How To Store Enums in DB

- When calling an API, you should always send your enumeration's string value, not number value. So the value in the request should be like this: "OperationType": "Other".

{{baseUrl}}/api/Catalog/ActionDefinitions

```
{
    "Name": "Test Action",
    "CommandName": "TestCommand",
    "Description": "Test Action",
    "IconPath": "TestIconPath",
    "OperationType": "Other",
    "ActionType": "button",
    "Order": 99
}
```

> Enumerations can be standart Enumeration or FiSmartEnum. If it is FiSmartEnum, then a JsonConverter should be used for the field in model class.

Sample Model Class:

```
   public record ActionDefinitionInputModel : InputModelBase
    {
        public int Id { get; set; }
        public string Name { get; set; }

[JsonConverter(typeof(FiSmartEnumCodeConverter<OperationType, byte>))]
        public OperationType OperationType { get; set; }
        ...
    }
```

Sample Entity Class:

```
   public class ActionDefinition : EntityBaseWithBaseFieldsWithIdentity
    {
        public string Name { get; set; }
        public OperationType OperationType { get; set; }
        ...
    }
```

Configurator Class:

```
   public class ActionDefinitionConfigurator :
EntityConfigurator<ActionDefinition>
    {
        protected override void
OnConfigure(EntityTypeBuilder<ActionDefinition> builder)
        {
            builder.Property(m => m.Name).IsRequired(true);
            builder.Property(m => m.Name).HasMaxLength(50);
            builder.HasIndex(m => m.Name).IsUnique();

            builder.Property(m =>
m.OperationType).IsRequired(true).HasConversion(
                p => p.Value,
                p => OperationType.FromValue(p));
            ...
```

In this case, you sent enumeration string in request, and you store but you store int/byte value in DB as well.

- The field name should be same in model and entity classes. But for exceptional cases the model can use Enum, but entity can use Id like example below:

```
   public record CustomerInputModel : InputModelBase
    {
        public int Id { get; set; }
        public string Name { get; set; }

[JsonConverter(typeof(FiSmartEnumCodeConverter<ISOLanguageCode
s>))]
        public ISOLanguageCodes LanguageCode { get; set; }
        ...
    }
   public class Customer :
EntityBaseWithBaseFieldsWithIdentity, IFiRowVersionEntity
    {
        public string Name { get; set; }
        public ISOLanguageCodes LanguageId { get; set; }
        ...
    }
namespace Fi.Customer.Api
{
    public class AutoMapperProfile : Profile
    {
        public AutoMapperProfile()
        {
            CreateMap<CustomerInfoInputModel,
CustomerEntity>()
                .ForMember(dest => dest.LanguageId,
                           opt => opt.MapFrom(src =>
src.LanguageCode));
...
```

In this case, you sent enumeration string in request, but you store int/byte value in DB. These cases are generally for Country, Currency and Language.

How To Store Enum as List in DB

If you want to store enumeration data in db as enumcode1, enumcode2, enumcode3 format and as a varchar field, you can transport your data in c# classes as List<Enum> and put a conversion on EntityConfigurator class like below.

```
   public class CustomerRelationType :
EntityBaseWithBaseFieldsWithIdentity
    {
        public string Name { get; set; }
        public List<CustomerType> CustomerTypes { get; set; }
    }
   public class CustomerRelationTypeConfigurator :
EntityConfigurator<CustomerRelationType>
    {
        protected override void
OnConfigure(EntityTypeBuilder<CustomerRelationType> builder)
        {
            builder.Property(m => m.Name).IsRequired(true);
            builder.Property(m => m.Name).HasMaxLength(255);
            builder.HasIndex(m => m.Name).IsUnique();

            builder.Property(m =>
m.CustomerTypes).HasMaxLength(255)
                    .HasConversion(v => string.Join(',',
v.Select(s => s.ToString()).ToArray()),
                                   v => new
List<string>(v.Split(',',
StringSplitOptions.RemoveEmptyEntries))

.Select(x => (CustomerType) Enum.Parse(typeof(CustomerType),
x)).ToList());
        }
    }
   public record CustomerRelationTypeInputModel :
InputModelBase
    {
        [JsonIgnore]
        public int Id { get; set; }
        public string Name { get; set; }
        public List<CustomerType> CustomerTypes  { get; set; }
    }
   public record CustomerRelationTypeOutputModel :
OutputModelBase
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public List<CustomerType> CustomerTypes  { get; set; }
    }
```

Sample API Body

```
{
  "Name": "Relation Type Test",
  "CustomerTypes": [
        "Individual",
        "Corporate",
        "IndividualFirm",
        "JointAccount",
        "GroupCustomer"
    ]
}
```

> If your table will have many rows, then do not choose this method.