技术债（Technical Debt）

# 级别： 基础类型

1. AdditiveMethodType vs AdditiveBlendMethod

AdditiveMethodType is a reference data based on CommonType, was created in early version of eService. It is part of Product.BlendAdditiveSection, but not mandatory and there was no business logic depends on it.

When we build Bulk Plant Calculator, we dive in lower level business process and logic, the concept of AdditiveMethod becomes critical to Blend Recipe. Since existing AdditiveMethodType and also CommonType it replys on are defined in Operation business domain, it is hard to refactor it in short time. So we introduce an enum type AdditiveBlendMethod as the equivalent.

TODO action: Further analysis is needed to determine if we need to consolidate these two types as one.

1. MeasureUnit vs UnitOfMeasure

MeasureUnit is a static class used as enum type. UnitOfMeasure was defined in SBS R2 project to align the definition with AX customized UnitOfMeasure.

MeasureUnit is used in old logics. UnitOfMeasure is used in new logics mainly pricebook related.

TODO action: Consolidate two into one to be consistent.

Techinical Hint: Unit of Measure is a very important reference data in the system, it needs to be consolidated asap. Since this reference data is basically a global standard, static class should the right way to be. The structure of UnitOfMeasure is well defined, especially for conversion and composite Unit, it should be adopted.

# 级别：硬代码

BlendBreakDownCalculator. CalculateAdditiveBreakDownsAdditionMethodBwocOrBwow()

double baseVolumn = (baseTonnage / 1000) \* baseYield.Value;

Assumption: UOM for base blend of recipe is kg, UOM for blend yield is m3/t

# 级别: MDD 生成代码工具

MDD生成服务接口，应提供丰富的查询接口供前台开发员使用，这样可以避免前台通过逻辑去过滤数据，而且生成的代码中可以实现数据库查询语句级别的数据读取，从而大大进提高系统整体性能。

1. 按实体属性ID查询。

例如：RigJob中有一关联实体ServicePoint, 是多对一的关系。需要实现以下接口。

Collection<RigJob> RigJobService.GetRigJobsByServicePoint(ServicePoint servicePoint);

1. 按实体属性ID 集合进行查询，这是为了满足实体属性多选查询的需要。

例如：RigJob中有一关联实体ServicePoint, 是多对一的关系。需要实现以下接口。

Collection<RigJob> RigJobService.GetRigJobsByServicePoints(Collection<ServicePoint> servicePoints);

注：集合类型是用Collection，还是用List，或其他类型，需要统一下来，以避免造成前台开发员的错误使用。

1. 单属性实体更新。在目前实践中，如果一个实体的一个属性值变动，则需要前台读取主对象的原有值，然后将一个属性的新值更新，再调用服务进行更新。在网络web应用中，时效性可能会出问题。特别是在异步的实现中。所以需要实现单属性更新接口。

例如：RigJob中有一关联实体ServicePoint, 是多对一的关系。需要实现以下接口。

int UpdateRigJobServicePoint(RigJob rigJob, ServicePoint servicePoint);

注：返回int值，应定义确切的返回码，给出明确的错误信息。如原始版本过期、数据库更新错误等

# 级别： Asp .Net Core MVC 系统设计

在应用中我们要根据实际情况使用缓存来保存常用数据，以提高系统性能和反应速度。

如常用Reference Data。

但是，如果ReferenceData的数据在管理端被修改，需要及时将应用缓存中的数据进行更新。更新机制需要进行设计和实现。

# 级别：架构遵从

eService Online的架构在D:\SESI Projects\SanjelDocuments\Projects\D-Design\Architecture\eServiceOnline\_Architecture.vsdx中明确定义，但是在实现中没有遵从这个设计。

主要是对缓存的ReferenceData没有明确定义。还存在着越级调用的情况。

架构图已经更新，没有遵从架构定义的代码需要重构修正。

eServiceOnlineGateway 只是调用相应服务的接口，不应当存在任务逻辑。它汇集不同服务源的服务。负责解决接入问题。

级别：工具库

1. 打印服务库：文本框内容过长会造成程序崩溃。

在eServiceExpress的打印中，Header中的Comments 部分是设置在一个GridControl的Cell中打印的。如果它的内容过长，如超过360个字符，会直接导致分页问题，从而造成死循环，最终导致程序崩溃。

由于打印字符的字体是由CSS设置的，所以在有限空间中能打印多少个字符只有在渲染的时候才知道。上述360个字符只是测试时的假设。当测试数据改为都用字母G来代替其他字符，大约只能打250个字符。因为字母G在Express中的字体设置比其他字符的宽度要更大。

解决方案建议：在模版设计中对于固定尺寸的单元，应该有关于溢出处理的属性。对于文本的溢出处理，可以采取截断或缩小字符大小等选择。

级别：通讯架构

在eService原有的代码中，Offline的状态是根据调用服务时出现异常来判断的，在内存中并没有维护一个离线标志供应用程序共享。在内网一般不会出现问题，因为服务器会给出确定的反应。但是在外网中，也就是本地机联入互联网，但是没有启动VPN的时候，如果外网地址与内网地址相同，但是又不及时给内网调用及时反馈，应用程序只好等待超时才能继续执行。

目前只是猜测，需要进一步证实。

# 级别：架构

ApiBuilder 不能正确支持有版本的实体

例如: RigService, 它实现的接口都是非版本的，对于后台以id为参数的操作，都会产生一个操作对多个记录的错误。