\_ELSYS\_TYPE\_LIST IN T\_IDLIST,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE,

P\_BRANCH\_ELEMENT\_ID IN NUMBER,

P\_SILANCE\_MODE IN NUMBER DEFAULT 1) RETURN NUMBER;

FUNCTION GET\_ENTITY\_ELEM\_ID(P\_ELSYS\_TYPE\_ID IN NUMBER,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE,

P\_BRANCH\_ELEMENT\_ID IN NUMBER,

P\_SILANCE\_MODE IN NUMBER DEFAULT 1) RETURN NUMBER;

Реализована функция с двумя вариантами параметров. 1-й – сущности передаются списком, 2-й – данные получаются для какой-то одной сущности.

Набор параметров:

P\_ELSYS\_TYPE\_LIST/P\_ELSYS\_TYPE\_ID – список сущностей/сущность

P\_BRANCH\_ID – идентификатор филиала

P\_BRANCH\_ELEMENT\_ID – идентификатор элемента на филиале

P\_SILANCE\_MODE – признак генерации исключения при отсутствии соответствия для данного элемента на филиале. По-умолчанию 1 – исключение не генерируется. Для того чтобы генерировалось исключение необходимо передавать 0. Генерируется исключение EL\_CONSTANT.EXP\_SYNC\_LINK\_MISSING с кодом EL\_CONSTANT. EXP\_SYNC\_LINK\_MISSING\_CODE.

1. Получение списка связей между элементами нескольких сущностей на филиале

FUNCTION GET\_ENTITY\_SYNC\_LIST(P\_ELSYS\_TYPE\_LIST IN T\_IDLIST,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE DEFAULT NULL)

RETURN T\_BRANCH\_SYNC\_DATA;

FUNCTION GET\_ENTITY\_SYNC\_LIST(P\_ELSYS\_TYPE\_ID IN ELSYS\_TYPE.ID%TYPE,

P\_BRANCH\_ID IN DIC\_PLACE.ID%TYPE DEFAULT NULL)

RETURN T\_BRANCH\_SYNC\_DATA;

Реализована функция с двумя вариантами параметров. 1-й – сущности передаются списком, 2-й – данные получаются для какой-то одной сущности.

Набор параметров:

P\_ELSYS\_TYPE\_LIST/P\_ELSYS\_TYPE\_ID – список сущностей/сущность

P\_BRANCH\_ID – идентификатор филиала. Если передается NULL или 0 то данные выводятся по всем филиалам. В противном случае только для указанного филиала

Возвращает список типа

CREATE OR REPLACE TYPE "R\_BRANCH\_SYNC\_DATA" AS OBJECT

(

BRANCH\_ID NUMBER(12),

ELEMENT\_ID NUMBER(12),

ELEMENT\_BRANCH\_ID NUMBER(12)

)

где

BRANCH\_ID – идентификатор филиала

ELEMENT\_ID – идентификатор элемента на ЦО

ELEMENT\_BRANCH\_ID – идентификатор элемента на филиале

1. Сущности-константы

В системе есть сущности, элементы которых находятся в таблице ELSYS\_CONST или любой другой, но работа с которыми отличается тем что эти элементы абсолютно одинаковые на всех филиалах и задаются разработчиками. Синхронизация таких сущностей не производилась. Их список заносится в переменную V\_CONST\_ENTITIES\_LIST пакета EL\_DIC\_SYNC и в запросе можно получить функцией EL\_DIC\_SYNC.GET\_CONST\_ENTITIES\_LIST.

1. Список связей сущностей с таблицами.

Используется в триггере при изменении времязависимых полей для определения что именно необходимо логировать. Если связка с сущностью не указана или эта сущность не синхронизируется (отсутствует в LINK\_SYNC\_SERVICES\_ENTITIES) то изменения не логируются. Этот список заносится в переменную V\_TD\_TABLES\_ENTITIES\_LINKS пакета EL\_DIC\_SYNC и в запросе можно получить функцией EL\_DIC\_SYNC.GET\_TD\_TABLES\_ENTITIES\_LINKS.

!!! Важно !!! Если в синхронизацию добавляется новая сущность с времязависимыми данными то этот список должен быть дополнен. В противном случае невозможно будет отследить изменение времязависимых данных.

1. Список связей сущностей.

В системе встречаются сложные сущности или множественные сущности – сущности в которых может лежать несколько сущностей. Например, сущность «Необоротные активы» состоит из сущностей «Торговое оборудование», «Основные средства» и «Автотранспорт». Или может быть ситуация, когда сущность синхронизируется под одним идентификатором а используется под другим (например, аналитика «Условия работы» лежит в проводках под 7 идентификатором а синхронизируется под идентификатором равным 5211). Эти данные заносятся в переменную V\_ENTITIES\_LINKS пакета EL\_DIC\_SYNC и в запросе можно получить функцией EL\_DIC\_SYNC.GET\_ENTITIES\_LINKS.

Структура служб/сервисов синхронизации

Решение состоит из трех проектов:

1. Базовые классы
2. Классы службы и сервиса инициатора, менеджер сеансов синхронизации
3. Классы службы и сервиса исполнителя

Базовые классы лежат в проекте Upsmile.Sync.BasicClasses.



Всего можно выделить 4 класса:

1. USInServiceValues – класс содержит структуру передаваемых данных от инициатора на исполнителя.
2. USInServiceRetValues – класс содержит структуру возвращаемых данных исполнителем инициатору.
3. USSyncParams – класс содержит настройки сервиса-исполнителя.
4. SyncServicesEntities – расширение класса EntityModel’и.

Классы инициатора лежат в проекте Upsmile.Sync.Initiator.



Всего можно выделить 4 класса:

1. USInService – класс WCF-сервиса инициатора. Основной метод – EntitySync. Метод отвечает за синхронизацию сущности на филиал. Последовательность выполняемых им действия следующая:
   1. Получение параметров службы-исполнителя где нужно запустить синхронизацию.
   2. Получить набор данных для передачи на исполнителя. При этом выполняется функция GET\_ENTITY\_DATA пакета EL\_DIC\_SYNC.
   3. В случае успешного получения данных выполнить их Json-серриализацию и создать из полученных данных MemoryStream.
   4. Создать соединение с WCF-сервисом исполнителем и запустить синхронизацию с передачей стрима полученного в предыдущем пункте.
2. SyncServiceProperties – класс отвечающий за синхронизации сущности и за контролем того чтобы в любой момент времени синхронизация каждой сущности на конкретный филиал запускалась не более одного раза. Последовательность выполняемых действий:
   1. Заблокировать текущий поток. Если не получилось – значит в данный момент уже идет синхронизации этой сущности на этот филиал и синхронизация не запускается.
   2. Создать соединение с WCF-сервисом инициатором и запустить синхронизацию текущей сущности.
   3. Закрыть соединение с WCF-сервисом и освободить поток.
3. SyncServiceManager – менеджер синхронизаций. Содержит список элементов типа SyncServiceProperties и отвечает за их уникальность в списке. Последовательность выполняемых действий:
   1. Определить наличие в списке элементов типа SyncServiceProperties элемента отвечающего за синхронизацию текущей сущности на текущий филиал. В случае отсутствия создать нужный элемент.
   2. Запустить синхронизацию найденного элемента класса SyncServiceProperties.
4. USInWinService – Win-сервис, управляющий запуском синхронизации по таймеру. Создается экземпляр класса SyncServiceManager. По событию таймера определяется список сущностей необходимых для синхронизации и если такие найдены то запускает метод CallSyncData класса SyncServiceManager. Также служба отвечает за старт WCF-сервиса инициатора при ее старте и закрытии сервиса при ее закрытии.

Классы исполнителя лежат в проекте Upsmile.Sync.Executant.



Всего можно выделить 3 класса:

1. USExDicSync – базовый класс синхронизации исполнителем. Содержит алгоритм запуска методов БД для синхронизации сущности (функция UPDATE\_ENTITY\_DATA пакета EL\_DIC\_SYNC).
2. USExService – класс WCF-сервиса исполнителя. Последовательность выполняемых действий:
   1. Получение и десерриализация данных и инициатора.
   2. Запуск синхронизации вызовом метода DicSync класса USExDicSync
3. USExWinService – Win-сервис, управляющий запуском WCF-сервисом исполнителя. Его задачи – запустить сервис при старте и завершить при закрытии Win-службы.

Настройки сервисов.

На каждой базе-инициаторе заносятся данные о сервисах- исполнителях в таблицу ELSYS\_SYNC\_SERVICES. Поля:

1. ID – Идентификатор. Генерируется автоматически.
2. BRANCH\_ID – Идентификатор филиала-исполнителя.
3. DESCRIPTION – Описание сервиса. Выводится в логе.
4. ENDPOINT\_ADDRESS – Конечная точка доступа к сервису-исполнителю.

Также каждый из сервисов имеет два config-файла.

1. NLog.config – настройки логирования данных.
2. Upsmile.Sync.Executant.exe.config/Upsmile.Sync.Initiator.exe.config – настройки самого сервиса:
   1. RunSyncFrequency – Частота запуска сервиса-инициатора (Есть только в Upsmile.Sync.Initiator.exe.config)
   2. MaxPoolSize – число сессий в пуле если пул используется.
   3. Pooling – Признак необходимости использования пула сесиий.
   4. UserID/DataSource/Password – параметры подключения к БД. **!!!Лежат в незакодированном виде. Желательно со временем изменить!!!!**
   5. NeedLogSyncData – Признак того необходимо ли логировать передаваемые от инициатора на исполнителя данные. Если значение True то данные будути писаться в trace-файл при условии что уровень логирования не выше Trace. Необходимость в установке значения в True возникает только при поиске ошибок. При передаче большого набора данных возможны падения сервиса связанные с тем что NLog не может логировать очень большой набор данных.
   6. ServiceHostAddress/ServiceEndPointAddress – по этим параметрам Win-служба будет запускать сервисы при старте.
   7. OpenTimeout/ReceiveTimeout/SendTimeout/CloseTimeout – ограничения по времени на соответствующие операции сервисов.
   8. DBEncodingName – кодировка получения и занесения данных в БД инициатора или исполнителя.
   9. TransmissionEncodingName – кодировка передаваемых данных между инициатором и исполнителем.

Где запущены сервисы:

На сервере 93.157.14.7:3387(терминальный сервер ЦО):

1. C:\Program Files\Distributor\USDicSync\Initiator – инициатор ЦО.
2. C:\Program Files\Distributor\USDicSync\Executant – исполнитель ЦО.
3. C:\Program Files\Distributor\USDicSync\Initiator\_HR – инициатор Харькова.
4. C:\Program Files\Distributor\USDicSync\Executant\_HR - исполнитель Харькова.
5. C:\Program Files\Distributor\USDicSync\Initiator\_MR – инициатор Мариуполя.
6. C:\Program Files\Distributor\USDicSync\Executant\_MR – исполнитель Мариуполя.

Логи по всем сервисам лежат в папке C:\Program Files\Distributor\USDicSync\Log. На этой же машине запущены Win- службы отвечающие за запуск WCF-сервисов.

На сервере 193.23.58.40 (сервер БД Макеевки):

1. C:\inetpub\wwwroot\USDicSync\Initiator - инициатор Макеевки.
2. C:\inetpub\wwwroot\USDicSync\Executant - исполнитель Макеевки.

Логи по всем сервисам лежат в папке C:\inetpub\wwwroot\USDicSync\Log. На этой же машине запущены Win- службы отвечающие за запуск WCF-сервисов.

На сервере 195.5.19.54:33253 (сервер БД Краматорска):

1. C:\inetpub\wwwroot\USDicSync\Initiator - инициатор Краматорска.
2. C:\inetpub\wwwroot\USDicSync\Executant - исполнитель Краматорска.

Логи по всем сервисам лежат в папке C:\inetpub\wwwroot\USDicSync\Log. На этой же машине запущены Win- службы отвечающие за запуск WCF-сервисов.

На сервере 62.122.56.156 (сервер БД Донецка):

1. C:\inetpub\wwwroot\USDicSync\Initiator - инициатор Донецка.
2. C:\inetpub\wwwroot\USDicSync\Executant - исполнитель Донецка.

Логи по всем сервисам лежат в папке C:\inetpub\wwwroot\USDicSync\Log. На этой же машине запущены Win- службы отвечающие за запуск WCF-сервисов.