Приклади деяких перевезень з використанням C# версії TMCalcExtV2. Цей код також доступний в C# демо проекті який поставляється з TMCalcExtV2.

1. Приклад повагонного перевезення

```
var cons = new ConsSharp //головний об’єкт перевезення
{
    Dispatch = new DispatchWagonLoadSharp()//тип перевезення – повагонне, контейнерне…
    {
        DispatchParameters = new DispatchParametersSharp()
        {
            Speed = DispatchSpeedEnum.Freight,
        }
    },
    SpecialRate = new ConsSpecialRateSharp() // експедитори
    {
        Forwarder = 0,
        Recipient = 0,
        Sender = 0,
    },
    Locomotive = new ConsLocomotiveSharp() // тип рухомого складу
    {
        DeclaredType = DispatchLocomotiveEnum.Default,
    },
    OnDate = DateTime.Today,
    RouteHeader = new RouteHeaderConsSharp // маршрут перевезення
    {
        CountrySource = new CountrySharp()
        {
            Code = 804,
        },
        CountryTarget = new CountrySharp()
        {
            Code = 804,
        },
        StationFrom = new StationSharp { ESR = 320007, },
        StationTo = new StationSharp { ESR = 330009 },
        StationToBack = new StationSharp { ESR = 324008 },
        TypeFormation = FormationPlanEnum.Wagon, // план формування
        //Приклад використання станцій Через
        //ViaStations = new List<StationSharp>() { new StationSharp() { ESR = 411403 }
}
        //Приклад користувацького налаштування тарифної відстані для перевезення з
одним сегментом. Використання Повернення або станцій Через може збільшити кількість
сегментів. Для використання цієї властивості потрібно знати кількість сегментів та їх
коди ЖД з урахуванням всього маршруту.
        //TariffSegmentSettings = new List<TariffSegmentSettingsSharp>() { new
TariffSegmentSettingsSharp() { RwAdminId = 22, TariffLenght = 500, PostType =
PostsTypeEnum.Auto, TransportationKind = TransportationKindEnum.Auto } }
    }
};
var wagon = new WagonSharp //вагон
{
    Axis = 4,
    EquipmentWeight = 0,
    EscortCount = 0,
    Number = "0",
    Ownership = OwnershipTypeEnum.Common,
    PurposeWagonUsing = PurposeWagonUsing.Default,
    RwAdminId = 22,
    RsType = new RsTypeSharp { RollingStockId = 10 }, // код рухомого складу, 10 –
критий
    Substitution = 0, //подано в замін
};
var cargo = new CargoSharp //вантаж
{
    //CargoBulk = true,
    //CargoPerishable = true,
```

```
    //CargoPrecooling = true,
    DangerClass = 0,
    DangerSubClass = 0,
    EtsngCode = 11005,
    EtsngRwAdminId = 22,
    GngCode = 10011900,
    RealWeight = 60, //вага
    WeightMin = 0,

};
cons.Wagons.Add(wagon);
wagon.Cargoes.Add(cargo);
return _tmCalcExtV2.CalculateCons(cons);
```

2. Контейнерне перевезення

Базується на повагонному.

```
    Dispatch = new DispatchContainerLargeLoad() //змінився тип перевезення на
контейнерне

    RouteHeader = new RouteHeaderConsSharp
    {
        TypeFormation = FormationPlanEnum.LargeContainer, //змінився тип плану
формування на контейнерний
    }
```

Порівняно з по вагонним перевезенням в контейнерному перевезенні треба вкладати вантаж в контейнер, а контейнер в вагон.

```
var wagon = new WagonSharp
{
    Axis = 4,
    EquipmentWeight = 0,
    EscortCount = 0,
    Number = "0",
    Ownership = OwnershipTypeEnum.Common,
    PurposeWagonUsing = PurposeWagonUsing.Default,
    RwAdminId = 22,
    RsType = new RsTypeSharp { RollingStockId = 30 },//код вагону
    Substitution = 0,
};
var cont = new ContainerSharp
{
    //Length,
    //Number = "0",
    Ownership = OwnershipTypeEnum.Common,
    RealFreightCapacity = 66,
    RsType = new RsTypeSharp { RollingStockId = 720 },//Код контейнера
    RwAdminId = 22,
    TareWeight = 18
};
var cargo = new CargoSharp
{
    //CargoBulk = true,
    //CargoPerishable = true,
    //CargoPrecooling = true,
    DangerClass = 0,
    DangerSubClass = 0,
    EtsngCode = 11005,
    EtsngRwAdminId = 22,
    GngCode = 10011900,
    RealWeight = 60,
    WeightMin = 0,

};
cons.Wagons.Add(wagon);
wagon.Containers.Add(cont);
```

```
cont.Cargoes.Add(cargo);
```

3. Збірне перевезення

Базується на повагонному.

```
    Dispatch = new DispatchWagonCargoSetSharp () //змінився тип перевезення на збірне перевезення
```

Порівняно з по вагонним перевезенням в збірному перевезенні треба вкладати всі в один вагон. Сумарна вага вантажу не повинна перевищувати вантажопідйомність вагону. Перевезення обмежене одним вагоном.

```
var wagon = new WagonSharp
{
    Axis = 4,
    EquipmentWeight = 0,
    EscortCount = 0,
    Number = "0",
    Ownership = OwnershipTypeEnum.Common,
    PurposeWagonUsing = PurposeWagonUsing.Default,
    RwAdminId = 22,
    RsType = new RsTypeSharp { RollingStockId = 10 },
    Substitution = 0,
};
var cargo1 = new CargoSharp
{
    //CargoBulk = true,
    //CargoPerishable = true,
    //CargoPrecooling = true,
    DangerClass = 0,
    DangerSubClass = 0,
    EtsngCode = 11005,
    EtsngRwAdminId = 22,
    GngCode = 10011900,
    RealWeight = 20,
    WeightMin = 0,

};
var cargo2 = new CargoSharp
{
    //CargoBulk = true,
    //CargoPerishable = true,
    //CargoPrecooling = true,
    DangerClass = 0,
    DangerSubClass = 0,
    EtsngCode = 012008,
    EtsngRwAdminId = 22,
    GngCode = 10021000,
    RealWeight = 32,
    WeightMin = 0,

};
var cargo3 = new CargoSharp
{
    //CargoBulk = true,
    //CargoPerishable = true,
    //CargoPrecooling = true,
    DangerClass = 0,
    DangerSubClass = 0,
    EtsngCode = 013000,
    EtsngRwAdminId = 22,
    GngCode = 10041000,
    RealWeight = 12,
    WeightMin = 0,

};
cons.Wagons.Add(wagon);
wagon.Cargoes.Add(cargo1);
wagon.Cargoes.Add(cargo2);
wagon.Cargoes.Add(cargo3);
```

4. Розрахунок маршруту

```
RouteHeaderSharp header = new RouteHeaderSharp() //головний об’єкт маршруту
{
    StationFrom = new StationSharp()
    {
        ESR = 320007,
        RwAdminId = 22,
    },
    StationTo = new StationSharp()
    {
        ESR = 330009,
        RwAdminId = 22,
    },
    TypeFormation = FormationPlanEnum.Wagon //план формування
};
return _tmCalcExtV2.CalculateRoute(DateTime.Now, header);
```

5. Вивід результатів розрахунку

Повний приклад виводу результатів розрахунку аналогічно до виводу ТМКарти доступний в C# прикладі який поставляється з TMCalcExtV2 бібліотекою (LogBuilder.cs)

`ConsCalcResultSharp` містить результати розрахунку перевезення – загальний час перевезення, вартість та детальні результати по країнам які в свою чергу містять детальний опис по тарифам.

`RouteSharp` містить результат розрахунку маршруту – відстань, тарифна відстань, сегменти та детальний список станцій маршруту.