# **Occtoo generic importer**

Generic importer is assumed to become universal tool for importing data from inRiver into Occtoo. Basic idea is to control creation of datasources and their structure through json configuration file.

Exporter is implemented as inRiver extensions: scheduled, entity listener and link listener. Entity and Link listeners will take their events and create a ConnectorState that the Scheduled Extension will pick up and create, update, or delete the affected datasources according to the settings for the solution. There is also a Scheduled extension to perform full export but only if it was manually triggered. Intended to perform initially and in cases when full synchronization is required.

## **Extension settings structure**

Settings structure contains seven simple settings and one complex setting that is configuration for datasources structures.

1. Environment – a GUID used when creating media URLs.
2. DefaultLanguage – default language that will be used as backoff for localized fields.
3. DocumentIdForbidenChars – list of chars that are forbidden to be used in creation of onboarding document id. It is created as a list of forbidden chars and their replacements separated with ‘|’ sign. (e.g. “ ;\_|+;\_|/;\_” – forbidden chars are blank, + and / and they will be replaced with \_ if found in ID).
4. SkipFilteredEntities – Boolean to control if entities not fulfilling the filter values should be skipped in the export.
5. SkipMediaKeyGuidTransform - Boolean to control if the Environment GUID should be used when creating mediaIds
6. OcctooDataProviderId – obtained in the Occtoo studio to be able to import data.
7. OcctooDataProviderSecret – obtained in the Occtoo studio to be able to import data.
8. ExportSettings – complex structure that will be explained in the next chapter.

## **Export Settings Structure**

Export settings contains list of EntitySettings that each contains settings what entity is going to be exported, in what datasource and rules about linking and merging data with other entity type(s).

### **EntitySettings**

Properties:

1. Name – entity name in inRiver
2. DataSource – name of datasource in Occtoo where entity data will be imported.
3. EntityIdAlias – name of property where we are going to store entity id (inRiver system id)
4. UniqueIdFields – list of field ids that will be used to create document ID (fields from different entities can be used if they are going to be fully merged). Field values will be concatenated in one string separated by “\_” and sanitized from DocumentIdForbidenChars.
5. Type – entity type (entity, media, sku) used to define export flow (e.g. if it is media it will be uploaded to media service)
6. Filters – list of filter settings used to filter out unwanted entities.
7. ChildrenMerges – list of merge settings used to define rules how to merge linked entities (outbound links)
8. ParentMerges – list of merge settings used to define rules how to merge linked entities (inbound links)
9. ExceptionalFields – list of exceptional fields settings. Used to define rules how to export “special” fields, mostly non-existing fields in inRiver but calculated somehow. This is one of the places where you as a developer will be able to add new functionalities.
10. Fields – list of field settings that will define which fields are going to be exported and rules how to export some complex fields (e.g. sku level field). If null or empty list, all fields will be exported.

### **Filter Settings**

Properties:

1. Type – defines filter type how value is going to be matched (equal, all, any)
2. Field – inRiver field ID whose value will be checked.
3. Values – list of string values that will be used for filtering.

### **Merge Settings**

Properties:

1. Type – defines merging type
   1. None – default value, means no merging will happen.
   2. Full – two entities will be fully merged in one datasource in onboarding. All allowed fields from both entities will be extracted together in one list and multiple documents are going to be created (for every child entity one document while parent values are going to be repeated every time). The best example is Product-Item relation, here one document for each Item will be created with all the properties of the Product inherited down.
   3. Ids – ids from children (outbound links) are going to be extracted in one property of datasource (e.g. media ids for product)
   4. ParentIds – ids from all parents (inbound links) are going to be extracted in one property of datasource.
2. LinkUpdateType – defines which side of link is going to be updated when link is changed (UpdateSource or UpdateTarget) – here we need to be careful to check what is merge type.
3. Name – name of merging entity in inRiver
4. Link – name of link between entities in inRiver
5. DataSource – name of data ource where entity is going to be exported. In the event of a full merge this value must be the same as parent entity. In other case it must be different.
6. PropertyAlias – if merge type is Ids or ParentIds this is going to be property ID in onboarding datasource.

### 

### **Exception Field Settings**

This is one of the places where you as a developer will easily be able to add new functionalities as new Exception field types can be developed and used. To extend with your won Exception field just create a class implementing the IExceptionFieldExtractor interface and then update the ExceptionFieldType Enum to hold your type and use it in the setting for the Entity you wish to use it.

Properties:

1. Id – inRiver field id
2. Alias – will be used in onboarding datasource as property id. Mandatory if the field itself is part of regular export. (good practice would be to always set this value)
3. Type – type of exceptional field. It’s based of the Enum ExceptionFieldType