|  |
| --- |
| vancl |
| VESB AIO |
| Manual |
|  |
| **程诚** |
| **2013/2/17** |

|  |
| --- |
| VESB AIO(all in one)功能设计详解，其中包括服务合并，服务统一，灾备方案等 |

目录

[说明 2](#_Toc433621640)

[名词解释 2](#_Toc433621641)

[什么是VESB AIO？ 2](#_Toc433621642)

[VESB AIO给我们带来了什么(diff in one)？ 2](#_Toc433621643)

[服务方法的自由组合（diff-services functions in one） 3](#_Toc433621644)

[流程控制 3](#_Toc433621645)

[调用方式的统一（diff-invoke in one） 4](#_Toc433621646)

[VESB AIO是怎么工作的？ 4](#_Toc433621647)

[AIO分布式系统构架 4](#_Toc433621648)

[SDS 5](#_Toc433621649)

[VESB管理后台维护SDS数据 6](#_Toc433621650)

[为什么需要不把SDS的功能添加到VESB Core？ 6](#_Toc433621651)

[方法寻址 6](#_Toc433621652)

[方法地址列表 7](#_Toc433621653)

[系统架构图 8](#_Toc433621654)

[基本网络图 9](#_Toc433621655)

[数据流图 10](#_Toc433621656)

[服务订阅数据流 10](#_Toc433621657)

[服务调用数据流 11](#_Toc433621658)

[存在的风险 12](#_Toc433621659)

[SDS的崩溃 12](#_Toc433621660)

[方法地址列表同步失败 12](#_Toc433621661)

[写在最后 12](#_Toc433621662)

# 说明

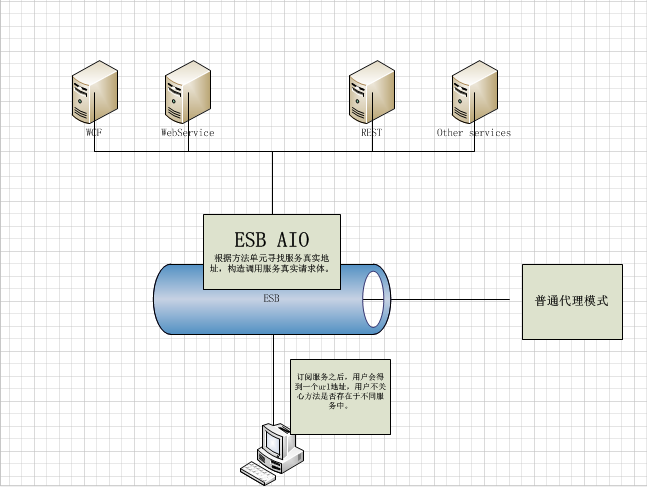
## 名词解释

**调用方**：也称为服务调用方或服务消费者，是VESB的主要参与者，他们是ESB中服务的使用者。

**发布方**：也成服务发布方或服务提供者，是VESB中服务提供者，他们将服务注册到VESB中。

# 什么是VESB AIO？

VESB AIO全称ESB All In One，是VESB的主要功能特性之一，主要实现了将服务以方法为最小单元，通过调用方的订阅，将这些方法单元组合到一起生成一个新的集合，通过一个url地址进行调用。实现了服务方法的自用组合订阅，统一方式调用。

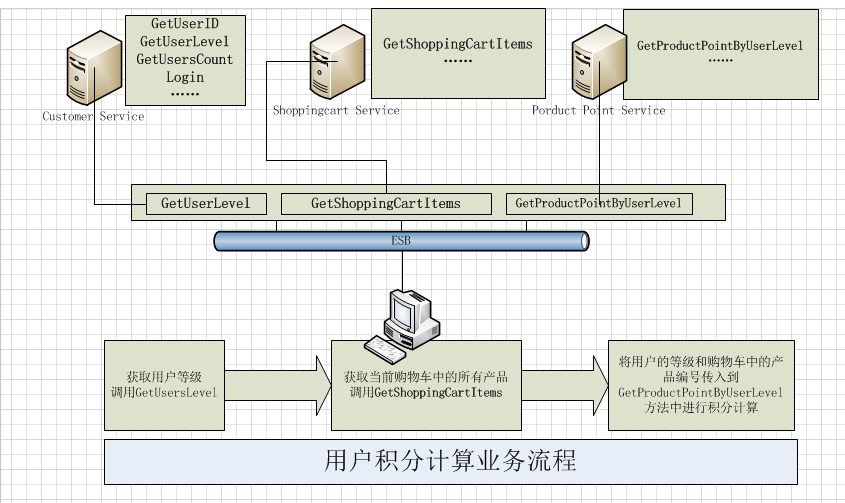


# VESB AIO给我们带来了什么(diff in one)？

有了AIO之后，体验全新的服务使用方式

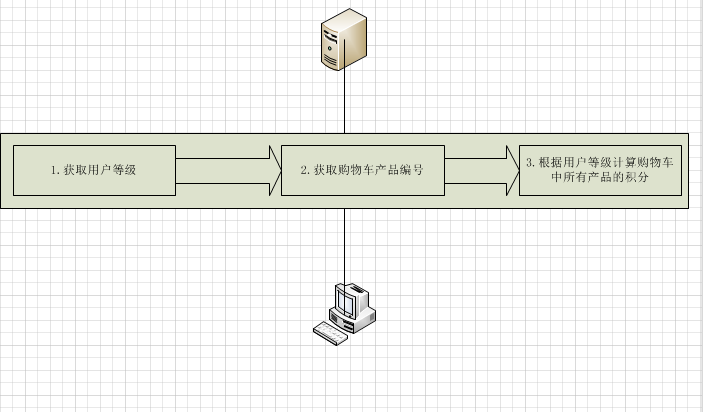
## 服务方法的自由组合（diff-services functions in one）

调用方不用再去按照固定的方式对多个服务地址进行维护，将不同服务中的方法按自己的方式进行统一管理（可以安逻辑分组，功能分组，业务分组等自由的组合模式）。



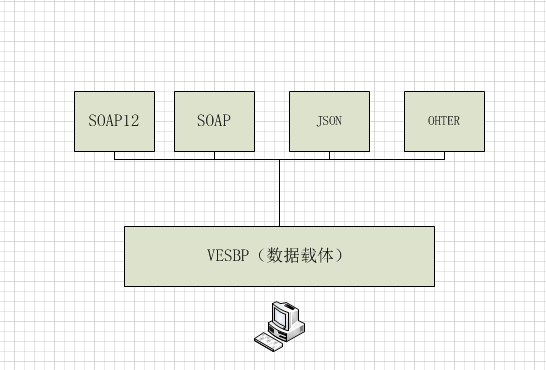
### 流程控制

当然有了方法的自由组合之后我们将在VESB中体验到全新的流程控制管理，在VESB后台中你将可以自由订阅所需要的流程，并自由设置流程中每个步骤的监控、进行流程化的测试等。



## 调用方式的统一（diff-invoke in one）

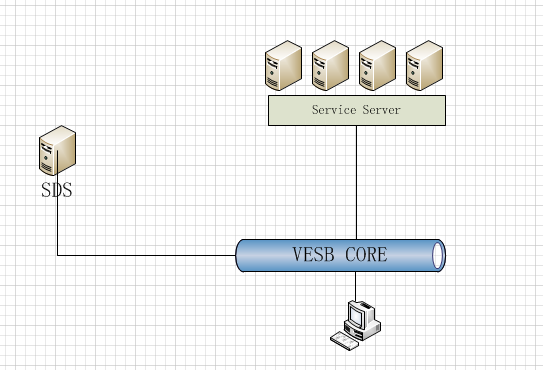
目前的服务种类多种多样，例如WEBSERVICES，WCF，REST等。 有了VESB AIO 之后我们将统一调用方式，采用统一的数据载体和统一的通讯方式，调用方将不会知道也不用关心自己调用的方法是存在于WCF、WEBSERVICES或者说是REST中，他只会关心数据是否正确，这也是调用方唯一关心的事情。



# VESB AIO是怎么工作的？

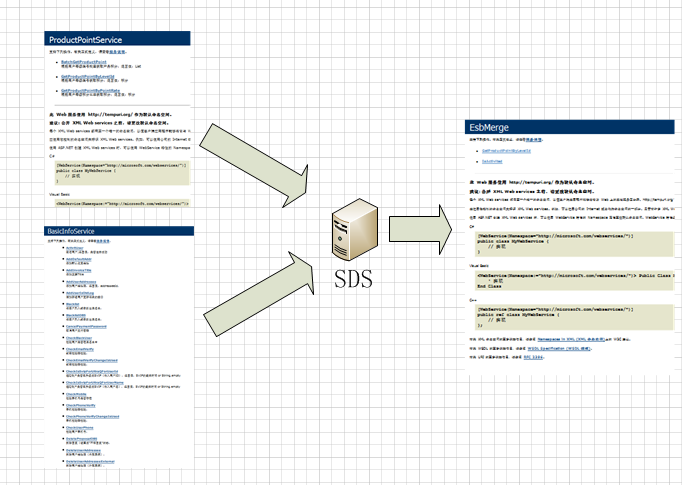
## AIO分布式系统构架

VESB AIO采用分布式的系统构架，要完成VESB AIO需要有其中最主要的两个系统服务VESB Core Server和Services Description Server(SDS)。



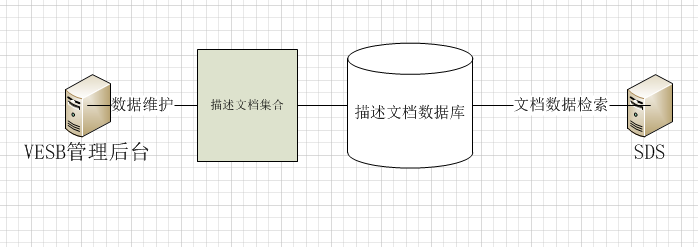
### SDS

SDS就是Services Description Server的缩写，它在VESB AIO中担任着重要的角色，主要功能是根据VESB Core的指令生成对应的服务描述文档并返回给VESB Core。开发人员应该知道，在我们使用Visual Studio编写服务方法调用之前，需要对服务进行引用，之后IDE将会根据服务返回的描述文档进行调用代码及配置文件的生成，之后开发人员只需要直接调用生成好的代码就可以了，但服务方法来自不同的服务，那么这个合并之后的描述文档到由谁来生成呢？这就是我们SDS需要做的事情了。



### VESB管理后台维护SDS数据

管理后台中将会对SDS的元数据进行维护，主要是根据合并的服务生成描述文档，之后将数据持久化（主要是数据库形式）。



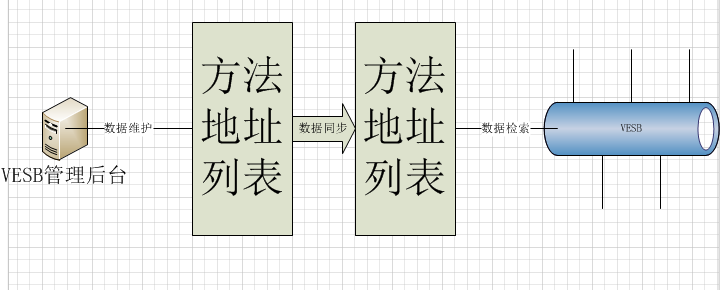
### 为什么需要不把SDS的功能添加到VESB Core？

## 方法寻址

在用户调用方法的时候，由于地址信息对客户端完全隐藏的，所以当VESB接收到请求之后首先会在方法地址列表中找到方法所在的服务的地址，之后VESB将会根据这个真实的服务地址去请求相应的方法。

### 方法地址列表

方法地址列表是以方法名 – url地址的键值对形式存在的。通过方法名称可以在其中找到方法所在的服务的url地址。VESB方法地址列表以文件的形式存在与VESB的程序中，VESB管理后台对方法地址列表的数据进行维护。



## 系统架构图

**系统主要参与者**

1. 服务调用方

2. 服务发布方

3. 平台管理员



## 基本网络图



## 数据流图

### 服务订阅数据流



### 服务调用数据流



# 存在的风险

就VESB AIO来说目前的风险分为，SDS的崩溃，方法地址列表同步失败。

**警报指数定义：**

最高级：属于最危险的情况，例如VESB核心部件异常甚至VESB系统崩溃等导致所有通信中断或异常的情况，直接影响到生产环境。

高级：属于危险的情况，部分影响VESB的通信或异常，直接影响生产环境。

中级：属于一般级别的警报，不影响生产环境。

低级：对VESB的正常使用不产生影响。

### SDS的崩溃

**造成的后果**：SDS崩溃之后将导致合并文档不能正常生成，直接导致引用服务时候的代码及配置自动生成失败，主要存在于开发阶段。

**警报指数**：中级

### 方法地址列表同步失败

**造成的后果：**同步失败之后，新添加的数据不能更新到VESB服务中，导致新的合并方法的调用不能正常完成。

**警报指数：**高级

# 写在最后