企业集成模式（EIP）

一、简介

1. 集成解决方案必须应对的挑战

- 网络不可用

  ```
  * 不得不通过网络传输数据
  * single or distribute
  * 网络间传输数据时常延迟、阻塞
  ```

- 网络延迟

  ```
  * 对比本地方法调用，通过网络传递数据更慢
  * 使用single app方法设计distribute app可能存在潜在的问题
  ```

- 不同的应用程序

  ```
  * 系统间传输信息受编程语言、操作平台、数据格式影响
  * 集成方法需要接入不同技术
  ```

- 不可避免的改变

  ```
  * 应用随时都在变化
  * 应用变化导致雪崩效应
  * 集成方案需要最小化依赖（系统间）、松散的连接（系统间）
  ```

2. 通常的解决方法

- 文件传输

  ```
  * 某些应用写文件，另外的应用读文件
  * 系统间需要就文件名、文件位置、文件格式、何时读写、谁可以删除达成共识
  ```

- 共享数据库

  ```
  * 多个应用共享相同的数据库定义、共用一个物理数据库
  * 没有多个数据存储、系统间没有数据需要传输
  ```

- 远程(过程)调用

  ```
  * 一个应用暴露一些能力，并由其它应用远程调用
  * 调用时实时、同步的
  ```

- 消息

* 某些应用发布消息到公共消息渠道
* 其它应用从公共消息渠道读取消息（少量延迟）
* 应用间必须就消息格式达成共识
* 异步通信

* 通常应用（多个分布式应用的集合）需要集成多种方案，各系统根据自身特性选择最佳方案

3. 什么是消息?

* 电话系统
* 同步通信：呼叫方与接听方需要都在线

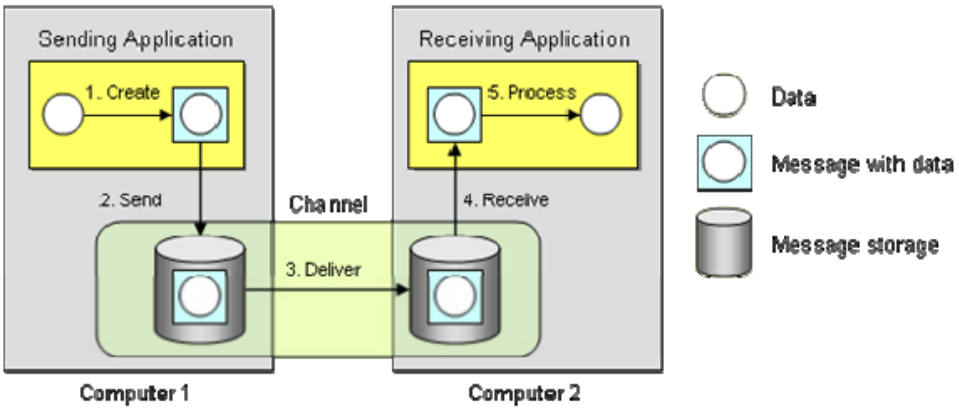
* 邮件系统
* 异步通信：发送方、接收方不必都在线

* 允许高速、异步、程序到程序可靠的信息传输
* 程序间通过发送数据包（消息）通信
* 渠道（channel）
  > 可以视为队列，一种逻辑路由，连接应用、传输消息
  > 类似于集合/数组的能力，但是提供了多应用间的共享和消息使用
* sender and producer
  > sender，通过向渠道写入消息实现发送消息
  > producer，通过读取（删除）渠道消息实现接收消息
* 消息本身是一个有序的数据结构，比如string、byte array、record、object
  > 可简单视为数据
  > 可视为即将在接收端执行的命令描述
  > 可视为发送端发生的事件
  > 消息包含两部分，header and body
  > header ：消息元信息，谁发的、发到哪里...消息系统使用，而消息使用方应用忽略
  > body ：需要传递的数据，消息系统忽略，而消息使用方使用
* 异步消息架构

4. 什么是消息系统?

* 消息能力
  > 由消息系统/面向消息中间件(MOM)提供
* 消息系统管理消息
  > 与数据库管理数据持久化一样
* 系统管理员配置channels，构建应用间的通信，消息系统随后协调和管理消息的发送和接收，
  消息系统是以可靠的方式将消息由发送端搬运到接收端
* 邮件系统投递消息依赖于网络的可用性和应用的实际状态
* 消息系统需要克服邮件系统的约束，通过重复尝试直至消息发送成功，理想环境下一次成功，但是常常不理想

* 消息投递的`五步`
  > create
    sender 创建消息并填充数据
  > send
    sender 添加消息到 channel
  > deliver
    消息系统将消息由 sender 的计算机移动到 receiver 的计算机，确保接收方可使用消息
  > receive
    receiver 从 channel 读取消息
  > process
    receiver 从消息中提取数据



* send and forget
  > （step 2）发送应用发送消息到渠道后无需等待消息发送成功并被接收方接收到，完全信任消息系统
* store and forward
  > （step 2）发送应用将消息发送后，消息系统将消息存储于发送端的内存或者磁盘上；（step 3）而后消息系统
    将消息由发送端的计算机投递到接收端的计算机上并存储于接收端的计算机上，store and forward 可能会重复
    多次，直至接收端收到消息

* create、send、receive、process步骤看书不必要的开销，为什么不简单的将消息投递给接收方？
  > 将数据包装为消息，交由消息系统存储，并委托消息系统投递数据
  > 数据被包装为原子消息，投递可以被重复投递，直至投递成功，接收方被确保能够获取到一份消息的副本

5. 为什么使用消息系统?

* 比`文件传输`直接
* 比`共享数据库`更好的封装性
* 比`远程调用`更可靠

* 消息的益处
  > remote communication(远程通信)
  > platform/language integration(平台/语言集成)
  > asynchronous communication(异步通信)
  > variable timing(可变时序)
  > throttling(可调节)
  > reliable communication(通信可靠)
  > disconnected operation(失联操作)
  > mediation(调度)
  > thread management(线程管理)

6. 异步消息的挑战

* 挑战
  > complex programming model(复杂的编程模型)
  > sequence issue(序列问题)
  > synchronous scenarios(同步方案)
  > performance(性能)
  > limited platform support(平台支持受限)
  > vendor lock-in(供应商锁定)

7. 关于异步的思考

8. 分布式应用 VS 集成

9. 商业消息系统