☰  ⬡  ordwaylabs  /  **ordway** 🔒                                    🔍  📥  ⬣

<> Code          ⊙ Issues  5k+       ⑂ Pull requests  284       ▶ Actions       ▦ Projects       📖 **Wiki**       ⚠

# Metadata

Edit    New page                                                        **Jump to bottom**

Pradeep edited this page on May 22 · **15 revisions**

## What is Metadata in Ordway?

Metadata is core of the system, which defines the default configuration, using these configuration specific result can be obtained. There are certain keys and value which has special meaning with Ordway system.

This metadata will potentially be used through out modules like filters/search/import/export...

It can be extended

## Metadata Structure

### Base

```
{
  "version": {
    "number": 1,
    "schema": {"subscription":...}
  }
}
```

### Schema Structure

```
{
  "subscription": {
    "fields": [...],
    "description": "",
    "child_association":["susbcription_charge"],
    "associations":["customer", "company"], # what association do it have
    "filter_associations":["customer"], # this will be used by filter
```

```
      "version": 1
    }
  }
```

## Path

```
config/metadata/modules/*.json
```

## Fields structure

```
[{
  "field_name": "name", // Column name
  "db_name": "db_name", // when field name is different for db name
  "field_type": "text", // input type (text, number, date, boolean) This will have ba
                        // input type details can be grabbed from Api get('/api/v1/me
  "display_name": "Customer Name", // Label
  "primary_key": false, // Boolean, indicate whether field is primary key
  "description": "Name of the Customer", // short description about field
  "tooltip": "", // tooltip on label
  "filter": true, // field is allowed for filter
  "search": true,// field is allowed for search
  "export": true,// field is allowed for export
  "import": true,// field is allowed for import
  "formatter": "", // formatter to be applied this is the method name(Which needs to
  "mapper": [], // change display value can be used both ways to -> from and from ->
  "allowed_values": [ // which valued are allowed ],
  "validations": { // That will applied on the source
   "min_value": 10,
   "max_value": 30,
  },
  "inheritance_config": { "value": "customer.customer_id" // if FK needs to be used t
  "filter_config": { "auto_suggest": true, "prefix": "customer" // specific configura
  "search_config": { "auto_suggest": true // specific configuration for search },
  "export_config": { "auto_suggest": true // specific configuration for export },
  "import_config": { "auto_suggest": true // specific configuration for import },
  "embedded_checkout": { // Specific for Self Service checkout Porta
        "editable": false,
        "displayable": true,
        "detail": false,
        "table": false
     }

},
{
  "field_name": "status", // Column name
```

```
    "db_name": "db_name", // when field name is different for db name
    "field_type": "text", // input type (text, number, date, boolean) This will have ba
                          // input type details can be grabbed from Api get('/api/v1/me
    "display_name": "Customer Status", // Label
    "primary_key": false, // Boolean, indicate whether field is primary key
    "description": "Status of the Customer", // short description about field
    "tooltip": "", // tooltip on label
    "filter": true, // field is allowed for filter
    "search": true,// field is allowed for search
    "export": true,// field is allowed for export
    "import": true,// field is allowed for import
    "formatter": "", // formatter to be applied this is the method name(Which needs to
    "mapper": [], // change display value can be used both ways to -> from and from ->
    "allowed_values": [
                { // which valued are allowed
                    "value": "active",
                    "label": "Active"
                },
                {
                    "value": "inactive",
                    "label": "In Active"
                }
    ],
    "validations": { // That will applied on the source },
    "inheritance_config": { "value": "customer.customer_id" // if FK needs to be used t
    "filter_config": { "auto_suggest": true // specific configuration for filter },
    "search_config": { "auto_suggest": true // specific configuration for search },
    "export_config": { "auto_suggest": true // specific configuration for export },
    "import_config": { "auto_suggest": true // specific configuration for import }

}]
```

###Field Type Config

```
{
  "text": [
      {
        "value": "equal",
        "label": "is"
      },
      {
        "value": "not_equal",
        "label": "is not"
      },
      {
        "value": "like",
        "label": "contain"
      },
```

```
      {
        "value": "not_like",
        "label": "does not contain"
      },
      {
        "value": "is_present",
        "label": "is present"
      },
      {
        "value": "is_not_present",
        "label": "is not present"
      }
    ]
  }
```

# **Metadata Api **

```
post('/api/v1/metadata', { object_keys: [
                        {name: 'customer' ,version: 1},{name: 'invoice'
,version: 1},{name: 'subscription' ,version: 1}
                        ], association: true})
```

**Params** object_keys is array of hash name is mandatory(it should match with the MetaConf file schema keys) and version is optional. if no version is provided it will take the latest one.

association(optional) true/false will add schema for associations

**Result** {"subscription":...,"customer":...}

Metadata is stored in dynamo db under table called **metadata** and **version** as primary key.

## Metadata can be Inserted/updated using Api's

Only CSV user is authorise to do it **Insert(Version in primary key)**

```
post('api/v1/metadata/add_schema', {"version": 1, "metadata_schema": {...schema}
```

### Update

```
put('api/v1/metadata/update_schema', {"version": 1, "metadata_schema":
{...schema}}))
```

## Get

```
get('api/v1/metadata')
```

Config File are cached

### To clean cache

```
delete('api/v1/metadata/flush_schema',{})
```

It will flush all cache related to schema

+ Add a custom footer

▼ **Pages**  111

Find a page…

▸ **Home**

▸ **Add user to a company**

▸ **Aging Report | Transaction history events**

▸ **Anomaly Detection Queries**

▸ **Avalara Avatax Address Validation**

▸ **Avalara Settings**

▸ **AWS Infrastructure Services**

▸ **AWS Ordway Dev accounts**

▸ **Change Mandrill API Gateway To Async Mode**

▸ **Code Merge Process**

▸ **Company and User Authentication Token Encryption in Database**

▸ **Component: FlexGrid**

▸ **Creating a new API Key for Clients**

▸ **Customer MRR Reset**

▸ **Data Discrepancy Scans and Follow Up actions**

Show 96 more pages…

+ Add a custom sidebar

## Clone this wiki locally

```
https://github.com/ordwaylabs/ordway.wiki.git
```