

# ???????

- [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](#)

?????????? ?????????? ???  
???????????? ????????

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

?????? ??????????????  
???????? ?????????????

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.

????????? ?????????? ????

# 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);
}
```



????????? ?????????? ????

# 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 свойства 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);
}
```

???????????? ???? ????  
???????????? ???? ?????????

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(D00OutInvoice 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 D00OutInvoice, true);
        }

        /// <inheritdoc />
        public override void OnPostUpdate(PostUpdateEvent @event)
        {
            SyncItem(@event.Entity as D00OutInvoice, true);
        }

        public static Pair<IBaseDataItem, IEntity> SyncItem(D00OutInvoice 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));
        }
    }
}
```

# ????? ? ?????????

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);
    }
}
```