



Интеграция

Интеграционное приложение с ELMA365

Манифест решения

- Монолит является источником исторических данных для 365 в режиме «непрерывной миграции» (CM - continuous migration)
- Небольшие изменения в функционале монолита возможны, если они критичны для бизнеса в конкретный момент времени
- Создание новой функциональности должно быть только на стороне 365
- Плавный переход с монолита на 365 предполагает постепенный отказ от использования функционала в монолите и заменой его (с возможным рефакторингом) в 365
- Контур в монолите (не связанный с другими контурами) замененный в 365 должен быть отключен и недоступен для последующего использования
- В случае связанности контуров их отключение должно происходить без потери действующих активных связей
- Запрещается обратная синхронизация исторических данных из 365 что будет противоречить данному манифесту
- Заказчик обязан понимать и принимать положения данного манифеста для успешной реализации проекта

- Описание функционала
- Контур CRM
- Контур ЕСМ
- Контур Проекты
- Контур Управление договорами
- Точки расширения
 - IForceSyncHandler
 - IDataItemExtension
- Хелперы

- ServerHelper
- ListenerHelper
- Менеджеры
 - E365ProcessLinkManager
- Примеры
 - Реализация миграции для справочника Страна
 - Полная принудительная миграция данных сущности
 - Свойства сущности типа IEntity
 - Свойства сущности типа BinaryFile
 - Реализация миграции для документа Счет исходящий
 - Блоки в сущности

Описание функционала

Базовый (ограниченный) функционал

ITino.ELMA.E365.Common:

- Настройки подключения к ELMA365
- Пинг ELMA365 (определение статуса)
- Проверка наличия модуля на стороне ELMA365
- Принудительная синхронизация старт-флагом E365ForceSync
- Возможность включать/отключать синхронизацию:
 - Пользователи
 - Оргструктура
 - Файлы
 - Контрагенты (без учета связанных элементов CRM)
 - Контакты (без учета связанных элементов CRM)
 - Комментарии к сущностям
 - Действия для сущностей
- Проверка оргструктуры на возможность выгрузки
- Точки расширения:
 - IForceSyncHandler
 - IDataItemExtension
 - IFeedItemProvider
- Механизм маппинга идентификаторов между системами
- Возможность установить статус Приложений
- В случае невозможности связи по __id используется специальный механизм на стороне ELMA3/4
- Поддержка лимитов для SaaS редакции

ELMA3/4 ↔ ELMA365:

- Передача дополнительных (кастомных) свойств сущностей
 - Простые типы
 - Сущности
 - Файлы
 - Документы
 - Блоки в сущностях
- Произвольные сущности конфигурации
- Процессы
 - Синхронный/асинхронный запуск из скриптов процессов
 - Запуск с контролем выполнения или без
 - Простые параметры контекста
 - Сущности (справочники) в параметрах контекста

- Файлы
- Версия исполняемого процесса (переменная *processVersion*)

ELMA3/4 ↔ ELMA365:

- Получение __id для создаваемых приложений
 - Статусы выполнения процессов
-

Расширенный функционал с установленным модулем RMG 365 | Интеграция с ELMA3/4 в ELMA365:

•

ELMA3/4 ↔ ELMA365:

- Установка связи Администратора инстанса (ELMA3/4) с Супервизором системы (ELMA365)
- Перенос (создание/изменение) Пользователей
 - ФИО
 - Эл. почта
 - Сотовый
 - Рабочий
 - Дата рождения
 - Дата приема на работу
 - Фото (если в ELMA3/4 пустое, то не обновляется в ELMA365)
 - Временная зона
 - Отображаемая должность (включается в настройках)
 - Должности (привязка к оргструктуре)
- Блокировка/разблокировка Пользователей
- Перенос (создание) оргструктуры - добавление и изменение элементов
- Протоколирование проблем и предупреждений при экспорте оргструктуры и подсказки что исправить
- **TODO** Смена порядка и удаление элементов в оргструктуре

ELMA3/4 ↔ ELMA365:

- Получение версии модуля и серверного времени в UTC
 - Получение версии и редакции ELMA365
 - Получение идентификатора Супервизора системы
 - Получение идентификаторов Пользователей при переносе
 - Получение идентификаторов элементов оргструктуры
-



Ограничения

- Компоненты собираются под свежие сборки ядра 3.15 и 4.1. Кастомные сборки под старые релизы возможны по договоренности
- Не поддерживается кластер Enterprise более чем с одним активным узлом
- Смена супервизора не предусмотрена - будет потеряна связь с Администратором инстанса
- ~~Пересоздание корневого элемента оргструктуры в ELMA3/4 не поддерживается~~
- ~~Перенос (создание) пользователей в ELMA365 осуществляется только для статусов Активный (не заблокирован)~~
- В ELMA365 не переносится информация о AD/LDAP пользователей - создаются обычные пользователи с авторизацией по email. Интеграцию нужно настраивать в ELMA365 вручную.
- Не поддерживается синхронизация изменения порядка и удаления элементов оргструктуры
- В поступлениях используется валюта Рубли
- ~~Размер файла не может быть больше 50 Мб~~
- ~~Не поддерживаются блоки~~
- Не поддерживаются блок в блоке
- Передача блоков при запуске процессов не предусмотрена
- Переносится только заполненный аватар пользователя. Если его удалить в ELMA3/4, то он останется в ELMA365. Это связано с особенностями формирования "пустых" аватаров в ELMA365.

Контур CRM

Функционал управления клиентами

ITino.ELMA.E365.CRM:

- Возможность включать/отключать синхронизацию справочников и сущностей
- **BackLog** Принудительная инициализация старт-флагом E365InitCRM
- Маппинг базовых сущностей в ELMA3/4 и ELMA365
 - ФЛ - Контакт
 - ЮЛ - Компания
 - Контакт ЮЛ - Контакт
 - Отрасль - Отрасль
 - Источник возможности - Источник лидов
 - Возможность - Лид
 - Контакт возможности - Контакт
 - Сделка - Сделка (Основание)
 - Поступление - Поступление (план)
 - Валюта - Валюта
 - Маркетинговое мероприятие - Маркетинговое мероприятие
- Маппинг сущностей ITino.ELMA.CRM в ELMA3/4 и ELMA365
 - Бизнес группа - Компания
 - Контакт Бизнес группы - Контакт
 - Филиал - Компания
 - Контакт Филиала - Контакт
 - Предприниматель - Компания
 - Контакт Предпринимателя - Контакт
- Маппинг сущностей ITino.ELMA.CRM.RU в ELMA3/4 и ELMA365
 - ЮЛ РФ - Компания
 - Контакт ЮЛ РФ - Контакт
 - ИП РФ - Компания
 - Контакт ИП РФ - Контакт
 - Банковский реквизиты РФ - Банковский реквизиты (Компания)

ELMA3/4 ↔ ELMA365:

- Отрасли
 - Название
- Источники лидов
 - Название
- Компании - наполнение полей определяется в зависимости от передаваемого объекта

- Контакты - наполнение полей определяется в зависимости от передаваемого объекта
- Лиды
 - Статус
 - Название
 - Ответственный
 - Компания
 - Контакты (контроль создания дублей для Компании и Лида)
 - Источник лида
 - UtmCampaigning
 - UtmSource
 - UtmMedium
- Сделки
 - Статус
 - Название
 - Компания
 - Бюджет (поддержка валюты)
- Поступления (план)
 - Контрагент
 - Основание (сделка)
 - Сумма поступления
 - Плановая дата
 - Фактическая сумма (равна сумме поступления)
 - Фактическая дата
 - Статус (Ожидает, Оплачено, Отменено)
- Валюты
 - Название
 - Код
- Маркетинговые мероприятия
 - Название
 - Описание
 - Ответственный
 - Дата начала
 - Дата окончания
 - Бюджет

ELMA3/4 □ ELMA365:

Контур ЕСМ

Функционал управления документооборотом

ITino.ELMA.E365.Documents:

- Возможность включать/отключать синхронизацию справочников и сущностей
- Перенос номенклатуры при публикации
- Перенос информации о регистрации документа (поддерживается множественная регистрация в разных делах)
- Возможность отображать в ленте ELMA365
 - Информация по согласованию документа
 - Информация по ознакомлению с документом
- Маппинг базовых сущностей в ELMA3/4 и ELMA365
 - Мои юр. лица

ELMA3/4 ↔ ELMA365:

- Номенклатура
 - Место регистрации
 - Название
 - Раздел
 - Название
 - Дело
 - Название
 - Вид нумерации (Вречную или Автоматически)
 - Следующий номер
 - Шаг (всегда 1)
 - Максимальное количество знаков
 - Документопоток
 - Шаблон номера
 - {\$GroupIndex} удаляется при переносе
 - {\$RegCardNumber} заменяется на {\$__index}
- Регистрационная карточка документа
 - Регистрационный номер
 - Дата регистрации
- Мои юр. лица
 - Название
 - Полное наименование
 - Юридический адрес
 - Фактический адрес
 - ИНН
 - КПП

- Расчетный счет
- Корреспондентский счет
- Банк
- БИК
- Телефон

Ограничения

- Номенклатура
 - Ссылки на дело не поддерживаются (миграция прерывается)
 - Вложенность разделов не должна быть более одного уровня (миграция прерывается)
 - Дело должно иметь хотя бы один типа документа и документопотока (иначе исключается из миграции)
 - Дело не должно иметь разные документопотоки (миграция прерывается)
 - {\$GroupIndex} не поддерживается и удаляется при передаче
 - Документопоток Договора не поддерживается (миграция прерывается)
 - **TODO** Сквозная нумерация не поддерживается
- Регистрационная карточка документа
 - Автором регистрации является пользователь миграции

Контур Проекты

Функционал управления проектами

ITino.ELMA.E365.Projects:

-

Контур Управление договорами

Функционал управления проектами

ITino.ELMA.E365.BS.Contracts:

-

Точки расширения

IForceSyncHandler

Используйте наследование от ForceSyncHandler

```
/// <summary>
/// </summary>
[ExtensionPoint(ServiceScope.Shell)]
public interface IForceSyncHandler
{
    /// <summary>
    /// Включено
    /// </summary>
    bool Enabled { get; }

    /// <summary>
    /// Тип сущности
    /// </summary>
    Type Type { get; }

    /// <summary>
    /// Выполнить миграцию
    /// </summary>
    /// <param name="query">Дополнительные условия</param>
    /// <returns></returns>
    void Process(string query = null);
}
```

IDatalItemExtension

Используйте наследование от `DatalItemExtension`

```
/// <summary>
/// </summary>
[ExtensionPoint(ServiceScope.Shell)]
public interface IDatalItemExtension
{
    /// <summary>
    /// Проверка типа от IBaseDatalItem
    /// </summary>
    /// <param name="type">Тип</param>
    /// <returns></returns>
    bool CheckType(Type type);

    /// <summary>
    /// Получить кастомные простые свойства сущности
    /// </summary>
    /// <param name="item"></param>
    /// <param name="entity"></param>
    JObject GetCustomSimple(IBaseDatalItem item, IEntity entity);

    /// <summary>
    /// Получить связанные (справочники) кастомные свойства сущности
    /// </summary>
    /// <param name="item"></param>
    /// <param name="entity"></param>
    void GetCustomLinked(IBaseDatalItem item, IEntity entity);
}
```

Хелперы

ServerHelper

```
/// <summary>
/// Возвращает признак среды разработки
/// </summary>
public static bool IsDev
```

```
/// <summary>
/// Логгер интеграции
/// </summary>
public static ILog E365Logger { get; }
```

```
/// <summary>
/// Запустить процесс в ELMA365
/// </summary>
/// <param name="context">Контекст процесса монолита</param>
/// <param name="namespace">Пространство ELMA365</param>
/// <param name="code">Процесс ELMA365</param>
/// <param name="action">Модель контекста процесса ELMA365</param>
/// <param name="ack">Контроль выполнения в ELMA365</param>
/// <param name="async">Асинхронно</param>
/// <param name="throwException">Вызывать исключение</param>
/// <typeparam name="T">Класс контекста процесса</typeparam>
public static Guid E365StartProcess<T>(T context, string @namespace, string code,
Action<E365ProcessModel<T>> action, bool ack = false, bool async = false, bool throwException = false)
```

```
/// <summary>
/// Полная синхронизация справочника
/// </summary>
/// <param name="type">Тип сущности</param>
/// <returns></returns>
public static bool E365FullSync(Type type)
```

```
/// <summary>
/// Синхронизация системной информации (оргструктура и пользователи)
```



```
/// </summary>
```

```
/// <returns></returns>
```

```
public static bool E365SystemSync()
```

ListenerHelper

```
/// <summary>
/// Проверка на возможность мягкого удаления
/// </summary>
/// <param name="event">Событие</param>
/// <param name="action">Проверочное действие</param>
/// <typeparam name="T">Тип</typeparam>
public static void TrySoftDeleting<T>(PreUpdateEvent @event, Action<T> action) where T : class, IEntity
```

```
/// <summary>Получить старое значение</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Старое значение</returns>
public static T GetOldValue<T>(PostUpdateEvent @event, string propertyName)
```

```
/// <summary>Получить старое значение</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Старое значение</returns>
public static T GetOldValue<T>(PreUpdateEvent @event, string propertyName)
```

```
/// <summary>Получить значение свойства</summary>
/// <typeparam name="T">Тип</typeparam>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <returns>Значение</returns>
public static T GetValue<T>(PostUpdateEvent @event, string propertyName)
```

```
/// <summary>Присвоить значение свойству</summary>
/// <param name="event">Событие</param>
/// <param name="propertyName">Название свойства</param>
/// <param name="value">Значение</param>
```

```
public static void SetValue(PreUpdateEvent @event, string propertyName, object value)
```

```
/// <summary>Получить значение свойства</summary>  
/// <typeparam name="T">Тип</typeparam>  
/// <param name="event">Событие</param>  
/// <param name="propertyName">Название свойства</param>  
/// <returns>Значение</returns>  
public static T GetValue<T>(PostInsertEvent @event, string propertyName)
```

```
/// <summary>Присвоить значение свойству</summary>  
/// <param name="event">Событие</param>  
/// <param name="propertyName">Название свойства</param>  
/// <param name="value">Значение</param>  
public static void SetValue(PreInsertEvent @event, string propertyName, object value)
```

Менеджеры

E365ProcessLinkManager

```
/// <summary>
/// Запущен ли процесс в ELMA365
/// </summary>
/// <param name="uid"></param>
/// <returns></returns>
public bool IsRunning(Guid uid)
```

```
/// <summary>
/// Статус процесса в ELMA365
/// </summary>
/// <param name="uid"></param>
/// <returns></returns>
public WorkflowInstanceStatus GetStatus(Guid uid)
```

```
/// <summary>
/// Получить список запущенных процессов
/// </summary>
/// <returns></returns>
public IDictionary<Guid, Guid> GetActiveProcesses()
```

Примеры

Реализация миграции для справочника Страна

Для передачи данных сущности из ELMA3/4 в ELMA365 в глобальном модуле необходимо определить класс маппинга. Пример для справочника Страна (с дополнительными полями в конфигурации ELMA3/4). В ELMA365 создаем Приложение как предложено здесь:

Элемент обмена данными (маппинг):

```
using ITino.ELMA.E365.Common.Models;
using Newtonsoft.Json;

namespace E365
{
    public class CountryDataItem : BaseDataItem
    {
        public override string Path => "app/_clients/rmgCountry";

        [JsonProperty(PropertyName = "__name")]
        public string Name { get; set; }

        public string Code { get; set; }

        public string ShortName { get; set; }

        public long? CountryCode { get; set; }

        public string EnglishName { get; set; }

        public string Alpha2 { get; set; }

        public string Alpha3 { get; set; }

        public string Location { get; set; }
    }
}
```

```

        public string LocationPrecise { get; set; }
    }
}

```

Лисенер для регистрации изменений элемента сущности:

```

using EleWise.ELMA.Runtime.NH.Listeners;
using EleWise.ELMA.ComponentModel;
using NHibernate.Event;
using ITino.ELMA.E365.Common.Components;
using EleWise.ELMA.Model.Common;
using EleWise.ELMA.Model.Entities;
using ITino.ELMA.E365.Common.Managers;
using EleWise.ELMA;
using ITino.ELMA.CRM.Models;

namespace E365
{
    [Component]
    public class CountryListener : PostFlushEventListener
    {
        public override void OnPostInsert(PostInsertEvent @event)
        {
            SyncItem(@event.Entity as ICOCountry, true);
        }

        /// <inheritdoc />
        public override void OnPostUpdate(PostUpdateEvent @event)
        {
            SyncItem(@event.Entity as ICOCountry, true);
        }

        public static Pair<IBaseDataItem, IEntity> SyncItem(ICOCountry item, bool syncLink = false)
        {
            var data = new CountryDataItem();

            if (item == null)
                return new Pair<IBaseDataItem, IEntity>(data, item);
        }
    }
}

```



```

        data.Uid = item.Uid;
        data.Name = item.Name;
        data.Code = item.Code;
        data.ShortName = item.ShortName;
        data.CountryCode = item.CountryCode;
        data.EnglishName = item.EnglishName;
        data.Alpha2 = item.Alpha2;
        data.Alpha3 = item.Alpha3;
        data.Location = item.Location;
        data.LocationPrecise = item.LocationPrecise;

        E365DataItemManager.Instance.PushItem(typeof(CountryDataItem), item.Uid, data, item, SR.T("Страна:
{0}", item.Name));

        return new Pair<IBaseDataItem, IEntity>(data, item);
    }
}
}

```

Обработчик события полной синхронизации справочника. Так же используется для вызова принудительной синхронизации всех данных из хелпера:

```

using System;
using System.Linq;
using EleWise.ELMA.ComponentModel;
using EleWise.ELMA.Model.Services;
using ITino.ELMA.E365.Common.Components;
using EleWise.ELMA.Model.Managers;
using ITino.ELMA.CRM.Models;

namespace E365
{
    [Component]
    public class CountrySyncHandler : ForceSyncHandler
    {
        public override Type Type => InterfaceActivator.TypeOf<ICOCountry>();

        public override void Process()
        {
            EntityManager<ICOCountry>.Instance.FindAll().ToList().ForEach(x => CountryListener.SyncItem(x));
        }
    }
}

```

```
}  
}  
}
```

ICOCountry является расширением модульной сущности ICountry

Полная принудительная миграция данных сущности

Если необходимо сделать разово принудительную полную миграцию данных для сущности, у нее должен быть реализована точка расширения **IForceSyncHandler**. Самый простой способ - вызвать метод хелпера в сценарии процесса. При этом достаточно реализовать элемент Сценарий с со следующим кодом и запустить отладку процесса на данном элементе:

```
namespace EleWise.ELMA.Model.Scripts
{
    public partial class P_StartE365Process_Scripts : EleWise.ELMA.Workflow.Scripts.ProcessScriptBase<Context>
    {
        /// <summary>
        /// Запустить полную миграцию справочника Валюта
        /// </summary>
        /// <param name="context">Контекст процесса</param>
        public virtual void FullSync (Context context)
        {
            ServerHelper.E365FullSync(InterfaceActivator.TypeOf<ICurrency>());
        }
    }
}
```

В данном примере используется справочник Валюта. Если используется компонент ITino.ELMA.E365.CRM, то точка расширения **IForceSyncHandler** уже присутствует. Иначе ее нужно создать самостоятельно.

Свойства сущности типа IEntity

При реализации маппинга простых типов данных не требуется каких либо ухищрений. Однако, если в сущности есть свойства типа ссылки на справочник или документ, то необходимо реализовать дополнительную логику.

Пример для сущности Контакт (исходный код урезан для простоты понимания):

```
/// <inheritdoc />
public class ContactDataItem : BaseDataItem
{
    /// <inheritdoc />
    public override string Path => "app/_clients/_contacts";

    /// <summary>
    /// </summary>
    [JsonProperty(PropertyName = "__name", NullValueHandling = NullValueHandling.Ignore)]
    public string Name { get; set; }

    ...

    /// <summary>
    /// </summary>
    [JsonProperty(PropertyName = "_companies", NullValueHandling = NullValueHandling.Ignore)]
    public List<Guid> Companies { get; set; }

    ...
}
```

Данная реализация позволяет подготовить пакет передачи в ELMA365 в нужном формате.

Лисенер для регистрации изменений элемента сущности (исходный код урезан для простоты понимания):

```

public static Pair<IBaseDataItem, IEntity> SyncContact(IContact item, bool syncLink = false)
{
    var data = new ContactDataItem();
    if (item == null)
        return new Pair<IBaseDataItem, IEntity>(data, item);

    ...

    E365DataItemManager.Instance.PushItem(typeof(ContactDataItem), item.Uid, data, item, SR.T("Контакт:
{0}", item.Name));

    if (!syncLink)
        return new Pair<IBaseDataItem, IEntity>(data, item);

    var link = new LinkDataItem(data);

    link.Links.Add(new LinkImpl(ContractorListener.SyncContractor(item.Contractor),
        BaseDataItem.GetJsonPropertyName<ContactDataItem>(x => x.Companies)));

    E365DataItemManager.Instance.PushItem(typeof(LinkDataItem), link.Uid, link, item, SR.T("Связи в контакте:
{0}", data.Name));

    return new Pair<IBaseDataItem, IEntity>(data, item);
}

```

Свойства сущности типа BinaryFile

Если в сущности есть свойства типа BinaryFile, то необходимо реализовать дополнительную логику.

Лисенер для регистрации изменений элемента сущности (исходный код урезан для простоты понимания):

```
public static Pair<IBaseDataItem, IEntity> SyncContact(IContact item, bool syncLink = false)
{
    var data = new ContactDataItem();
    if (item == null)
        return new Pair<IBaseDataItem, IEntity>(data, item);

    ...

    E365DataItemManager.Instance.PushItem(typeof(ContactDataItem), item.Uid, data, item, SR.T("Контакт:
{0}", item.Name));

    if (!syncLink)
        return new Pair<IBaseDataItem, IEntity>(data, item);

    var link = new LinkDataItem(data);

    // vCard для Контакта имеем тип BinaryFile
    // contact_vCard - название свойтва в Приложении ELMA365 типа Файл

    link.Links.Add(new LinkImpl(FileDataItem.Create(item.vCard), "contact_vCard"));

    E365DataItemManager.Instance.PushItem(typeof(LinkDataItem), link.Uid, link, item, SR.T("Связи в контакте:
{0}", data.Name));

    return new Pair<IBaseDataItem, IEntity>(data, item);
}
```


Реализация миграции для документа Счет исходящий

Для передачи данных сущности из ELMA3/4 в ELMA365 в глобальном модуле необходимо определить класс маппинга. Пример для справочника Страна (с дополнительными полями в конфигурации ELMA3/4). В ELMA365 создаем Приложение (документ) как предложено здесь:

Элемент обмена данными (маппинг):

```
using System;
using ITino.ELMA.E365.Common.Models;
using ITino.ELMA.Accounting.Documents.Models;

namespace E365
{
    public class OutInvoiceDataItem : DocumentDataItem
    {
        public override string Path => "app/_clients/rmgOutInvoice";

        public string Number { get; set; }

        public DateTime Date { get; set; }

        public OutInvoiceDataItem(DOOutInvoice document) : base(document)
        {
        }
    }
}
```

Лисенер для регистрации изменений элемента сущности:


```

using EleWise.ELMA.ComponentModel;
using EleWise.ELMA.Runtime.NH.Listeners;
using NHibernate.Event;
using ITino.ELMA.Accounting.Documents.Models;
using EleWise.ELMA.Model.Common;
using ITino.ELMA.E365.Common.Components;
using EleWise.ELMA.Model.Entities;
using ITino.ELMA.E365.Common.Managers;
using EleWise.ELMA;

namespace E365
{
    [Component]
    public class OutInvoiceListener : PostFlushEventListener
    {
        public override void OnPostInsert(PostInsertEvent @event)
        {
            SyncItem(@event.Entity as DOOutInvoice, true);
        }

        /// <inheritdoc />
        public override void OnPostUpdate(PostUpdateEvent @event)
        {
            SyncItem(@event.Entity as DOOutInvoice, true);
        }

        public static Pair<IBaseDataItem, IEntity> SyncItem(DOOutInvoice item, bool syncLink = false)
        {
            var data = new OutInvoiceDataItem(item);

            if (item == null)
                return new Pair<IBaseDataItem, IEntity>(data, item);

            data.Number = item.IEEDocNumber;
            data.Date = item.IEEDocDate;

            E365DataItemManager.Instance.PushItem(typeof(OutInvoiceDataItem), item.Uid, data, item, SR.T("Счет
исходящий: {0}", item.Name));

            data.PushVersion();
        }
    }
}

```

```

        return new Pair<IBaseDataItem, IEntity>(data, item);
    }
}
}

```

Обработчик события полной синхронизации справочника. Так же используется для вызова принудительной синхронизации всех данных из хелпера:

```

using System;
using System.Linq;
using EleWise.ELMA.ComponentModel;
using EleWise.ELMA.Model.Services;
using ITino.ELMA.E365.Common.Components;
using EleWise.ELMA.Model.Managers;
using ITino.ELMA.CRM.Models;

namespace E365
{
    [Component]
    public class CountrySyncHandler : IForceSyncHandler
    {
        public Type Type => InterfaceActivator.TypeOf<ICOCountry>();

        public bool Enabled => true;

        public void Process()
        {
            EntityManager<ICOCountry>.Instance.FindAll().ToList().ForEach(x => CountryListener.SyncItem(x));
        }
    }
}

```

Блоки в сущности

Если в сущности есть блоки, то их можно мигрировать в ELMA365. Для этого нужно реализовать следующий код в точках расширения:

Для реализации миграции блока необходимо определить его в коде, например:

```
using System;
using System.Collections.Generic;
using ITino.ELMA.E365.Common.Models;
using Newtonsoft.Json;

namespace E365
{
    public class TestTablePartsDataItem : BaseDataItem
    {
        public override string Path => "app/_clients/testtableparts";

        [JsonProperty(PropertyName = "__name")]
        public string Name { get; set; }

        [JsonIgnore]
        public TableDataItem TestTable { get; set; }

        public TestTablePartsDataItem()
        {
            TestTable = new TableDataItem("testtable", this);
        }
    }
}
```

Структура блока и его записи заполняются для передачи, например (при этом указываются простые свойства и справочники):

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using EleWise.ELMA.API;
using System.Text;
using EleWise.ELMA.ComponentModel;
using EleWise.ELMA.Runtime.NH.Listeners;
using NHibernate.Event;
using EleWise.ELMA.ConfigurationModel;
using EleWise.ELMA.Model.Common;
using ITino.ELMA.E365.Common.Components;
using EleWise.ELMA.Model.Entities;
using EleWise.ELMA.Extensions;
using ITino.ELMA.E365.Common.Managers;
using EleWise.ELMA;
using ITino.ELMA.E365.Common.Models;
using ITino.ELMA.E365.CRM.Listeners;
using ITino.ELMA.E365.Common.Listeners;

namespace E365
{
    [Component]
    public class TestTableParts_TestTableListener : PostFlushEventListener
    {
        public override void OnPostInsert(PostInsertEvent @event)
        {
            SyncItem(@event.Entity as ITestTableParts_TestTable, true);
        }

        public override void OnPostUpdate(PostUpdateEvent @event)
        {
            SyncItem(@event.Entity as ITestTableParts_TestTable, true);
        }

        public override void OnPostDelete(PostDeleteEvent @event)
        {
            SyncItem(@event.Entity as ITestTableParts_TestTable, true);
        }

        public static Pair<IBaseDataItem, IEntity> SyncItem(ITestTableParts_TestTable item, bool syncLink = false)
        {
            return TestTablePartsListener.SyncItem(item?.Parent);
        }
    }
}

```

[Component]

```
public class TestTablePartsListener : PostFlushEventListener
{
    public override void OnPostInsert(PostInsertEvent @event)
    {
        SyncItem(@event.Entity as ITestTableParts, true);
    }

    public override void OnPostUpdate(PostUpdateEvent @event)
    {
        SyncItem(@event.Entity as ITestTableParts, true);
    }

    public static Pair<IBaseDataItem, IEntity> SyncItem(ITestTableParts item, bool syncLink = false)
    {
        var data = new TestTablePartsDataItem();

        if (item == null)
            return new Pair<IBaseDataItem, IEntity>(data, item);

        data.Uid = item.Uid;
        data.Name = item.Name;

        item.TestTable?.ForEach(tableItem => {
            var row = new RowDataItem();
            row.SimpleContext.Add ("boolean", tableItem.Boolean);
            row.SimpleContext.Add ("string", tableItem.String);
            row.LinkedContext.Add ("currency", new List<LinkImpl> {
                new LinkImpl (CurrencyListener.SyncItem(tableItem.Currency))
            });
            row.LinkedContext.Add ("newuser", new List<LinkImpl> {
                new LinkImpl (UserListener.SyncItem(tableItem.NewUser))
            });
            row.SimpleContext.Add ("newnumber", tableItem.NewNumber);
            row.SimpleContext.Add ("date", tableItem.Data);
            data.TestTable.Rows.Add(row);
        });

        E365DataItemManager.Instance.PushItem(typeof(TestTablePartsDataItem), item.Uid, data, item,
        SR.T("TestTableParts: {0}", item.Name));
```

```
        return new Pair<IBaseDataItem, IEntity>(data, item);  
    }  
}  
}
```