Утилита переноса настроек Directum RX. Инструкция по использованию

Утилита переноса настроек Directum RX предназначена для переноса данных справочников из одной системы Directum RX в другую. Например, из тестовой в продуктивную.

Утилита включает в себя готовые правила для переноса настроек из следующих справочников системы:

- Правила и этапы согласования;
- Категории договоров;
- Виды документов;
- Настройки регистрации;
- Журналы регистрации;
- Роли.

В поставку входит папка с утилитой и документация.

Системные требования

Поддерживается перенос настроек между системами Directum RX 3.4 и выше.

На компьютере, на котором запускается утилита, должна быть установлена одна из версий ОС:

- Microsoft Windows 8 и выше;
- Microsoft Windows Server 2012 и выше.

Для модификации правил переноса необходимо установить:

- среду разработки, поддерживающую язык программирования C#. Например, Microsoft Visual Studio Community или SharpDevelop;
- Microsoft.Net Framework 4.6.1 и выше.

Использование утилиты

Утилита представляет собой консольное приложение, которое передает данные через файл в формате JSON. Для экспорта или импорта данных нужно запустить утилиту в командной строке с необходимыми параметрами.

Чтобы перенести настройки из одной системы в другую:

1. Экспортируйте данные из системы-источника.
2. Импортируйте данные в систему-приемник.

Экспорт данных

1. Из полученного дистрибутива скопируйте на компьютер папку с утилитой DrxTransfer.exe.
2. В папку с утилитой из папки DrxUtil системы-источника скопируйте конфигурационный файл _ConfigSettings.xml. Проверьте, что путь до сервера приложений (**SERVER_ROOT_HTTP** и **SERVER_ROOT_HTTPS**) совпадают с соответствующими значениями, указанными при установке системы, из которой нужно перенести данные. Утилита использует этот файл для подключения к нужной базе данных.
3. Скопируйте интерфейсную сборку сервера приложений системы-источника и положите её в папку lib проекта TranferSerializers. При этом:
   - если система установлена локально, то путь до сборки по умолчанию следующий: C:\inetpub\wwwroot\DirectumRx\bin\AppliedModules\Sungero.Domain.Interfaces.dll.
   - если система установлена в частном облаке, запросите интерфейсную сборку в службе поддержки Directum RX.
4. В командной строке запустите исполняемый файл DrxTransfer.exe с указанными ниже параметрами. Настройки системы-источника выгрузятся в заданный файл для последующего импорта в систему-приемник.

ВАЖНО. За один запуск утилиты можно выгрузить только один тип настроек. Например, только справочник **Роли**.

Параметры экспорта

-n – имя пользователя, от имени которого выполняется подключение к системе Directum RX. Рекомендуется использовать системного пользователя **Integration Service**.

-p – пароль пользователя, от имени которого выполняется подключение к системе Directum RX.

-x <Путь к файлу с данными> – путь к файлу, в который выгружаются данные.

-j – тип выгружаемого объекта. Возможные значения:

- **ApprovalRule** – правила и этапы согласования;
- **ContractCategory** – категории договоров;
- **DocumentKind** – виды документов;
- **DocumentRegister** – журналы регистрации;
- **RegistrationSetting** – настройки регистрации;
- **Role** – роли.

ПРИМЕЧАНИЕ. В ключе **-j** можно указать только одно значение.

Пример:

```
DrxTransfer.exe -n administrator -p 11111 -x “C:\Roles.txt” -j Role
```

Импорт данных

1. Подготовьте утилиту к импорту.
2. Запустите импорт.

Подготовка утилиты

Способ импорта экспортированных данных зависит от условий размещения утилиты относительно системы-источника и системы-приемника:

- утилита находится в сети, из которой есть доступ к обеим системам;
- утилита находится в сети, из которой есть доступ только к системе-приемнику.

Утилита имеет доступ к обеим системам

В папке с утилитой, которая использовалась для экспорта, в конфигурационном файле _ConfigSettings.xml измените значения параметров **SERVER_ROOT_HTTP** и **SERVER_ROOT_HTTPS** на соответствующие пути до сервера приложений системы-приемника. Утилита использует этот файл для подключения к нужной базе данных.

Утилита имеет доступ только к системе-приемнику

1. Из полученного дистрибутива скопируйте на компьютер, на котором расположена система-приемник, папку с утилитой DrxTransfer.exe.
2. В папку с утилитой из папки DrxUtil системы-приемника скопируйте конфигурационный файл _ConfigSettings.xml. Проверьте, что путь до сервера приложений (**SERVER_ROOT_HTTP** и **SERVER_ROOT_HTTPS**) совпадают с соответствующими значениями, указанными при установке системы, в которую нужно перенести данные. Утилита использует этот файл для подключения к нужной базе данных.
3. Скопируйте интерфейсную сборку сервера приложений системы-приемника и положите её в папку lib проекта TranferSerializers. При этом:
   - если система установлена локально, то путь до сборки по умолчанию следующий: C:\inetpub\wwwroot\DirectumRx\bin\AppliedModules\Sungero.Domain.Interfaces.dll.
   - если система установлена в частном облаке, запросите интерфейсную сборку в службе поддержки Directum RX.

Запуск импорта

После подготовки утилиты к работе в командной строке запустите файл DrxTransfer.exe с указанными ниже параметрами. Настройки импортируются из указанного файла системы-источника в систему-приемник.

ВАЖНО. Импорт данных – ресурсоемкая операция, поэтому рекомендуется выполнять ее в нерабочее время.

Параметры импорта

-n – имя пользователя, от имени которого выполняется подключение к системе Directum RX. Рекомендуется использовать системного пользователя **Integration Service**.

-p – пароль пользователя, от имени которого выполняется подключение к системе Directum RX.

-i <Путь к файлу с данными> – путь к файлу с данными для импорта.

Пример:

```
DrxTransfer.exe -n administrator -p 11111 -i “C:\Roles.txt”
```

Модификация утилиты

Утилита переноса настроек состоит из проектов:

- DrxTransfer. Представляет собой ядро утилиты и его исходный код. Содержит базовое правило **SungeroSerializer** и логику экспорта-импорта данных;
- TransferSerializers. Содержит правила переноса настроек конкретного типа справочника и их исходными кодами.

  Модификация утилиты осуществляется путем перекрытия методов и типов сущностей базового правила в рамках проекта TransferSerializers.

ВАЖНО. Для корректной работы, правила необходимо помечать атрибутом **[Export(typeof(SungeroSerializer))]**.

Правила загружаются в процессе выполнения утилиты с помощью технологии Managed Extensibility Framework (MEF).

Перекрываемые методы класса **SungeroSerializer**:

- **Export()**. Используется для экспорта коллекции ключей и значений в JSON файл.
- **Import()**. Используется для получения данных из JSON-файла, создания, заполнения данных и сохранения сущности в Directum RX.
- **Filter()**. Используется для фильтрации сущности по определенному признаку, например по состоянию.

Утилиту можно модифицировать для:

1. Разработки правил переноса данных, которые не включены в стандартную поставку.
2. Доработки правил из комплекта поставки после изменений в прикладной части системы. В этом случае предварительно обновите интерфейсную сборку.

ВАЖНО. Перед любыми модификациями утилиты ознакомьтесь с ограничением и добавьте ссылки на библиотеки в проекты утилиты.

Ограничение

Правила рассчитаны на перенос данных справочников. Перенос других объектов Directum RX, например документов или задач, не поддерживается.

Обновление интерфейсной сборки

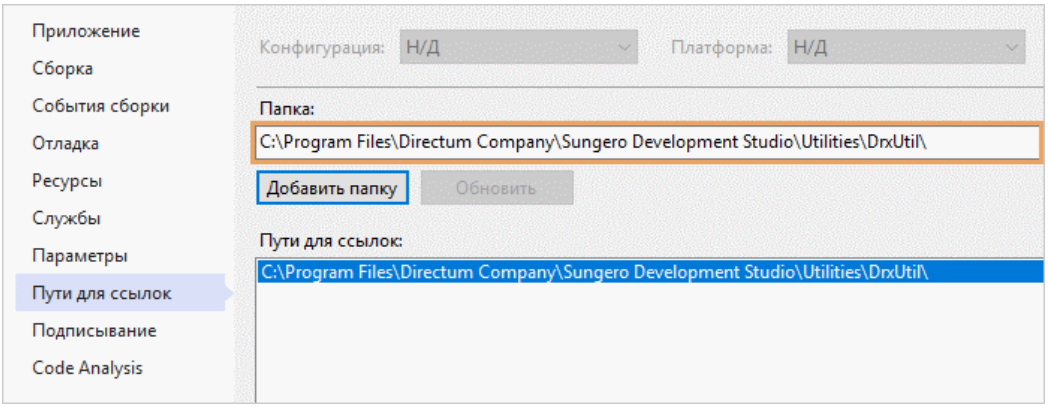
Первичная настройка интерфейсной сборки выполняется на этапе экспорта-импорта базовых правил переноса. Ее необходимо обновить вручную, если параллельно модификации правил переноса проводилась модификация системы Directum RX и после публикации пакета разработки изменились интерфейсные сборки – библиотеки Sungero.Domain.Interfaces.dll. Такая ситуация может сложиться, когда:

- добавлена или удалена сущность;
- добавлена или удалена функция;
- изменена сигнатура функции.

Добавление библиотек в проекты утилиты

После копирования утилиты нужно добавить ссылки на библиотеки в проекты DrxTransfer и TransferSerializers. Для этого для каждого проекта:

1. Выделите проект и нажмите сочетание клавиш Alt+Enter. Откроется окно со свойствами проекта.
2. На закладке «Пути для ссылок» в поле **Папка** добавьте путь до папки с утилитой DrxUtil и нажмите на кнопку **Добавить папку**. Путь до папки с утилитой по умолчанию: C:\Program Files\Directum Company\Sungero Development Studio\Utilities\DrxUtil\.



Разработка новых правил переноса данных

Задача

Перенести данные справочника «Политики хранения документов» из одной системы Directum RX в другую.

Решение

Создайте новое правило переноса на основе базового правила **SungeroSerializer**. Для этого:

1. Создайте путем наследования от **SungeroSerializer** новое правило. Убедитесь, что указан атрибут класса, который необходим для загрузки и вызова правила утилитой:

```
[Export(typeof(SungeroSerializer))]
class StoragePolicySerializer : SungeroSerializer { }
```

2. Заполните конструктор класса, указав наименование правила и тип передаваемого объекта:

```
public StoragePolicySerializer() : base()
{
    // Имя сериализатора, которое используется в параметрах вызова утилиты.
    this.EntityName = "StoragePolicy";
    // Тип передаваемого объекта.
    this.EntityTypeName = "Sungero.Docflow.IStoragePolicy";
}
```

3. Переопределите метод **Filter()**, чтобы при переносе не учитывались закрытые записи политик хранения:

```
public override IEnumerable<IEntity> Filter(IEnumerable<IEntity> entities)
{
return entities.Cast<Sungero.Docflow.IStoragePolicy>().Where(c => c.Status
== Sungero.CoreEntities.DatabookEntry.Status.Active);
}
```

4. Переопределите метод **Export()**. Для выполнения текущей задачи необходимо записать информацию о ссылке на хранилище и свойстве коллекции для видов документов:

```
protected override Dictionary<string, object> Export(IEntity entity)
{
    // Вызываем базовый метод для обработки свойств с простым типом.
    base.Export(entity);
    // Преобразуем переданный объект в необходимый тип.
    var storagePolicy = (entity as Sungero.Docflow.IStoragePolicy);
    // Запишем информацию о ссылке на хранилище в словарь.
    content["Storage"] = storagePolicy.Storage;
    // Запишем информацию о видах документов из табличной части в словарь.
    content["DocumentKinds"] = storagePolicy.DocumentKinds.Select(d
=>d.DocumentKind);
    return content;
}
```

5. Переопределите метод **Import()**. Для этого последовательно:

- преобразуйте словарь Dictionary в JObject с указанием предопределенного элемента «Card»:

```
var entityItem = content["Card"] as JObject;
```

- получите наименование создаваемого объекта и проверьте, чтобы записи с таким же наименованием не дублировались:

```
var storagePolicyName = entityItem.Property("Name").Value.ToString();
var activeStoragePolicy =
Session.Current.GetEntities(this.EntityTypeName)
.Cast<Sungero.Docflow.IStoragePolicy>().Where(p => p.Name ==
storagePolicyName
&& p.Status ==
Sungero.CoreEntities.DatabookEntry.Status.Active).FirstOrDefault();
if (activeStoragePolicy != null)
     throw new System.IO.InvalidDataException(string.Format("Политика
хранения {0} уже существует", storagePolicyName));
```

- добавьте ссылку на хранилище для дальнейшего заполнения через наименование. Если хранилище не найдено, то остановите обработку данной записи:

```
var storageItem = content["Storage"] as JObject;
var storageName = storageItem.Property("Name").Value.ToString();
var storage =
Session.Current.GetEntities("Sungero.CoreEntities.IStorage")
.Cast<Sungero.CoreEntities.IStorage>().FirstOrDefault(s => s.Name ==
storageName);
if (storage == null)
throw new System.IO.InvalidDataException(string.Format("Хранилища {0} не
существует", storageName));
```

- создайте новую запись политики хранения и запишите информацию в лог-файл и консоль:

```
var storagePolicy = Session.Current.CreateEntity(this.EntityTypeName) as
Sungero.Docflow.IStoragePolicy;
Log.Console.Info(string.Format("ИД = {0}. Создание политики хранения
{1}", storagePolicy.Id, storagePolicyName));
```

- заполните реквизиты данными из JSON-файла:

```
storagePolicy.Name = storagePolicyName;
storagePolicy.Priority =
entityItem.Property("Priority").ToObject<int?>();
storagePolicy.Storage = storage;
```

- добавьте заполнение табличной части видов документов через метод **GetEntities()** класса SungeroRepository, который позволяет получить данные JSON-файла в виде списка сущностей. Поиск происходит по наименованию записей:

```
var documentKinds =
SungeroRepository.GetEntities<Sungero.Docflow.IDocumentKind>(content,
"DocumentKinds", true, true);
foreach (var documentKind in documentKinds)
{
var documentKindItem = storagePolicy.DocumentKinds.AddNew();
documentKindItem.DocumentKind = documentKind;
}
```

- сохраните запись:

```
storagePolicy.Save();
Session.Current.SubmitChanges();
```

Итоговый код метода будет выглядеть следующим образом:

```
public override void Import(Dictionary<string, object> content)
{
  var entityItem = content["Card"] as JObject;
  var storagePolicyName = entityItem.Property("Name").Value.ToString();
  var activeStoragePolicy =

Session.Current.GetEntities(this.EntityTypeName).Cast<Sungero.Docflow.IStora
gePolicy>()
    .Where(p => p.Name == storagePolicyName && p.Status ==
Sungero.CoreEntities.DatabookEntry.Status.Active).FirstOrDefault();
  if (activeStoragePolicy != null)
```

```
  {
    throw new System.IO.InvalidDataException(string.Format("Политика
хранения {0} уже существует", storagePolicyName));
  }
  var storageItem = content["Storage"] as JObject;
  var storageName = storageItem.Property("Name").Value.ToString();
  var storage =

Session.Current.GetEntities("Sungero.CoreEntities.IStorage").Cast<Sungero.Co
reEntities.IStorage>()
    .FirstOrDefault(s => s.Name == storageName);
  if (storage == null)
    throw new System.IO.InvalidDataException(string.Format("Хранилища {0} не
существует", storageName));
  var storagePolicy = Session.Current.CreateEntity(this.EntityTypeName) as
  Sungero.Docflow.IStoragePolicy;
  Log.Console.Info(string.Format("ИД = {0}. Создание политики хранения {1}",
storagePolicy.Id, storagePolicyName));
  storagePolicy.Name = storagePolicyName;
  storagePolicy.Priority = entityItem.Property("Priority").ToObject<int?>();
  storagePolicy.Storage = storage;
  Log.Console.Info("Заполнение видов документов");
  var documentKinds =
SungeroRepository.GetEntities<Sungero.Docflow.IDocumentKind>(content,
"DocumentKinds", true, true);
  foreach (var documentKind in documentKinds)
  {
  var documentKindItem = storagePolicy.DocumentKinds.AddNew();
  documentKindItem.DocumentKind = documentKind;
  }
  storagePolicy.Note = entityItem.Property("Note").ToObject<string>();
  storagePolicy.Save();
    Session.Current.SubmitChanges();
  }
```

Доработка правил из комплекта поставки

Задача

Допустим, в перекрытии справочника правил согласования **ContractsApprovalRule** добавлено новое свойство **ContractFunctionality** – функциональность договора с типом перечисление. Также в перекрытии справочника условий **ContractCondition** добавлено свойство коллекция **Regions** – возможные регионы контрагентов с ссылками на справочник «Region». Нужно доработать правило переноса правила согласования с учетом внесенных в него модификаций.

Решение

Перед началом доработки убедитесь, что обновлена интерфейсная сборка. После чего дополните базовое правило для переноса записей справочника правил согласования:

1. В правиле **ApprovalRuleSerializer** измените метод **Export()**. Для этого последовательно:
   - в секции заполнения значения **Card** измените тип справочника на тип перекрытия так, чтобы перечисление **ContractFunctionality** попадало в JSON-файл:

     ```
     if (isApprovalRule)
     content["Card"] = Sungero.Docflow.ApprovalRules.As(entity);
     else
     content["Card"] = DirRX.Solution .ContractApprovalRoles.As(entity);
     ```

   - в секции обработки условий согласования добавьте условие для обработки коллекции регионов:

     ```
     if (isApprovalRule)
        conditionContent["Addressees"] =
     Sungero.Docflow.Conditions.As(condition.Condition).Addressees.Select(a
     => a.Addressee);
     else
        conditionContent["Regions"] = DirRX.Solution
     .ContractConditions.As(condition.Condition).Regions.Select(r =>
     r.Region);
     ```

2. Измените метод **Import()** для заполнения новых свойств. Для этого последовательно:
   - в секции создания нового правила измените тип создаваемого справочника на тип перекрытия:

     ```
     if (isContractsRule)
        rule =
     Session.Current.CreateEntity("DirRX.Solution.IContractsApprovalRule") as
     DirRX.Solution.IContractsApprovalRule;
     else
        rule = Session.Current.CreateEntity("Sungero.Docflow.IApprovalRule")
     as Sungero.Docflow.IApprovalRule;
     ```

   - в секции заполнения параметров правила добавьте заполнения свойства **ContractFunctionality**:

     ```
     if (isContractsRule)
       {
         foreach (var documentGroup in documentGroups)
          {
            var documentGroupItem = rule.DocumentGroups.AddNew();
            documentGroupItem.DocumentGroup = documentGroup;
          }
         var contractFunctionality =
     entityItem.Property("ContractFunctionality").Value.ToString();
         DirRX.Solution.ContractsApprovalRules.As(rule).ContractFunctionality
     = Sungero.Core.Enumeration.GetItems
     (typeof(DirRX.Solution.ContractsApprovalRule.ContractFunctionality)).Fir
     stOrDefault(e => e.Value == contractFunctionality);
       }
     ```

   - в секции создания условий добавьте заполнение коллекции регионов:

```
if (!isContractsRule)
 {
  Log.Console.Info("Проверка наличия регионов");
  var customConditionEntity =
DirRX.Solution.ContractConditions.As(conditionEntity);
  var regionsEntity =
SungeroRepository.GetEntities<Sungero.Commons.IRegion>(conditionObject,
"Regions", true, true);
  foreach (var region in regionsEntity)
   {
    var regionItem = customConditionEntity.Regions.AddNew();
    regionItem.Region = region;
   }
  }
```