Creatio Academy

Sync Engine synchronization mechanism

Version 8.0

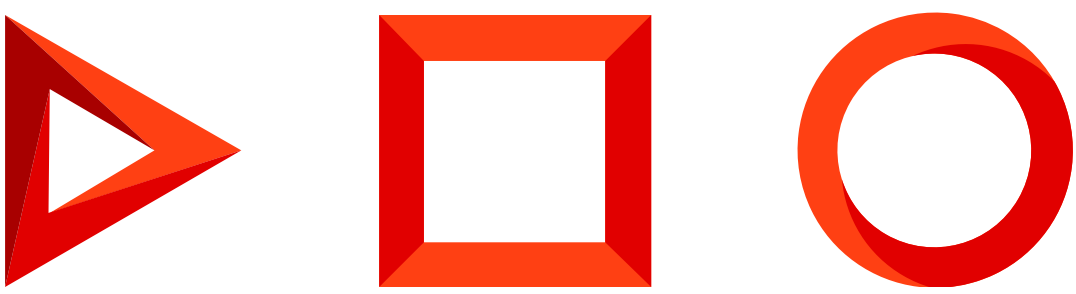


Table of Contents

Sync Engine synchronization mechanism basics

Advanced

Synchronization with external storages

The mechanism in Creatio for synchronization with external data storages is the Sync Engine. This mechanism enables you to create, modify, and delete `Entity` in the system based on synchronization with external systems and export data to external systems.

Synchronization is performed by using the `SyncAgent` class implemented in the `Terrasoft.Sync` namespace of the application kernel.

Classes used in the synchronization mechanism

- Synchronization agent (`SyncAgent`) — a class with one public `Synchronize` method that triggers synchronization between storages.
- Synchronization context (`SyncContext`) — a class representing the aggregation of providers and metadata for `SyncAgent` .
- Storage — storage of synchronized data.
  - Local storage (`LocalProvider`) — enables you to work with `LocalItem` in Creatio.
  - External storage (`RemoteProvider`) — an external service or application from which data is synchronized with Creatio.
- Synchronization item (`SyncItem`) — a set of objects from external and local storage which are synchronized.
  - External storage synchronization item (`RemoteItem`) — represents a set of data from external storage that syncs automatically. It can consist of one or more entities (records) from the external storage.
  - Synchronization item (`SyncEntity`) — a wrapper of a specific `Entity` . `SyncEntity` is required to work with `Entity` as the synchronizing object (add, delete, change).
  - Synchronization item (`LocalItem`) — one or more objects from Creatio that are synchronized with the external storage as a unit. The synchronization item from the external storage, converted in the `LocalItem` entity contains a set of instances of the `SyncEntity` class.
- `SysSyncMetadata` metadata table — contains service information of the synchronized elements and is essentially `RemoteItem` - `LocalItem` interchanges table.

General synchronization algorythm

Before starting synchronization, you must create an instance of `SyncAgent` and the `SyncContext` sync context, then update records in the metadata table with data from Creatio. For this, you need to call the

`CollectChangesInSyncedEntities` class method that implements the `IReplicaMetadata` interface.

The algorithm for updating metadata records is the following:

1. If any previously synchronized entity in Creatio has been modified since the last synchronization, then the corresponding record in the metadata changes its modification date, the `LocalState` property is set to “Modified”, and the source of the modification is set to the `LocalStore` ID.
2. If a synchronized entity in Creatio has been deleted since the last synchronization — the corresponding record in the metadata `LocalState` is set to “Deleted".
3. If there is no corresponding record in the metadata for the entity in Creatio — it is ignored.

The process of synchronizing storages then starts the following:

1. All changes from the external storage are requested alternately.
2. You need to obtain the metadata for each item in the external storage .
    a. If the metadata can not be obtained — this is a new item which should be converted to a Creatio element. To fill in a synchronization object, a `FillLocalItem` method is called from the specific `RemoteItem` instance. It is also recorded in the metadata (`Id` of the external storage, the element `Id` in the external storage, date of creation and modification is set as current, the source of creation and modification — external storage).
    b. If the metadata is received, so this item has already been synchronized with Creatio. You need to go to the version conflict resolution. By default, the last change in the application or external storage (`RemoteProvider`) has the priority.
    c. The metadata for the current pair of synchronization items is actualized.

After looking through all the modified items from the external storage, the elements that were changed in Creatio, but was not changed in the external storage remain in the metadata (in the interval between the last and the current synchronization).

1. You need to get elements changed in Creatio in the interval between the last synchronization and the current synchronization.
2. Save the changes in the external storage.
3. You must update the modification date of the items in the metadata (Creatio is the change source).

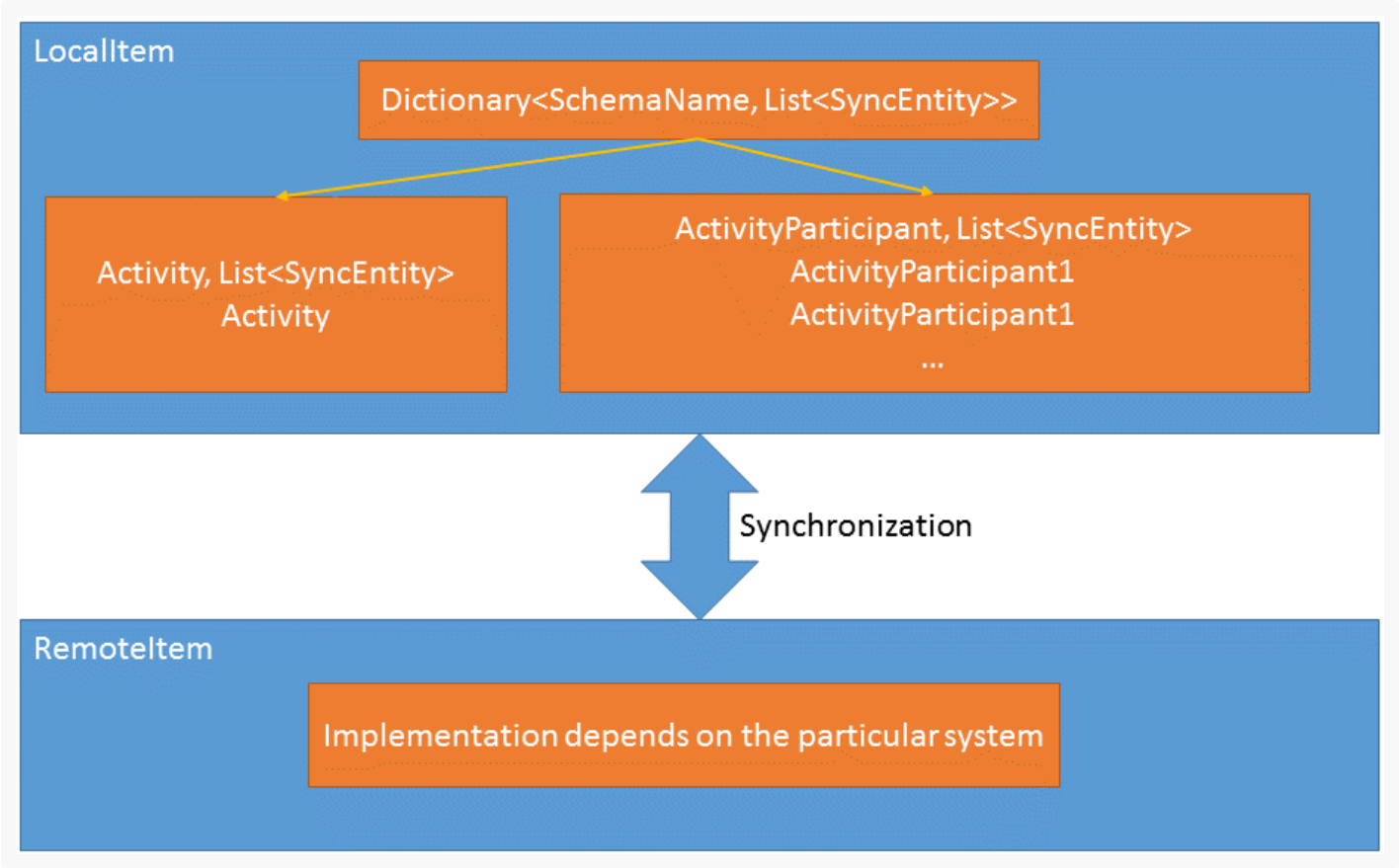
After that, you need to add new, not synchronized records to the external storage, and add metadata for new items.

Synchronization description

An activity and participants are synchronized into one Google-calendar task. An activity (`Activity`) and participants (`SyncEntity`) are one element of the synchronization - `LocalItem`.

`RemoteItem` - Google task received outside Creatio. `LocalItem` - a set of objects (`SyncEntity`), to which the Google task is converted.

The synchronization schema:



Working with synchronization metadata

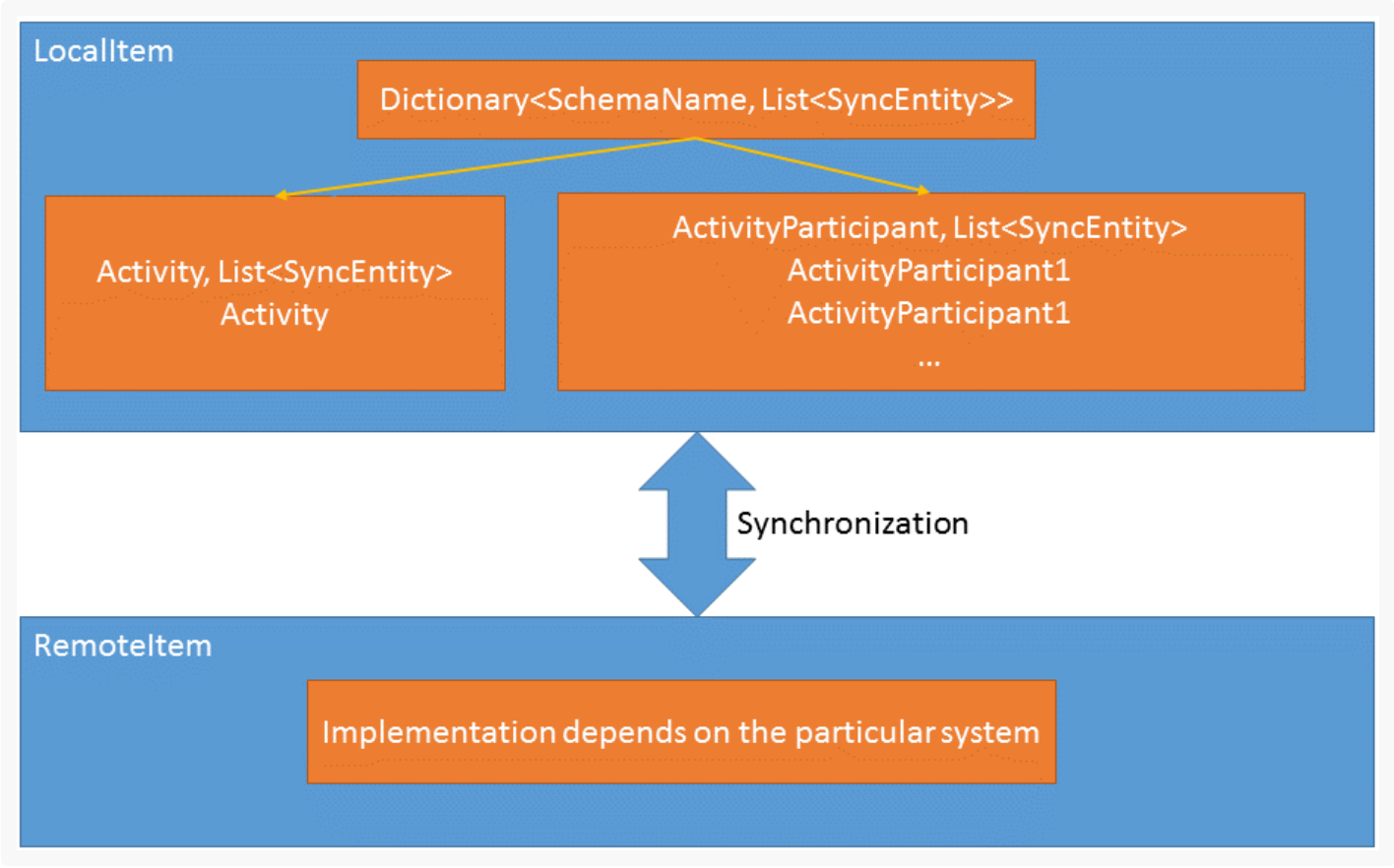
The auxiliary `SysSyncMetaDat` a metadata table is used for synchronization, which is the junction between the outer `RemoteItem` table (synchronizing element in external storage) and `LocalItem` (synchronization element in Creatio). Each table row is represented in the system as an instance of `SysSyncMetaData` class.

Only information about synchronized elements is stored in the metadata.

There can be multiple metadata table records for a single synchronization element - one for each application object included in a synchronization element.

Activity and participants — a single synchronization element. However, the metadata contains one record for each activity and one record for each participant.

Currently, only one object from external storage is transformed into several Creatio objects:



The metadata system for a single synchronization element is represented as the `ItemMetadata` object class (`SysSyncMetaData` element collection). Metadata management is carried out through the class that implements the `IReplicaMetadata` interface. An instance of the class that implements the `IReplicaMetadata` interface is created via the `MetaDataStore` factory class for a particular storage.

SyncContext class C#

Advanced

Namespace `Terrasoft.Sync`

A class representing the aggregation of providers and metadata for SyncAgent.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `SyncContext` class.

Constructors

`SyncContext()`

Creates a class instance.

Properties

`CurrentSyncStartVersion` `DateTime`

The current date and time synchronization in UTC. Set after the metadata update.

`LastSyncVersion` `DateTime`

The date and time of the last synchronization in UTC.

`LocalProvider` `LocalProvider`

Enables you to work with `LocalItem`.

`Logger` `ISyncLogger`

The object that enables messages to be saved into the integration log.

`MsgLogger` `Terrasoft.Sync.SyncMsgLogger`

Message logger for synchronization engine.

`RemoteProvider` `RemoteProvider`

External service or application, data from which is synchronized with Creatio.

`ReplicaMetadata` `IReplicaMetadata`

Works with metadata.

`UserConnection` `UserConnection`

User connection.

Methods

```
void LogError(SyncAction operation, SyncDirection direction, System.string format, params System
void LogError(SyncAction operation, SyncDirection direction, System.string format, System.Except
```

Writes error message to the log.

Parameters

`operation`	Log action for an object synchronization.
`direction`	Synchronization direction.
`format`	Format.
`exception`	Exception that caused this error.
`args`	Format parameters.

```
void LogInfo(SyncAction operation, SyncDirection direction, System.string format, params System.
```

Writes information message to the log.

Parameters

`operation`	Log action for an object synchronization.
`direction`	Synchronization direction.
`format`	Format.
`args`	Format parameters.

```
void SetLookupColumnValue(Entity entity, System.string lookupColumnName, System.string lookupDis
void SetLookupColumnValue(Entity entity, System.string lookupColumnName, System.string lookupDis
```

Sets entity lookup column value.

Parameters

`entity`	Instance of the `Terrasoft.Core.Entities.Entity` class.
`lookupColumnName`	Lookup column name.
`lookupDisplayValue`	Lookup column display value.
`lookupDisplayColumnName`	Lookup display column name.

RemoteProvider class C#

Advanced

Namespace `Terrasoft.Sync`

`RemoteProvider` — a basic class that enables you to work with an external storage. In fact, it is the only way to work with external systems.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `RemoteProvider` class.

Properties

`StoreId Guid`

ID of external storage that will be synchronized.

`Version DateTime`

Date and time of the last synchronization in UTC.

`SyncItemSchemaCollection List<SyncItemSchema>`

External storage element schema.

`RemoteChangesCount Int`

Number of items processed from external storage.

`LocalChangesCount Int`

Number of items processed from local storage.

Methods

`abstract IEnumerable<Type> KnownTypes()`

Returns a collection of all types that implement the `IRemoteItem` interface. `SyncAgent` builds the `SyncItemSchema` instances that describe the entities that participate in synchronization.

`abstract void ApplyChanges(SyncContext context, IRemoteItem synItem)`

Applies changes to external storage element.

Parameters

`context`	Synchronization context.
`syncItem`	Item of the synchronization.

```
abstract void CommitChanges(SyncContext context)
```

Called after processing changes in external and local storage. Intended for the implementation of necessary additional steps for the specific integration implementation.

```
abstract IEnumerable<IRemoteItem> EnumerateChanges(SyncContext context)
```

Returns an enumeration of new and modified elements of external storage.

```
abstract IRemoteItem LoadSyncItem(SyncItemSchema schema, string id)
virtual IRemoteItem LoadSyncItem(SyncItemSchema schema, ItemMetadata itemMetadata)
```

Fills in the `IRemoteItem` instance with data from external storage.

Parameters

`schema`	Synchronization schema.
`id`	Unique foreign key.
`itemMetadata`	Synchronization metadata.

```
abstract IRemoteItem CreateNewSyncItem(SyncItemSchema schema)
```

Creates a new instance of `IRemoteItem`.

```
abstract IEnumerable<LocalItem> CollectNewItems(SyncContext context)
```

Returns an enumeration of new Creatio entities that will be synchronized with external storage.

```
virtual SyncConflictResolution ResolveConflict(IRemoveItem syncItem, ItemMetadata itemMetaData,
```

Resolves conflicts between changed elements of local and external storages. By default, (`RemoteProvider`) priority is given to changes in Creatio.

```
virtual bool NeedMetaDataActualization()
```

Returns the sign that checks whether there is a need to update the metadata before starting synchronization.

```
virtual IEnumerable<ItemMetadata> GetLocallyModifiedItemsMetadata(SyncContext context, EntitySch
```

Returns synchronization elements changed in the local store since the last synchronization.

Parameters

`context`	Synchronization Context.
`modifiedItemsEsq`	Request to the scheme object `Terrasoft.Core.Configuration.SysSyncMetaData`.

IRemoteItem interface C#

Advanced

Namespace `Terrasoft.Sync`

Interface of the external storage element.

The class that implements the `IRemoteItem` interface is an indivisible unit of synchronization and represents one element of the synchronization of the external data storage. This class is a container for data coming from an external system, and it knows how to convert the data to the `Entity` entity, and vice versa. The interface contains two methods: `FillLocalItem` and `FillRemoteItem` for converting external synchronization elements (`IRemoteItem`) to `LocalItem`, and vice versa.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the members of the `IRemoteItem` interface.

Methods

```
void FillLocalItem(SyncContext context, ref LocalItem localItem)
```

Fills in properties of an element of the `LocalItem` local storage with values of external storage element. Used to apply changes in local storage.

```
void FillRemoteItem(SyncContext context, ref LocalItem localItem)
```

Fills in properties of an element of external storage with element values from the `LocalItem` local storage. Used to apply changes in external storage.

MapAttribute class C#

Namespace `Terrasoft.Sync`

The `Map` attribute decorates the `IRemoteItem` interface implementations.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `MapAttribute` class.

Properties

`SchemaName` `string`

Object schema name.

`Order` `int`

The entity processing order for the synchronization element.

`IsPrimarySchema` `bool`

A flag that indicates that this schema is a key element of this synchronization element. It can be installed in one schema only.

`PrimarySchemaName` `string`

The schema name of the main object. It can not be set in tandem with `IsPrimarySchema`.

`ForeignKeyColumnName` `string`

The column name for the connection with the details of the main object It can not be set in tandem with `IsPrimarySchema`.

`Direction` `SyncDirection`

It specifies the synchronization direction for the objects of this type. By default - `DownloadAndUpload`.

If it contains the `Download` flag - the changes will not apply to Creatio.

If it does not contain the `Upload` flag - the new entities will not be selected from Creatio.

`FetchColumnNames` `String[]`

The names of the columns that will be loaded from the local storage.

Examples of using map attribute

`SchemaName` is the main parameter. This is the name of the `EntitySchema` that is included in the current

synchronization element.

```
[Map("Activity", 0)]
[Map("ActivityParticipant", 1)]
public class GoogleTask: IRemoteItem {
    ...
```

This class declaration task from Google Calendar will sync with the activity and its participants from Creatio.

The second parameter, `Order`, specifies in which order `Entity` will be stored in the local storage. `Activity` is indicated first, because `ActivityParticipant` stores a link to the created activity.

In most cases, `SyncAgent` can automatically generate a request for a sample of the new elements of synchronization with Creatio. To do this, you must specify the basic entity and method of communication with the details:

```
[Map("Activity", 0, IsPrimarySchema = true)]
[Map("ActivityParticipant", 1, PrimarySchemaName = "Activity", ForeingKeyColumnName = "Activity"
public class GoogleTask: IRemoteItem {
    ...
```

In this case, a request for new activities will be sent to the database along with a request for each selected activity to receive their participants.

LocalProvider class C#

Advanced

Namespace `Terrasoft.Sync`.

Local Storage (`LocalProvider`) - encapsulates the work with data in internal storage (Creatio).

`LocalProvider` - basic class that implements work with the local storage. Enables you to work with `LocalItem`.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `LocalProvider` class.

Constructors

`public LocalProvider(UserConnection userConnection)`

Creates a class instance using `UserConnection`.

Properties

`StoreId` `Guid`

ID of Local storage that will be synchronized.

`MaxItemsPerSelect` `int`

Maximum items in select count.

`UserConnection` `UserConnection`

`UserConnection` instance.

Methods

`static void AddItemSchemaColumns(EntitySchemaQuery esqForFetching, EntityConfig entityConfig)`

Adds `EntitySchemaQuery` column specified in `EntityConfig`.

Parameters

`esqForFetching`	Instance of the `Terrasoft.Core.Entities.EntitySchemaQuery` class.
`entityConfig`	Instance of the `Terrasoft.Sync.EntityConfig` class.

`void ApplyChanges(SyncContext context, LocalItem entities)`

Applies changes to each `SyncEntity` in `LocalItem`.

Parameters

`context`	Synchronization context.
`entities`	Items of the synchronization.

`LocalItem FetchItem(ItemMetadata itemMetaData, SyncItemSchema itemSchema, bool loadAllColumns =`

Loads a collection of entities associated with a particular synchronization element.

Parameters

`itemMetaData`	Synchronization element.
`itemSchema`	Synchronization schema.
`loadAllColumns`	If true, loads all entities columns, loads only columns from entity `EntityConfig` instance.

SyncEntity class C#

Advanced

Namespace `Terrasoft.Sync`.

The `SyncEntity` class encapsulates `Entity` instance and all the necessary actions to perform the synchronization of the instance properties.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `SyncEntity` class.

Properties

`EntitySchemaName` `string`

The name of the schema for which the wrapper is created.

`Entity` `Terrasoft.Core.Entities.Entity`

`Entity` for which the wrapper is created.

`State` `Terrasoft.Sync.SyncState`

The last action performed on the `entity` (0 - not changed, 1 - new, 2 - changed 3 - deleted).

`Action` `Terrasoft.Sync.SyncAction`

Action to be done with the entity.

`ExtraParameters` `string`

Additional information that can be bound to a specific entity by an external provider for a specific purpose.

SyncItemSchema class C#

Advanced

Namespace `Terrasoft.Sync`

Synchronization element entity settings schema.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `SyncItemSchema` class.

Properties

`SyncValueName` `string`

Entity type name.<

`SyncValueType` `Type`

Entity type.

`PrimaryEntityConfig` `Terrasoft.Sync.EntityConfig`

Synchronization element entity settings.

`Configs` `List<EntityConfig>`

Synchronization element entity settings list.

`DetailConfigs` `List<DetailEntityConfig>`

Synchronization element detail entity settings list.

`Direction` `Terrasoft.Sync.SyncDirection`

It specifies the synchronization direction for the objects of this type. By default - `DownloadAndUpload`.

If it does not contain the `Download` flag, changes will not be applied in Creatio.

If it does not contain the `Upload` flag, new entities will not be selected from Creatio.

Methods

```
static SyncItemSchema CreateSyncItemSchema(Type syncValueType)
```

It creates a configuration element synchronization entity with all the settings of the related entities.

`void Validate(UserConnection userConnection)`

The method checks that `EntityConfig` is well-formed.

If authentication fails, an exception is applied. If the `EntityConfig` schema name is specified twice, `DublicateDataException` is generated. If the name of the defunct schema is specified, `InvalidSyncItemSchemaException` is generated).

EntityConfig class C#

Advanced

Namespace `Terrasoft.Sync`.

Synchronization element entity settings.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `EntityConfig` class.

Properties

`SchemaName string`

Object schema name.

`Order int`

The order of processing entities for a synchronization element. The lower the value, the sooner the entity will be processed in the processing of the synchronization element.

`Direction Terrasoft.Sync.SyncDirection`

It specifies the synchronization direction for the objects of this type. By default - `DownloadAndUpload`.

If it does not contain the `Download` flag, changes will not be applied in Creatio.

If it does not contain the `Upload` flag, new entities will not be selected from Creatio.

`FetchColumnNames string[]`

The names of the columns that will be loaded from the local storage. If the value is not specified, it will load all object columns.

DetailEntityConfig class C#

Advanced

Namespace `Terrasoft.Sync`.

Synchronization element detail entities settings.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `DetailEntityConfig` class.

Properties

`PrimarySchemaName` `string`

Creatio main synchronized entity schema name.

`ForeignKeyColumnName` `string`

The column name for the connection with the details of the main object.

LocalItem class C#

Advanced

Namespace `Terrasoft.Sync`.

One or more objects from Creatio that are synchronized with external storage as a unit. It contains a set of `SyncEntity` class instances.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `LocalItem` class.

Properties

`Entities` `Dictionary<string, List<SyncEntity>>`

The `SyncEntity` collection that is set in accordance with a `SyncItem`. It contains a collection of "key-value" pairs, where the key is the schema name, and the value is the `SyncEntity` collection of the scheme.

`Version` `DateTime`

The highest value of the date and time of the modification of all `Entities` in `LocalItem`.

`Schema SyncItemSchema`

Synchronization element entity settings schema.

Methods

`void AddOrReplace(string schemaName, SyncEntity syncEntity)`

Adds new `SyncEntity` to the collection. If `SyncEntity` with `EntityId` already exists, it replaces it.

`SyncEntity Add(UserConnection userConnection, string schemaName)`

Creates and adds a new `SyncEntity` collection.

SysSyncMetaData class C#

Advanced

The auxiliary `SysSyncMetaData` metadata table is used for synchronization, which is the junction between the outer `RemoteItem` table (synchronizing element in external storage) and `LocalItem` (synchronization element in Creatio). Each table row is represented in the system as an instance of `SysSyncMetaData` class.

Properties

`RemoteId string`

Element ID in external storage.

`SyncSchemaName string`

Synchronized element schema name.

`LocalId Guid`

Element ID in local storage.

`IsLocalDeleted bool`

It indicates whether an item has been removed from the local storage since the last synchronization. The parameter is updated before the synchronization and application of changes in the local storage. On the basis

of its value, when selecting modified elements from local storage, the `SyncEntity` state is set. Obsolete, left for compatibility. `LocalState` is currently used.

`IsRemoteDeleted` `bool`

It indicates whether an item has been removed from the external storage since the last synchronization. Obsolete, left for compatibility. `RemoteState` is currently used.

`Version` `Date`

Date of the last element modification.

`ModifiedInStoreId` `Guid`

`Id` of storage in which the last modification was performed.

`CreatedInStoreId` `Guid`

`Id` of storage in which the synchronization element was created.

`RemoteStoreId` `Guid`

`Id` of external storage with which the element was synchronized.

`ExtraParameters` `string`

Additional element parameters.

`LocalState` `int`

Element state in local storage (0 - not modified, 1 - new, 2 - modified, 3 - deleted).

`RemoteState` `int`

Element state in external storage (0 - not modified, 1 - new, 2 - modified, 3 - deleted).

MetaDataStore class C#

Advanced

Namespace `Terrasoft.Sync`.

Creates the specific class instance that implements the `IReplicaMetadata` interface for a storage.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `MetaDataStore` class.

Methods

`IReplicaMetadata GetReplicaMetadata(Guid localStoreId, Guid remoteStoreId)`

Creates the class instance that implements the `IReplicaMetadata` interface for the specific storage.

Parameters

`localStoreId`	Local storage id.
`remoteStoreId`	Remote storage id.

`void RemoveReplicaMetadata(Guid storageId)`

Removes all metadata for storage.

ItemMetaData class C#

Advanced

Namespace `Terrasoft.Sync`.

This class is an indivisible object of metadata synchronization. It contains a set of metadata for each synchronization element (`SysSyncMetadata` element collection).

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the methods, properties, parent classes, and implemented interfaces of the `ItemMetaData` class.

Properties

`RemoteId string`

Element ID in external storage..

`RemoteItemName string`

Element name in external storage.

IReplicaMetadata interface C#

Advanced

Namespace `Terrasoft.Sync`.

Class implementing the `IReplicaMetadata` interface, encapsulates the synchronization metadata and works with `ItemMetadata` objects.

Note. Use the ".NET class libraries of platform core" documentation to access the full list of the members of the `IReplicaMetadata` interface.

Properties

`RemoteStoreId` `Guid`

External storage ID.

`LocalStoreId` `Guid`

Local storage ID.

Methods

`ItemMetadata FindItemMetaData(string remoteItemId)`

Finds and returns the `ItemMetadata` synchronization metadata object by an ID in the `remoteItemId` external storage.

`void UpdateItemMetadata(ItemMetadata oldItemMetaDatas, IRemoteItem remoteItem, LocalItem localIt`

Updates metadata after synchronization.

Parameters

`oldItemMetaDatas`	Metadata instance from previous synchronization.
`remoteItem`	Remote item instance.
`localItem`	Local item instance.
`changesToBpm`	Current synchronization direction.

`IEnumerable<ItemMetadata> EnumerateItemsWithChangesInBpm (SyncContext context)`

Returns a collection of `ItemMetadata` objects that have been modified since the last synchronization and not processed during the current synchronization session.

`void CollectChangesInSyncedEntities (UserConnection userConnection, string schemaName, DateTime`

Updates metadata for synchronization elements modified in Creatio. If an element has been modified since the last `SysMetadata` synchronization, the `Version` column will be filled in with the date of element modification. The `ActualizeSysSyncMetaData` procedure is used to update metadata.

Parameters

`userConnection`	`Terrasoft.Core.UserConnection` Instance.
`schemaName`	Sync entity name.
`lastSyncVersion`	Last synchronization date.

`void CollectNewDetailsForSyncedEntities (UserConnection userConnection, DetailEntityConfig detai`

Creates new records in the `SysSyncMetaData` table for the synchronization element details.

Parameters

`userConnection`	User connection.
`detailEntityConfig`	Instance of the `Terrasoft.Sync.DetailEntityConfig` class.
`remoteItemName`	Remote storage item name.
`lastSyncVersion`	Last synchronization date.

`bool TryResolveRemoteId (Guid localId, out string remoteId)`

Returns the element ID in the external `remoteId` storage from metadata by a unique `localId` synchronization element in Creatio. If an element is marked as deleted, the `remoteId` wil not be returned, and the method will return `false`.

`bool TryResolveExtraParameters (Guid localId, out string extraParameters)` Boolean

Returns additional `extraParameters` parameters for the synchronization element by `localId`. If `extraParameters` are not found, the method returns `false`.

Synchronizing with MS Exchange

Advanced

Integration with various entities of MS Exchange via EWS protocol (Exchange Web Services) is supported by the Sync Engine synchronization mechanism.

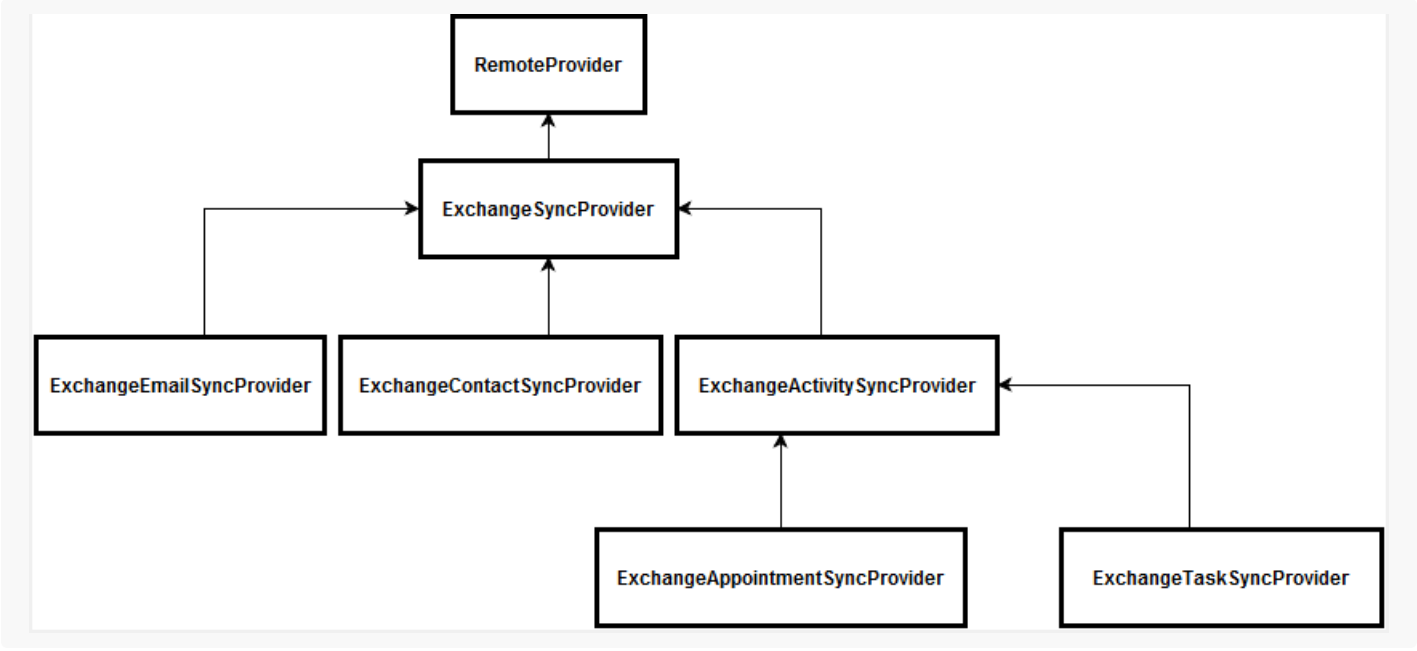
Synchronizing tasks with MS Exchange

The process runs in three stages:

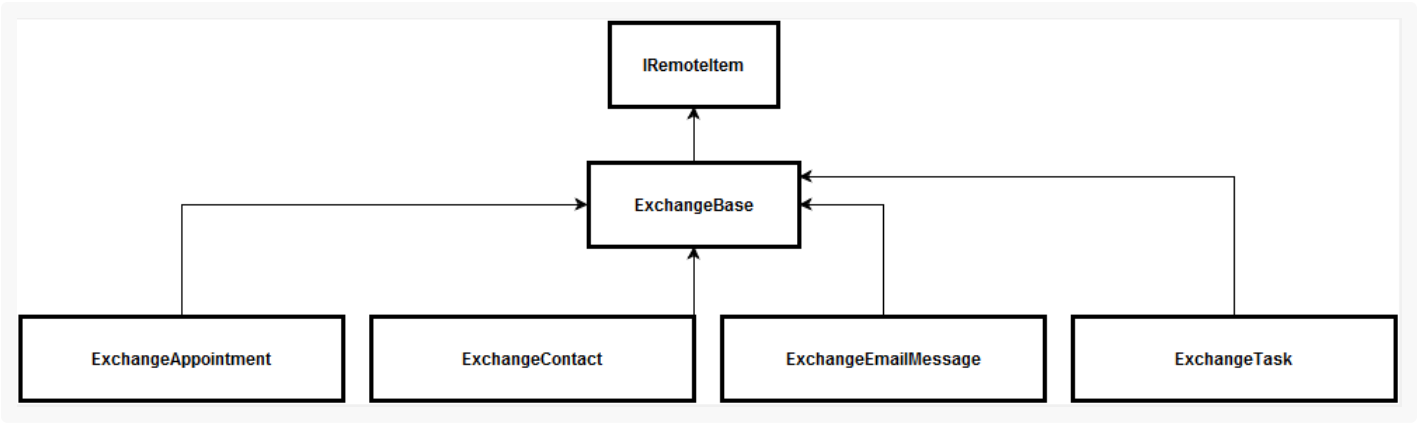
1. Retrieving changes from MS Exchange and applying them;
2. Retrieving changes from Creatio and applying them
3. Creating new tasks from Creatio in MS Exchange.

Integration classes

As described in the "Sync Engine synchronization mechanism basics " article, in order to implement an integration using this mechanism, you need a class that implements the logic of the external storage (an heir of the `RemoteProvider` class). The hierarchy of provider classes is shown in figure. You also need a class that implements the `IRemoteItem` interface, which represents a single instance of a synchronization item (in this case — the MS Exchange task). The `RemoteItem` hierarchy is shown in figure.



The `ExchangeTaskSyncProvider` class is the service provider for the MS Exchange external storage. This class implements the logic of selecting data and saving changes in Creatio and MS Exchange. The `ExchangeTask` class implements the `IRemoteItem` interface. The logic of filling in data in the corresponding systems is implemented in it.



Synchronized data

The correspondence of Creatio objects to the `ExchangeTask` class fileds is shown on table.

The correspondence of Creatio objects to the `ExchangeTask` class fileds

Creatio object	**Object field**	**The `ExchangeTask` class field**
`Activity`	`Title`	< `Subject`
	`StartDate`	`StartDate`
	`DueDate`	< `CompleteDate` or `DueDate` depending on whether a task is finished or not.
	`Priority`	`Importance`
	`Status`	`Status`
	`RemindToOwner`	`IsRemindSet`
	`RemindToOwnerDate`	`ReminderDueBy`
	`Notes`	`Body.Text`

Logic of selecting data for synchronization

To select changes to the list of tasks selected for MS Exchange folder synchronization, use the following terms: select tasks for MS Exchange, which were modified after the last task synchronization or an MS Exchange task, which was not marked as synced. The MS Exchange task which were modified have corresponding activities in Creatio. The updated changes are applied in the corresponding system.

When you select modified Creatio activities, select the following:

- activities that are recorded in the metadata synchronization as MS Exchange tasks.

- activities that have the current user as an author.
- activities with a date of the last modification that does not correspond to the date of the last synchronization.

When selecting new Creatio activities, configure a set of common and custom filters. The main filter conditions are:

1. Activity type is not "email".
2. Activity does not have the [*Display in calendar*] checkbox selected.

A user can specify activity groups that will be exported from Creatio.

Filling in the [*Start Date*] and [*Due Date*] fields

The `ExchangeTask` object has several features for working with start date and due date:

- These fields are stored without time values. If you change a task in MS Exchange after synchronization, Creatio will apply the date from the MS Exchange task, and the time from the Creatio activity.
- The due date in `ExchangeTask` has two fields: due date and complete date.
- The start date and due date are optional in MS Exchange. If either of them is not filled in, the current date is set. Due to this, conflicts may arise, as both the start date and due date are required in Creatio, and the start date should be earlier than the due date.

Synchronizing contacts with MS Exchange

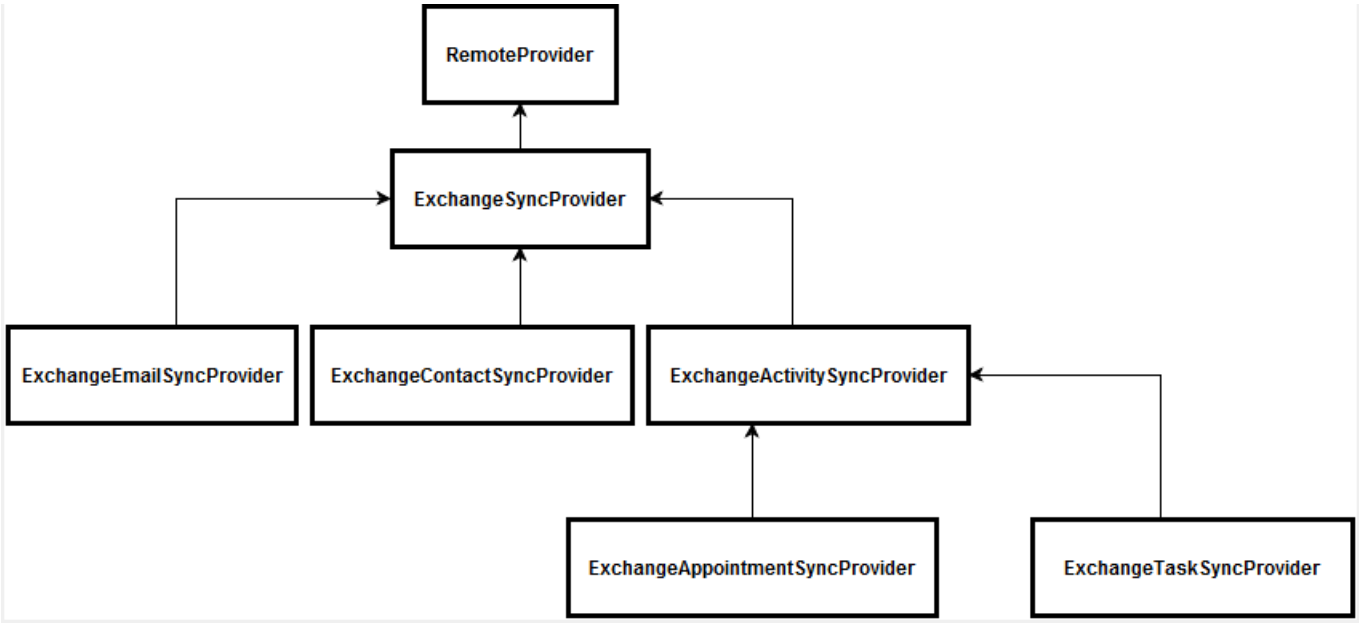
The process runs in three stages:

1. Retrieving changes from MS Exchange and applying them.
2. Retrieving changes from Creatio and applying them.
3. Creating new contacts from Creatio in MS Exchange.

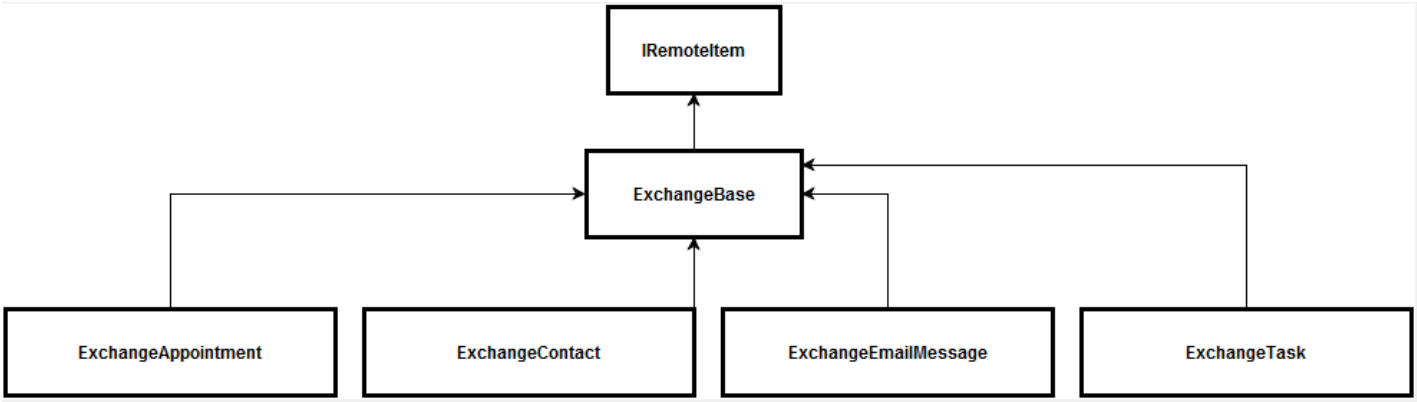
Integration implementation

As described in the "Sync Engine synchronization mechanism basics " article, to implement integration using this mechanism, a class is required that implements the logic of working with external storage (`RemoteProvider` heir) and a class that implements the `IRemoteItem` interface, which is an instance of the synchronization element (in our case — MS Exchange contact).

`RemoteProvider` hierarchy schema



`RemoteItem` hierarchy schema



The following classes are used for contact synchronization:

- The `ExchangeContactSyncProvider` class is the service provider for the MS Exchange external storage. This class implements the logic of selecting data and saving changes in Creatio and MS Exchange.
- The `ExchangeContact` class implements the `IRemoteItem` interface. The logic of filling in data in the corresponding systems is implemented in it.
- The `ExchangeAddressDetailsSynchronizer` class contains methods for converting contact addresses.
- The `ExchangeEmailAddressDetailsSynchronizer` class contains methods for converting contact email addresses.
- The `ExchangePhoneNumbersDetailsSynchronizer` class contains methods for convertingcontacts phones.

The logic of filling in details is located in separate classes, as there are significant differences in the formats of data storage in Creatio and MS Exchange. Additional conversion is required.

Synchronized data

The correspondence of Creatio objects to the Contact MS Exchange class fields is shown on table.

The correspondence of Creatio objects to the Contact MS Exchange class fields

Creatio object	Object field	The `Contact MS Exchange` class field
`Contact`	`Name`	`DisplayName`
	`Surname`	`Surname`
	`GivenName`	`GivenName`
	`MiddleName`	`MiddleName`
	`Account`	`CompanyName`
	`JobTitle`	`JobTitle`
	`Department`	`Department`
	`BirthDate`	`Birthday`
	`SalutationType`	`TitleTag`
	`Gender`	`GenderTag`
`ContactCommunication`	`Number`	The `PhoneNumbers` collection values
	`CommunitactionType`	The `PhoneNumbers` collection value key
`ContactAddresses`	`City`	The `PhysicalAddresses` collection element `City` field
	`Country`	The `PhysicalAddresses` collection element `CountryOrRegion` field
	`Region`	The `PhysicalAddresses` collection element `State` field
	`Address`	The `PhysicalAddresses` collection element `Street` field
	`Zip`	The `PhysicalAddresses` collection element `PostalCode` field
	`AddressType`	The `PhysicalAddresses` collection value key<

The correspondence of communication types is shown in table.

The correspondence of communication types of Creatio to MS Exchange

Creatio communication type	MS Exchange communication type
`Email`	`EmailAddress1`, `EmailAddress2`, `EmailAddress3`
`WorkPhone`	`BusinessPhone`, `BusinessPhone2`
`HomePhone`	`HomePhone`
`MobilePhone`	`MobilePhone`

The correspondence of addresses is shown in table.

The correspondence of addresses of Creatio to MS Exchange

Creatio address type	MS Exchange address type
`HomeAddress`	`Home`
`BusinessAddress`	`Business`

Logic of selecting data for synchronization

To select changes to the list of contacts selected for MS Exchange folder synchronization, use the following terms: select contacts for MS Exchange, which were modified after the last contact synchronization or an MS Exchange contact, which was not marked as synced. The contacts which were modified have corresponding contacts in Creatio. The updated changes are applied in the corresponding system.

When you select Creatio contacts for synchronization, select the following:

- contacts that have the current user as an author
- contacts with a date of last modification that does not correspond to the date of the last synchronization.
- contacts that were not used on the first step of synchronization

User settings also affect the rules for selecting new contacts in Creatio. Three settings are available:

1. Synchronize employees contacts. When you select this setting, the "Contact type" filter will be added to the request, and only the "Employee" type contacts will be synchronized.
2. Synchronize customers contacts. When you select this setting, the "Contact type" filter will be added to the request, and only the "Customer" type contacts will be synchronized.
3. Sync contacts from certain groups. When you select this setting, the selected contact group filters will be added to the request.

Additional features

Using the advanced contact keys in external storage

A situation may occur when there is a large number of MS Exchange contacts and of them will receive the same ID. As a result, synchronization may not correctly identify the appropriate contact in Creatio. To work around this situation, there are advanced external keys, which can be enabled by the [*Use composite IDs for MS Exchange synchronized contacts*] setting. Setting code - `UseComplexExchangeContactId`. After enabling it, you may need to resynchronize.

Synchronizing appointments with MS Exchange

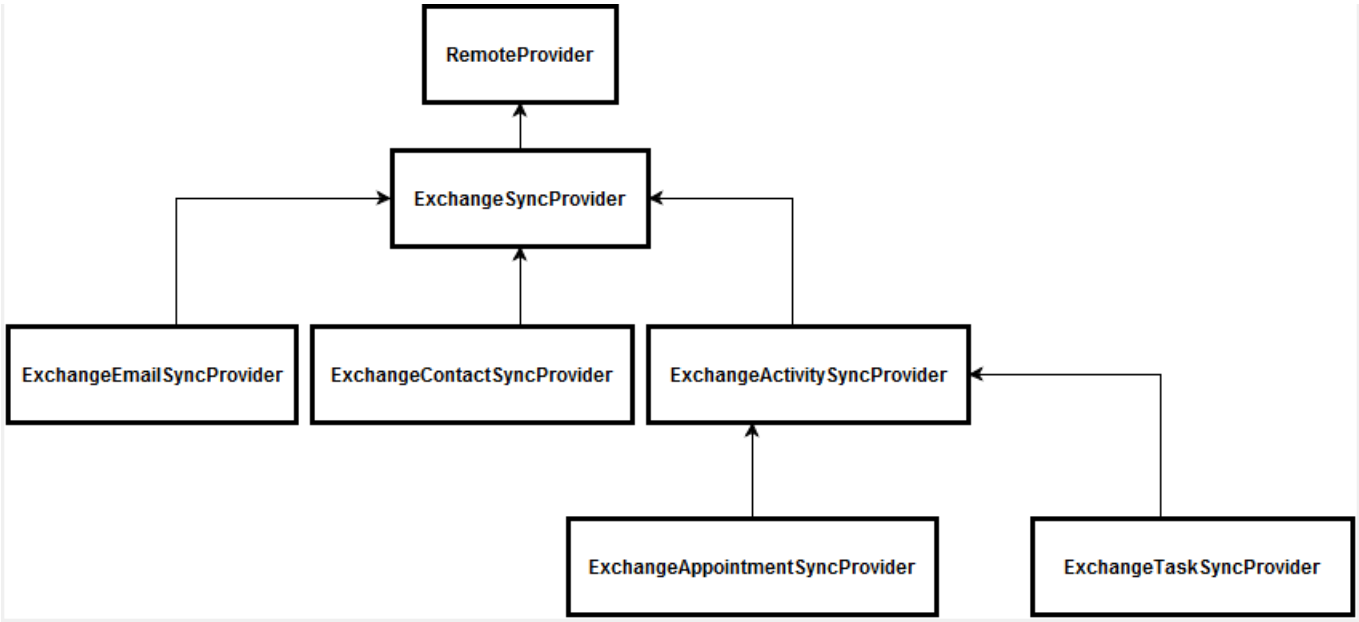
Creatio appointment synchronization is performed only for new activities or when the `Title`, `Location`, `StartDate`, `DueDate`, `Priority`, `Notes` fields are modified. The hash stored in the additional metadata parameters (in the `ExtraParameters` field) is formed according to these fields. If an appointment has been changed in Creatio, and the hash for `ExtraParameters` does not match the new hash, this appointment should be synchronized.

The process runs in three stages:

1. Retrieving changes from MS Exchange and applying them.
2. Retrieving changes from Creatio and applying them.
3. Creating new appointments from Creatio in MS Exchange.

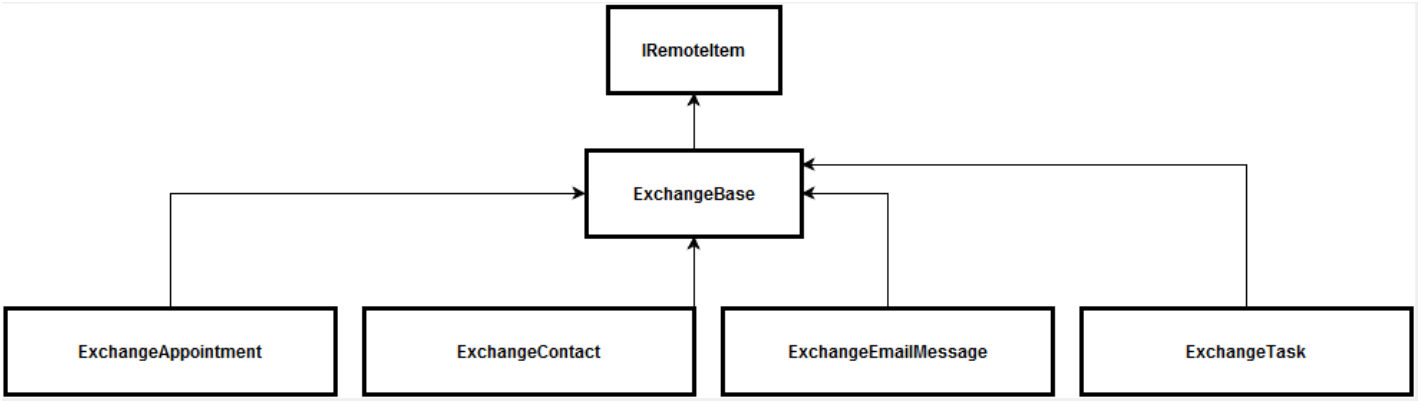
Integration implementation

As described in the "Sync Engine synchronization mechanism basics " article, to implement integration using this mechanism, a class is required that implements the logic of working with external storage (`RemoteProvider` heir) and a class that implements the `IRemoteItem` interface, which is an instance of the synchronization element (in our case — MS Exchange appointment).



The `ExchangeAppointmentSyncProvider` is the provider used to work with the Exchange external storage. It contains the logic of selecting data and saving changes in Creatio and Exchange.

The `ExchangeAppointment` class implements the `IRemoteItem` interface, in which the logic of filling in data in Creatio objects is implemented.



Synchronized data

The correspondence of Creatio objects to the `ExchangeAppointment` class fileds is shown on table.

The correspondence of Creatio objects to the `ExchangeAppointment` class fileds

Creatio object	Object field	MS Exchange Appointment corresponding field
`Activity`	`Title`	`Subject`
	`Location`	`Location`
	`StartDate`	`StartDate`
	`DueDate`	`CompleteDate` or `DueDate` depending on whether an appointment is finished or not.
	`Priority`	`Importance`
	`Status`	Filled in as follows: If the status is not specified and the due date is later than the current date — Creatio sets the "New Appointment" status. If the due date is earlier than the current date, the status is set as a closed appointment with the "Information received" status.
	`RemindToOwner`	`IsReminderSet`
	`RemindToOwnerDate`	`ReminderDueBy`
	`Notes`	`Body.Text`
`ActivityParticipant`	`InviteResponse`	If the checkbox in MS Exchange is selected that identifies that an appointment was received and the user is its owner, and the "Appointment confirmed" checkbox is selected. Otherwise, it the checkbox is selected that identifies that the appointment was canceled.

Logic of selecting data for synchronization

To select changes to the list of appointments selected for MS Exchange folder synchronization, use the following terms: select appointments for MS Exchange, which were modified after the last contact synchronization or an MS Exchange appointment, which was not marked as synced. The appointments which were modified have corresponding contacts in Creatio. The updated changes are applied in the corresponding system.

When you select modified Creatio activities, select the following:

- activities which are recorded in the synchronization metadata as MS Exchange appointments via `RemoteId` (determined by a unique appointment ID in the `ICalId` calendar);
- activities with a date of the last modification that does not correspond to the date of the last synchronization.

- one appointment in Creatio corresponds to several appointments in MS Exchange for each participant.

When selecting new Creatio activities, configure a set of common and custom filters. The main filter conditions are:

1. Activity type is not "Email".
2. Activity has the [*Display in calendar*] checkbox selected.

A user can specify activity groups that will be exported from Creatio.

Logic of selecting appointment participants

When you synchronize an activity from MS Exchange to Creatio, only contacts that have email addresses from the list of appointment participants in MS Exchange are added to the [*Participants*] detail.

When you synchronize activities from Creatio to MS Exchange, the appointment participants for MS Exchange are filled in with primary contact email addresses.

Synchronizing emails with MS Exchange

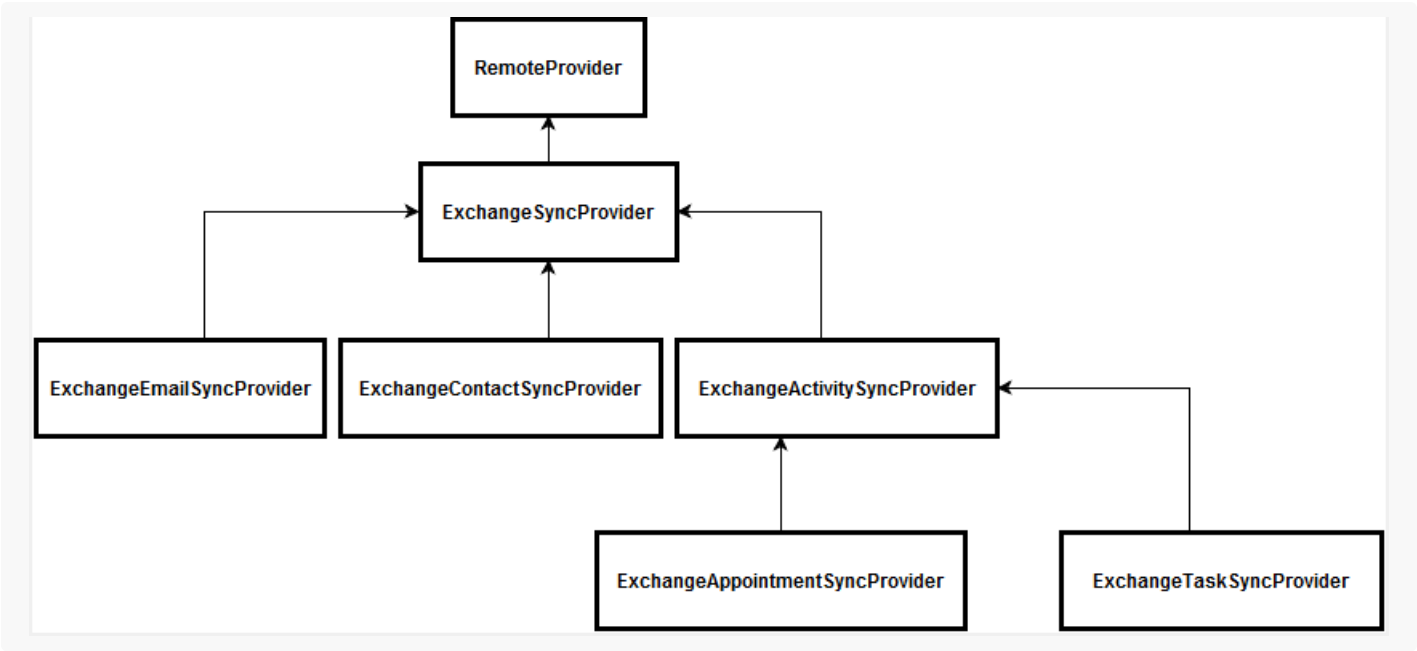
 Advanced

Synchronization with various services of MS Exchange via EWS protocol (Exchange Web Services) is supported by the Sync Engine synchronization mechanism. This article describes the synchronization of email in Creatio with MS Exchange. Email in Creatio is synchronized only from MS Exchange. Since emails can no longer be modified after they have been sent, only the emails that have not been synchronized previously are synchronized. The main difference between the email synchronization mechanism and integration is the email search engine in Creatio. Since the same email can be synchronized on behalf of any of the recipients or even via IMAP protocol, metadata synchronization can not be used for searching for previously synchronized emails. Use subject, send date and message to search for emails . All markups and spaces are removed from the message. To speed up the search, use the md5 hash that is stored in the `MailHash` column of the `Activity` object.

The second difference of this synchronization is that attachments are synchronized by a separate process, after all emails are processed. This is done in order to make the attachment download time not affect the email save time.

Integration implementation

As described in the "Sync Engine synchronization mechanism basics " article, to implement integration using this mechanism, a class is required that implements the logic of working with external storage (`RemoteProvider` heir) and a class that implements the `IRemoteItem` interface, which is an instance of the synchronization element (in our case — email MS Exchange).

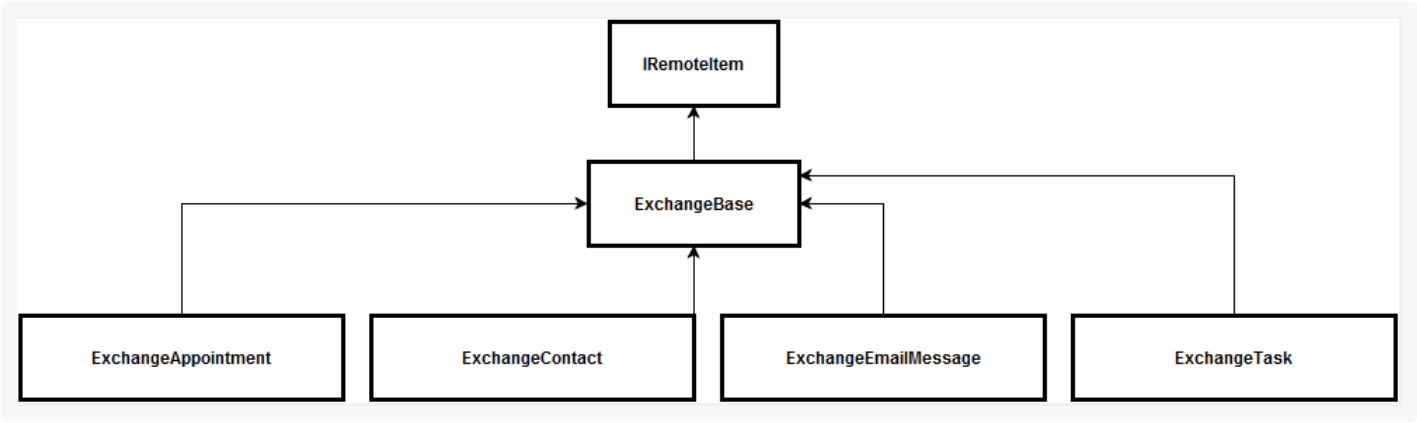


The `ExchangeEmailSyncProvider` class is the service provider for the exchange external storage. This class implements the logic of selecting data from MS Exchange.

The `ExchangeEmailMessage` class implements the `IRemoteItem` interface, in which the logic of filling in data in Creatio objects is implemented.

The `ExchangeUtility` class contains methods for the EWS API library and the methods used to download email attachments.

The `ExchangeEmailMessageUtility` class contains methods for converting the lookup values of the email fields.



Synchronized data

The correspondence of Creatio objects to the `EmailMessage` class fields is shown on table.

The correspondence of Creatio objects to the `EmailMessage` class fields

Creatio object	Object field	`EmailMessage` corresponding field
`Activity`	`Title`	`Subject`
	`Body`	`Body.Text`
	`Sender`	`Sender`
	`Recepient`	`ToRecepients`
	`CopyRecepient`	`CcRecepients`
	`BlindCopyRecepient`	`BccRecepients`
	`SendDate`	`DateTimeSpent`
	`Priority`	< `Importance`
	`DueDate, StartDate`	`DateTimeReceived`
`ActivityFile`	`Name`	`Name`
	`Data`	`Content`
	`Size`	`Content.Length`

Logic of filling in email participants

For an email to be displayed correctly only for users who have synchronized it, the following mechanism of filling in the [*Activity participants*] detail has been implemented. Conventionally, this logic can be divided into two parts:

1. Adding participants to a new email.
2. Updating the list of participants when an email changes (including re-synchronization).

Adding participants to a new email

The main value that affects who becomes the participants of an email is the [*Communication*] contact detail. If a contact has an email address in the [*Communication*] detail, and this email is listed in one of the email address fields ([*From*], [*To*], [*CC*], [*BCC*]), the contact can be added to the participants. Additionally, a check is made whether there is a system user for this contact who is not a portal user. A user is added to the participants only after they have synchronized their email.

Updating the list of participants

For a user to become a participant after the synchronization of an existing email, the list of participants is

updated - all participants who are not Creatio users are removed from the detail, and the algorithm of filling in detail for a new email runs. Thus, the users who have previously synchronized the email remain as participants, a new user is added, and the contact list is updated.

Logic of selecting data for synchronization

When selecting emails for synchronization with MS Exchange, use the following filter set: select emails that have been modified since the last synchronization and are not drafts. There is a limit for synchronization folders: "Deleted" and "Conflicting elements" folders do not participate in synchronization. When selecting emails the synchronization metadata is not taken into account. Always "save changes in Creatio". When processing each email, the system first checks for emails in Creatio. If an email already exists in Creatio, the participants are updated. If not, a new email is created. At the end of the synchronization, the system adds a task to synchronize attachments.