1. GetBlendSectionsByCallSheetId (int callSheetId)

入参： callSheetId

返回值：DTO(Id, CallSheetId, Interval, Quantity, BlendCategoryDescription, Density, IsNeedHaul, BlendFluidTypeName, UnitAbbreviation)

1. GetBlendSectionById(int blendSectionId)

入参： blendSectionId

返回值：DTO(Id, BlendAdditiveSections,BlendAdditiveSections.AdditiveAmountUnit, Quantity,BlendFluidTypeId, Density）

1. GetJobByUniqueId(string jobUniqueId)

入参： jobUniqueId

返回值：JobNumber

1. UpdateCallSheetStatus (int callSheetId, string eServiceEntityStatus)

入参： callSheetId, eServiceEntityStatus

返回值：

1. GetCallSheetByNumber (int callSheetNumber)

入参： callSheetNumber

返回值：DTO(Id,CompanyName, WellLocation, DownHoleLocation, RigCompany, RigNumber, JobTypeDescription, ExpectedTimeOnLocation, ConsultantPhone, ServicePoint, CallSheetNumber, CustomerId, ClientCompany)

1. GetCallCrewTime (int callSheetId)

入参： callSheetId

返回值：callCrewTime

1. CreateBinSection (int callSheetId, int binId)

入参： callSheetId, binId

返回值：

1. GetSecuredUserByApplicationAndUserName (string applicationName, string userName)

入参： applicationName, userName

返回值：DTO(Id,Group,Permissions)

1. GetJobPackageByCallSheetNumber (int callSheetNumber)

入参： callSheetNumber

返回值：DTO(Id)

存在的问题：

1. 所有接口的识别，应当以实体为核心。因为我们API都是以实体为核心的，而不是以方法为核心的。至少以哪个实体为核心？当前识别出是哪个实体，就用哪个实体。以后如果时境（Context）发生变化，也许所依附的实体也会发生变化。如BlendSection，现在只需要用BlendSection这个做为实体就可能。但是以后如果有需求同时提供CallSheet 和 ServiceReport中BlendSection，那么可能就要依附于CallSheet和ServiceReport了，或是构建出层次关系。这是后话。
2. 关于命名，不要怕麻烦，要遵从统一的命名规范，不能擅自简略。一个是为了可读性，方便其他人协作。别一个原因是，我们希望以后能尽量多的生成代码。规范的命名，可以有助于我们易于识别模式。错误的例子，GetCallSheetByNumber.
3. 在之前的实现中，因为是直接调用Library，在Online中直接写了一些应该在eService中的逻辑，如在Online中判断或赋值Callsheet的Status，混淆了系统边界，在识别API接口时，要把它扶正。接口以实体为核心，显性定义。

如 ：UpdateCallSheetStatus (int callSheetId, string eServiceEntityStatus)

eServiceEntityStatus 参数的存在直接产生了对eServiceEntityStatus这个enum的依赖，尽管你把它变成了string，但概念层面的依赖还是存在的。

显性定义这个方法应该是，CancelCallSheet(int callSheetNumber);

昨天我说在BlendSection的接口中使用继续使用CallSheetId, 是因为聚合关系的存在。大多数的时候还是应该用CallSheetNumber.

1. 在识别接口的过程中，你们做的太快了，没有深入考虑，不学习。我让你们先做BlendSection，就是给你们一个学习和体会的机会。然后一个一个实体地去识别。而从这个识别结果看，你是在根据代码去识别，看到一个MicroService的调用，就直接加一个接口。而没有从系统的角度考虑，这个接口应该是什么？比如： GetJobPackageByCallSheetNumber (int callSheetNumber)，你是研究了代码，但是没有发现逻辑中的这个调用是个无用的调用。因为RigJob中的JobNumber本身就是eService写的，根本不需要从eService重新读。