C# Marshaling - write bytes to struct and vice versa

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct MyStructure
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 101)]
    public char[] Name;
    public UInt16 Port;
    public byte Num;
    public byte Max;
    public UInt64 Version;
    public byte TOD;
    public byte Avg;
    public byte Flags;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
    public char[] Key;
}
public static MyStructure BytesToMyStructure(Byte[] bytes)
{
    MyStructure X = new();
    int size = Marshal.SizeOf(X);
    IntPtr ptr = Marshal.AllocHGlobal(size);
    Marshal.Copy(bytes, 0, ptr, size);
    X = (S.EnrollServer)Marshal.PtrToStructure(ptr, X.GetType());
    Marshal.FreeHGlobal(ptr);
    return X;
}
public static Byte[] MyStructureToBytes(MyStructure data)
{
    int size = Marshal.SizeOf(data);
    byte[] bytes = new byte[size];
    IntPtr ptr = Marshal.AllocHGlobal(size);
    Marshal.StructureToPtr(data, ptr, true);
    Marshal.Copy(ptr, bytes, 0, size);
    Marshal.FreeHGlobal(ptr);
    return bytes;
}
```

Generic method

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public class EnrollServer_Startup
{
    public UInt16 Port;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = NAME_LENGTH)]
    public char[] Name;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst =
```

```
DESCRIPTION_LENGTH)]
    public char[] Description;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst =
BANNER_URL_LENGTH)]
    public char[] BannerUrl;
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = KEY_LENGTH)]
    public char[] Key;
    public byte NumPlayers;
    public byte MaxNumPlayers;
    public UInt64 GameVersion; // major:8 minor:8 patch:16 changeset:32
    public byte TimeOfDay;
    public byte AverageFPS;
    public byte Flags; // PasswordProtected
    public byte Playstyle;
    public UInt16 SpawnAmounts;

    public static int SizeOf =>
Marshal.SizeOf(typeof(EnrollServer_Startup));
    public Byte[] ToBytes()
    {
        int size = Marshal.SizeOf(this);
        byte[] bytes = new byte[size];
        IntPtr ptr = Marshal.AllocHGlobal(size);
        Marshal.StructureToPtr(this, ptr, true);
        Marshal.Copy(ptr, bytes, 0, size);
        Marshal.FreeHGlobal(ptr);
        return bytes;
    }
}

public static T BytesToObject<T>(byte[] bytes)
{
    object X = Activator.CreateInstance(typeof(T))!;
    int size = Marshal.SizeOf(X);
    IntPtr ptr = Marshal.AllocHGlobal(size);
    Marshal.Copy(bytes, 0, ptr, size);
    X = (T)Marshal.PtrToStructure(ptr, X.GetType())!;
    Marshal.FreeHGlobal(ptr);
    return (T)X;
}
```