基于消息的SOA企业解决方案

第六章 基于消息中间件(ESB)的企业SOA案例分析

本章内容

1. 基于ESB的企业SOA项目需求分析
2. 基于ESB的企业SOA项目方案设计
3. 航空公司SOA案例分析
4. 制造企业SOA案例分析

本章内容

1. 基于ESB的企业SOA项目需求分析
2. 基于ESB的企业SOA项目方案设计
3. 航空公司SOA案例分析
4. 制造企业SOA案例分析

基于ESB的企业SOA项目需求分析

- 需求分析阶段是梳理项目中相关功能需求和非功能需求的重要步骤，它是整个项目成败的关键。在这个阶段我们将从企业业务需求出发，梳理端到端的跨系统业务流程；基于业务流程，依据科学的方法论进行服务鉴别；由服务列表出发，梳理服务的消费和提供关系；然后根据 SOA 的最佳实践，定义服务的接口，包括服务的 Schema 描述，字段的类型，编码的规则；依据服务的消费—提供关系，梳理 ESB 中的服务映射和转换规则和策略。

基于ESB的企业SOA项目需求分析-功能性需求

- 梳理出要被集成的系统的名称，个数。
- 针对每个系统而言，要了解：
  - 对外接口是向外调用（OutBound），被别人调用（Inbound），还是二者都有；
  - 接口的实时性要求，实时的、批量的，还是二者皆有
  - 接口的调用方式，同步、异步，还是二者皆有？
  - 应用系统所运行的操作系统平台。
  - 应用系统本身的编程语言？C/C++, Java…..
  - 这些系统现有接口的情况，是否已经可以提供对外接口，接口的方式是什么，包括接口的通讯协议是什么，HTTP/MQ/Socket/ 其它？接口的数据格式是什么，XML/ 自定义格式 / 其他行业标准格式？接口的编程语言是什么，Java/C/C++？如果本身不能提供接口，那么要做接口开发时有什么要求或限制条件？

基于ESB的企业SOA项目需求分析-功能性需求

- 针对每个系统而言，要了解：
  - 这些应用后台数据库的情况，数据库能否直接访问？
  - 每个应用跟其他应用交换数据时，源数据格式和目的数据格式，比如从文本格式转换为 XML 格式？
  - 交易特征：哪些处理要采用两阶段提交；是否需要多个消息组成一个交易；是否要保证消息之间的处理顺序；
  - 适配器的情况：对于一些特殊系统，是否已经具备现成的适配器；适配器是单向的还是双向的；
  - 消息通信的模式：是 Send and Forget、Request/Reply 还是 Pub/Sub?

基于ESB的企业SOA项目需求分析-非功能性需求

- ESB平台的扩展性和高可用性需求，包括 HA 和集群等；
- ESB平台的性能需求，主要包括系统间数据交换的频率，要交换的数据的大小（ 消息大小将直接对效率造成影响）；峰值时候对 ESB 数据吞吐量、响应时间的要求等；
- 哪些交易要保证数据传输的高可靠性；
- ESB 平台的可管理性需求，如服务的生命周期管理，ESB 平台的维护和管理；如果企业已经设立了 SOA 管控方面的规范，那么要遵从规范的制约，比如要考虑是否有规定的命名规则，企业是否有企业级的数据规范和底层通讯协议的规范等；
- 安全性方面的要求：是否采用 SSL 传输加密，是否对消息进行加密/解密处理等；
- 错误处理和日志以及平台本身的运行监控等方面的要求等

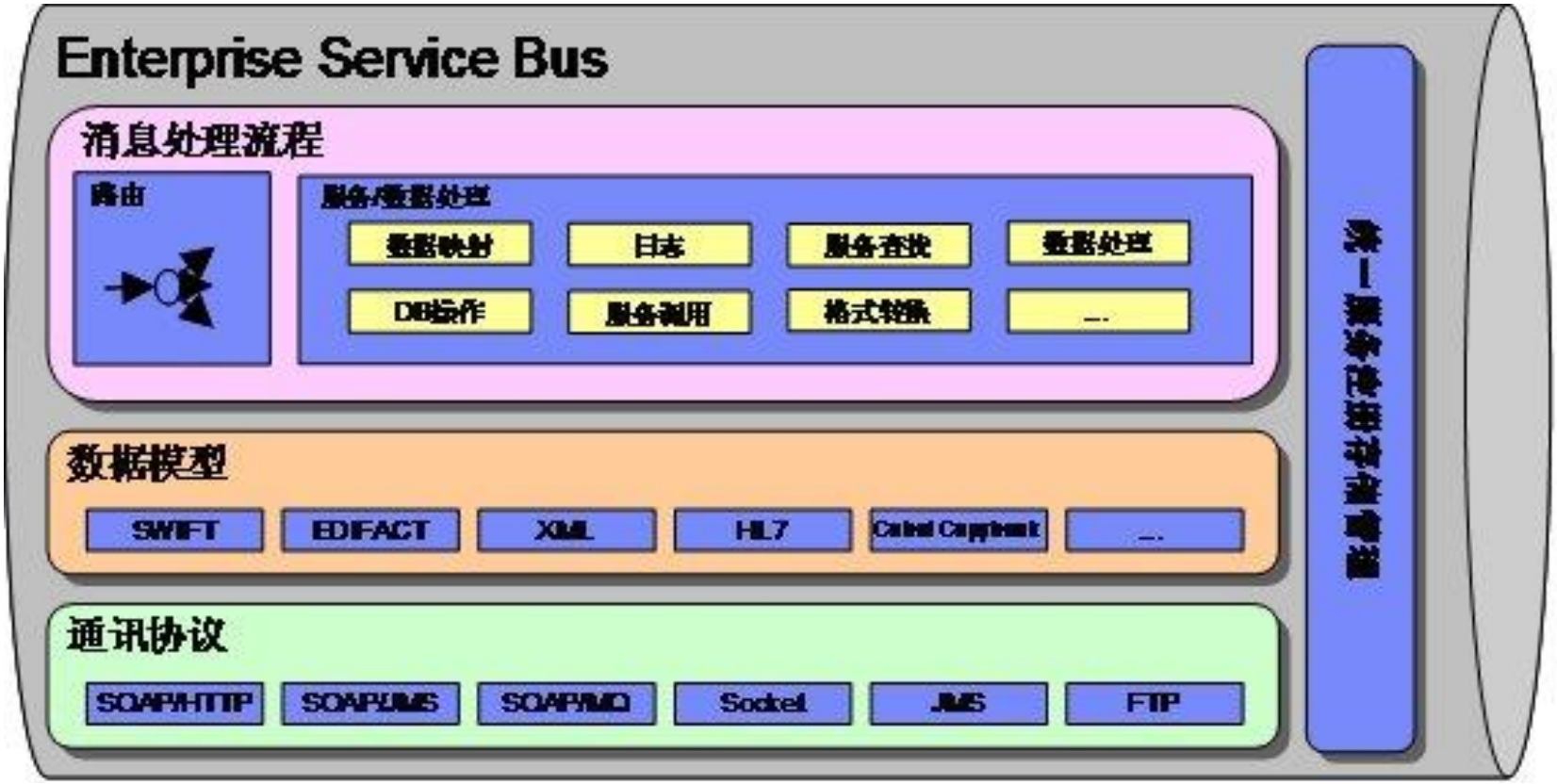
本章内容

1. 基于ESB的企业SOA项目需求分析
2. 基于ESB的企业SOA项目方案设计
3. 航空公司SOA案例分析
4. 制造企业SOA案例分析

基于ESB的企业SOA项目方案设计

- ESB涉及IT应用环境分析，定义ESB与相关应用的接口模式
- ESB架构概要设计，并定义架构原则；
- ESB相关产品选择，包括与外围系统的适配器选择和ESB产品选择；
- ESB组件模型设计，分解ESB的相关模块，满足SOA的分离关注点等架构原则；
- ESB运作模型设计，满足平台的非功能性需求；
- ESB平台的服务流设计，涉及路由、转换和映射等；
- ESB的同步、异步或者发布/订阅模式设计；
- ESB平台的接入渠道和数据接口设计，包括 XML/JMS、SOAP/HTTP、EDI/MQ 等；
- ESB相关的适配器设计，包括技术适配器或者自开发的适配器；
- ESB平台的容错和重试机制设计，包括日志等的统一管理

采用ESB整合的高层架构设计举例

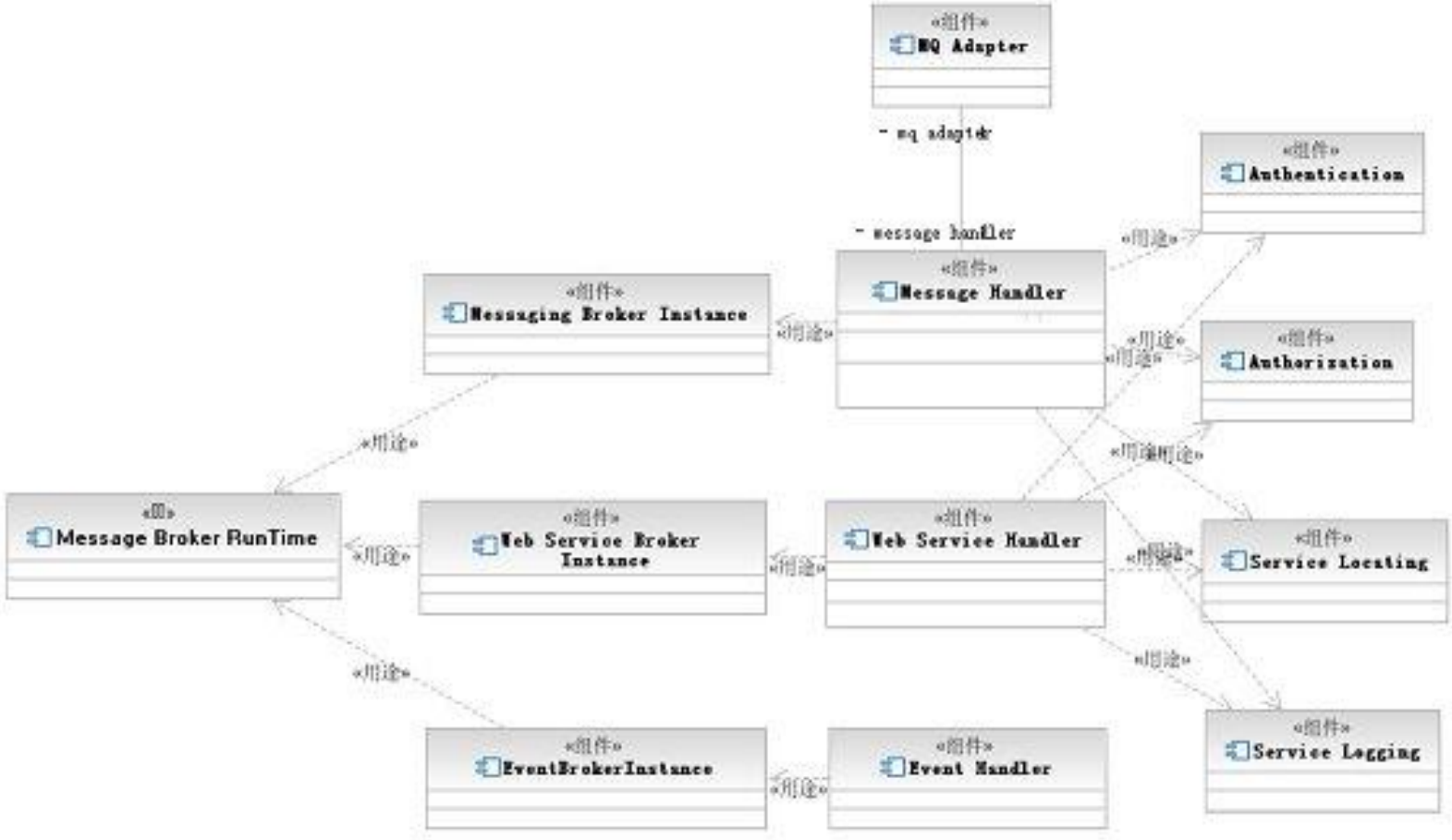


采用ESB整合的高层架构设计举例

- ESB架构设计时主要要考虑通讯协议接入和转换、数据接入和转换、数据处理流程以及服务的注册和管理等方面的内容。
  - 通讯协议接入和转换是指对各种被集成的应用系统的通讯协议的支持和转换能力，例如 HTTP、JMS、Socket、FTP 等；
  - 数据接入和转换是指对各种被集成的应用系统提供的数据格式的支持和转换能力，例如 XML、SOAP、自定义格式以及符合某些行业标准的专有格式（SWIFT、EDI、HL7 等）；
  - 数据处理流程是指路由、格式转换、数据库读写等对数据的各种处理；
  - 统一服务注册存储管理是指对服务的注册、发布、查询，以及对运营时服务的管控，并且提供服务运营状态的统计分析数据。

ESB的组件模型

- ESB组件模型示例



ESB组件模型

- 各组件的功能
  - Message Broker Runtime组件提供消息路由、格式转换、消息日志等操作的运行时环境。该运行环境由IBM Message Broker提供
  - Messaging Broker Instance组件是处理基于MQ消息业务请求的容器。它是作为一个Broker实例运行在 Message Broker Runtime 上的。该实例提供了MQ消息的业务请求处理器、服务日志、服务定位等功能的运行容器；
  - Web Service Broker Instance 组件是处理基于 Web Service 的业务请求的容器，它是作为一个 Broker 实例运行在 Message Broker Runtime 上的。该实例提供了 Web Service 的业务请求处理、服务日志、以及服务定位等功能。
  - Event Broker Instance组件是平台内部处理Pub/Sub事件的容器。它是作为一个 Broker 实例运行在 Message Broker Runtime 上的。该容器提供了 Event Handler 组件的运行环境，将基于 MQ/JMS 的事件分发到不同平台组件的目标队列上。

ESB组件模型

- 各组件的功能
  - Message Handler组件是处理基于 MQ 消息的业务请求，包括消息解析、格式转换，服务鉴权与认证、服务路由、服务日志等功能。Message Handler 组件处理 MQ 消息的典型流程如下：
    - 首先对 MQ 消息进行解析，对解析后的业务请求进行分析，之后通过 Authentication 与 Authorization 组件判断该请求者的业务请求是否可以进行后续处理；
    - 通过Service Locating组件对该业务请求进行服务定位与路由
    - 将基于MQ的业务请求消息转换成Web Service的业务请求消息
    - 通过 Service Logging 组件对整个业务请求进行日志记录；
    - 返回业务请求处理结果给业务发起者，如果失败，返回错误消息。

ESB组件模型

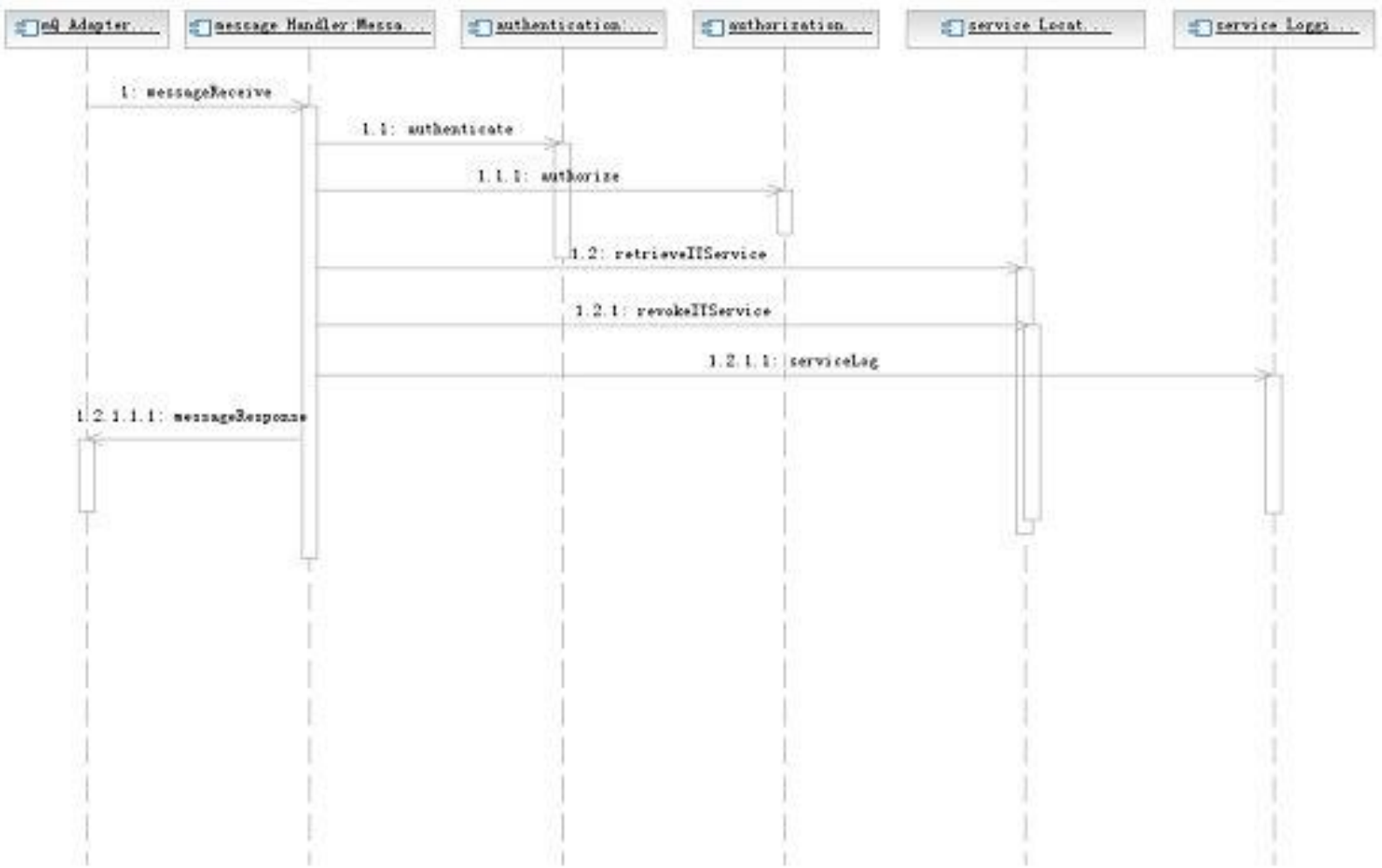
- 各组件的功能
  - Web Service Handler组件是处理基于Web Service的业务请求，与Message Handler组件功能类似，也包括消息解析、格式转换，服务鉴权与认证、服务路由、服务日志等功能。Web Service Handler组件处理 Web Service 请求的典型流程如下：
    - 首先对 Web Service 请求消息进行解析，对解析后的业务请求进行分析，之后通过 Authentication 与 Authorization 组件判断该请求者的业务请求是否可以进行后续处理；
    - 通过Service Locating组件对该业务请求进行服务定位与路由
    - 通过 Service Logging 组件对整个业务请求进行日志记录；
    - 返回业务请求处理结果给业务发起者，如果失败，返回错误消息。

ESB组件模型

- 各组件的功能
  - Event Handler组件实现对 Pub/Sub 的处理。
  - Service Locating组件负责根据业务请求定位具体的服务提供者。Service Locating 通过对服务目录的查询选择适合的服务进行后续的调用，该查询工作可以通过实时的服务目录查询获得结果。
  - Service Logging 组件负责记录整个业务请求处理过程中的情况，该组件的实现可以通过文件或者数据库的方式。
  - Authentication组件负责对业务请求者进行鉴权，判断该业务请求者是否可以访问平台服务，该鉴权的工作在企业服务总线的外部进行，Authentication组件只是调用外部功能完成。
  - Authorization组件判断业务请求者是否具备访问某特定服务的权限，该验证权限的工作在企业服务总线的外部进行，Authorization组件只是调用外部功能完成

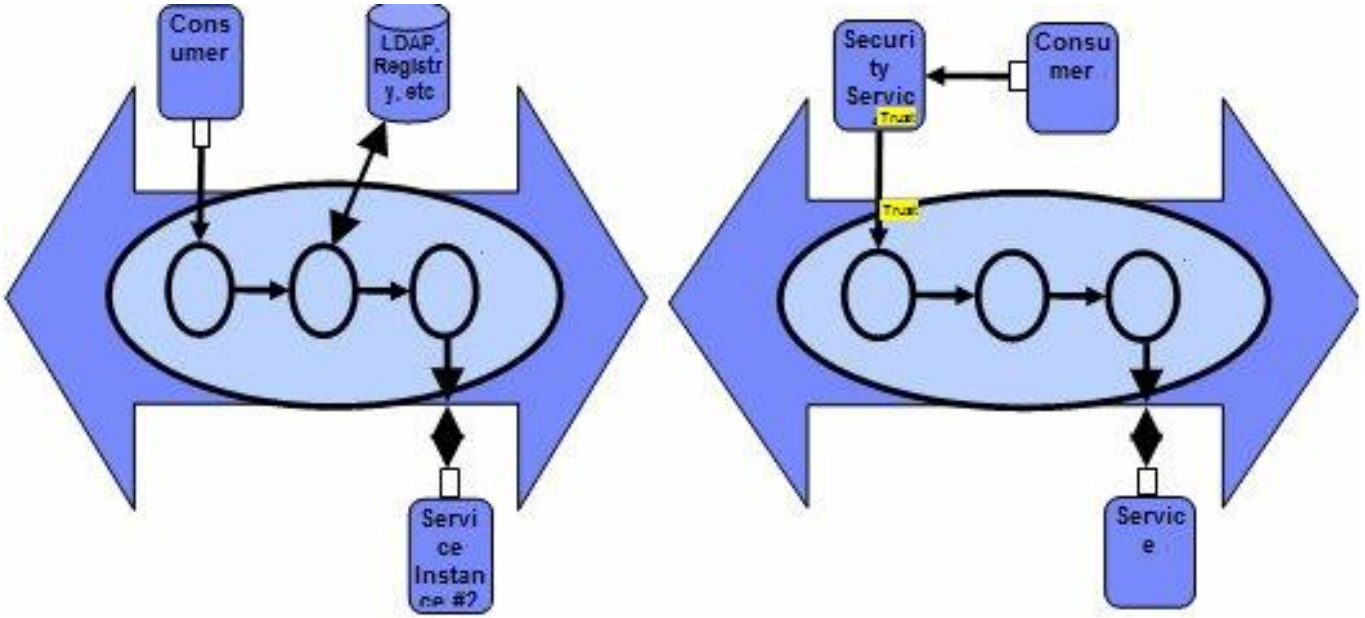
ESB组件的交互

- 处理基于MQ消息输入时的组件交互举例



ESB的安全性考虑

- 对于ESB的安全性考虑，主要有两种方式：
  - 通过 ESB 内部的 Mediation 节点来进行服务请求者的认证 / 授权；
  - 调用一个外部服务进行服务请求者的认证 / 授权



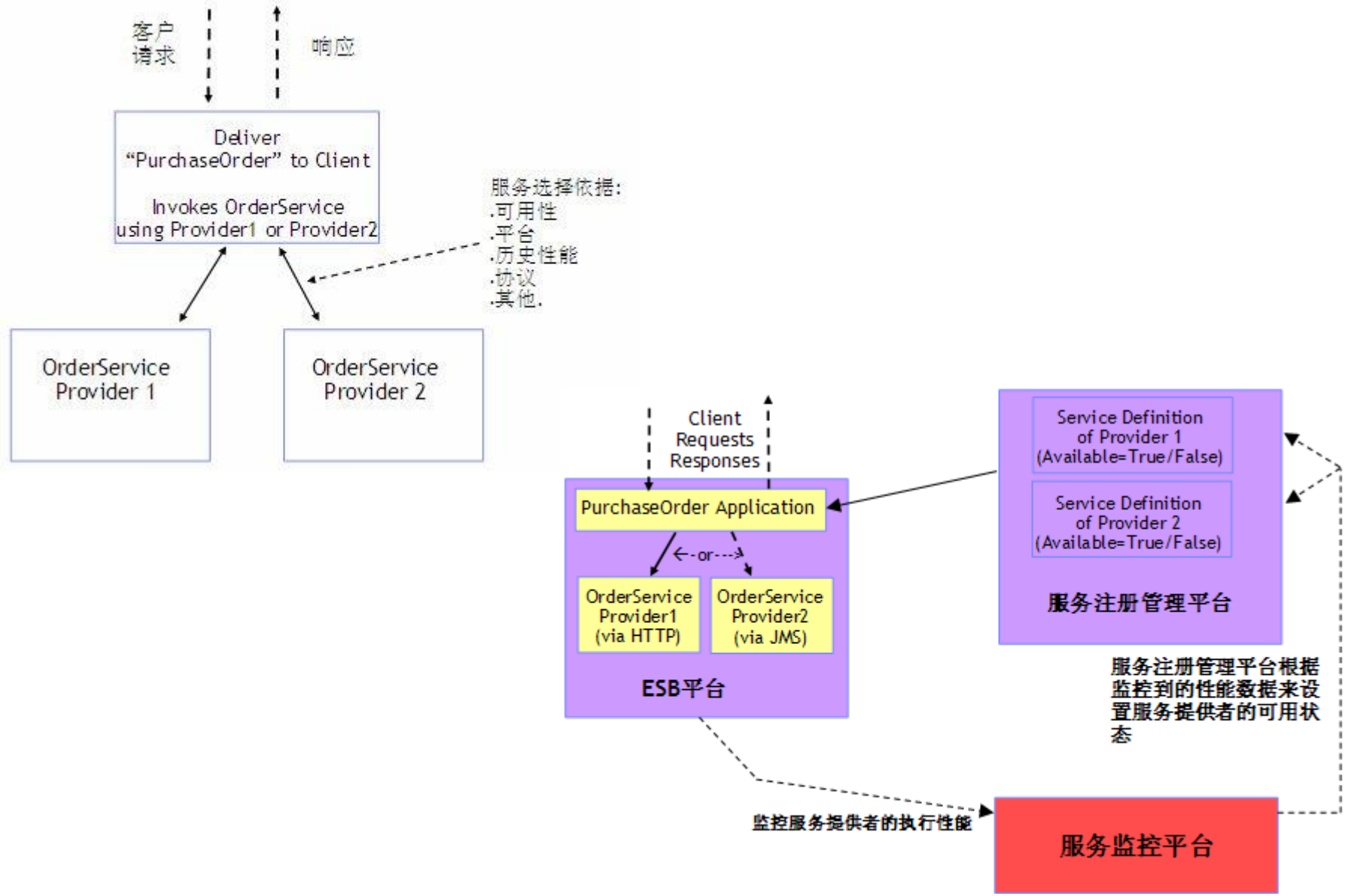
运行在 ESB 平台上的服务的管理和监控

- 在尽可能早的情况下考虑服务注册中心的建设。
- 服务注册中心是一个企业范围内的服务信息的存储库，该存储库存储了企业中注册的服务和服务相关的信息，它的主要功能包括：
  - 采用集中的方式来管理服务相关的信息，为服务元数据同时提供“注册中心”功能，允许用户存储、管理和查询包含服务描述的服务元数据构件；
  - 提供服务的治理功能，实现整个服务生命周期的管理
  - 提供服务间依赖、包容关系的管理；
  - 提供分类和版本控制等功能；
  - 提供服务发现和通知等能力等。

运行在 ESB 平台上的服务的管理和监控

- 服务不仅具有特定的功能，还应该满足一些诸如性能、可用性、安全性等 QoS 指标。服务响应的快慢、什么时间可用、可以被谁调用、在某个时间段里能被调用多少次、哪些事件要记录日志，这些都是服务管理要考虑的问题。
- 通过服务注册中 心和服务监控平台的有机配合，我们可以根据服务的响应时间、服务可用与否等策略来实现对服务的动态访问。。

服务监控平台对ESB的影响



ESB方案设计的最佳实践

- 确定标准的使用：使用与否、使用到什么程度；
- 确定在 ESB 上实现的业务逻辑：ESB 是一个服务路由和转换中心，而不是一个应用服务器，因此它并不能取代应有服务器。复杂的消息解析和转换相比简单的路由操作所需消耗的成本要高的多，因此在 ESB 上应该主要考虑路由、格式转换、服务调用等问题，而对于数据本身的处理应该交给相应的应用来完成；
- 确定消息格式：从标准化的角度而言，XML 当然是首选，但是从解析 / 处理性能、行业标准以及对现有应用的最大兼容性的角度而言，可能会采用某些特定格式，例如EDI、SWITF、平文本或者自定义格式等；

ESB方案设计的最佳实践

- 区别消息头和消息体：把数据的 Meta-data，例如：安全相关的信息、日志的等级、请求端的标识等放在消息头中，而不要放在消息体中。这样可以很容易地改变其内容及其对其的处理逻辑。在 ESB 中只处理消息头，避免对消息体的解析；
- 设计时参考 ESB 相关的成熟 Patterns；
- 使用服务注册库：如需要服务 Endpoint 的查找：推荐从服务注册中心进行查找，这样的好处在于：服务提供者可以容易地发布新的服务，服务提供者和 ESB 之间的耦合度可以更低，通过关于服务本身的 Metadata 来进行服务的查找和路由；
- 注重性能和高可用性的考虑；

ESB方案设计的最佳实践

- 在必须的情况下考虑交易完整性；
- 适配器的采用：应用系统通过适配器实现与 ESB 的双向交互，适配器主要分为技术适配器、应用适配器两种，适配器的物理部署可以与 EIS 部署在一起、或者与 ESB 部署在一起，也可以单独部署，在适配器设计时要考虑通信协议和消息格式两个方面；
- 多 ESB 的设计：ESB 也是一个逻辑的组件，在一个企业里可能需要多个 ESB，例如：企业内部 ESB 连接企业内部各个系统，外部 ESB 实现企业与合作伙伴等的外部连接；再如：企业内部可能存在若干个部门级 ESB 和一个全企业 ESB；

ESB方案设计的最佳实践

- 确定服务版本控制策略；
- 确定端到端的 QoS 准则；
- 注重安全性；
- 确定IT部门ESB平台的负责主体，长期的投入

ESB的开发和测试

- 在 ESB 开发和测试阶段要完成的工作主要包括：
  - 基于 eclipse 工具的模型驱动的快速开发；
  - ESB 集成流程的开发；
  - ESB 路由、消息处理逻辑的开发；
  - ESB 数据映射和转换的开发；
  - ESB 外围适配器的开发和配置；
  - 单元测试：基于模块的测试，包括适配器的测试，路由的测试，BO 的测试等；
  - 集成测试：ESB 与其他服务提供者和服务消费者的集成测试，重点关注服务接口；
  - ESB 平台的性能测试以及系统测试，即整个 ESB 涉及到的端到端业务场景的测试等。

本章内容

1. 基于ESB的企业SOA项目需求分析
2. 基于ESB的企业SOA项目方案设计
3. 航空公司SOA案例分析
4. 制造企业SOA案例分析

SOI案例分析

- 以一个经过简化的实际案例为例，介绍面向服务的企业集成的基本步骤，从业务分析，到服务建模，到架构设计，到系统开发的整个生命周期。所涉及到的主要技术也被穿插在各个步骤中进行详细的讲解。

案例背景

- 某航空公司的IT系统已有好几十年的历史。该航空公司的主要的业务系统构建于上世纪七八十年代，以IBM的主机系统为主 — 包括运行于TPF上的订票系统，和运行在IMS上的航班调度系统等。在这些核心系统周围也不乏基于Unix的非核心作业系统，和基于.Net的简单应用。这些形形色色的应用，有的用汇编或COBOL编写，运行于主机和IMS之上，有的以PRO*C编写，运行在Unix和Oracle上；这些应用虽然以基于主机终端的界面，但是基于Web和GUI的应用也为数众多。

案例背景

- 近年来，该公司在企业集成方面也是煞费苦心一已经在几个主要的核心系统之间构建了用于信息集成的信息HUB(information HUB)，其他应用间也有不少点到点的集成。尽管这些企业集成技术在一定程度上增进了系统间的信息共享，但是面对如此异构的系统，技术人员依然觉得企业集成困难重重：
  - 因为大部分核心应用构建在主机之上，所以Information Hub是基于主机技术开发，很难被开放系统使用；
  - Information Hub对Event支持不强，被集成的系统间的事件以点到点流转为主，被集成系统间耦合性强；
  - 牵扯到多个系统间的业务协作以硬编码为主，将业务活动自动化的成本高，周期长，被开发的业务活动模块重用性差；

案例背景

- 为了解决这些企业集成中的问题，该公司决定以Ramp Control系统为例探索一条以服务为中心的企业集成道路。
- 以Ramp Control系统中的Ramp Coordination流程为例说明如何用以服务为中心的企业集成技术一步步解决该公司IT技术人员面临的企业集成问题。

业务环境分析

- 在航空业中，Ramp Coordination是指飞机从降落到起飞过程中所需要进行的各种业务活动的协调过程。通常每个航班都有一个人负责Ramp Coordination，这人通常称为Ramp Coordinator。由Ramp Coordinator协调的业务活动有，检查机位环境是否安全，卸货，装货，补充燃料等。

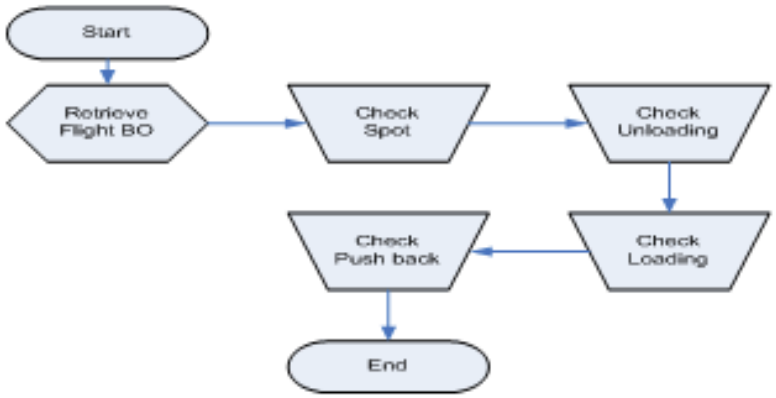


图 2: Ramp Coordination 流程图

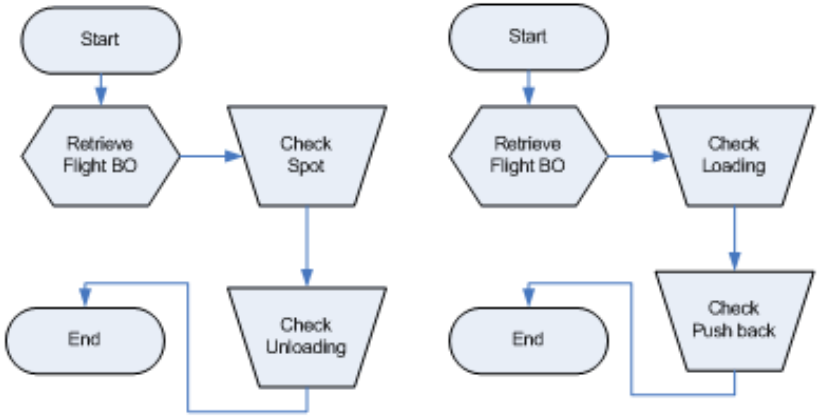


图 3：Arrival Only 和 Departure Only 的航班

业务环境分析

- 如此形形种种的流程之间共享着一个业务活动的集合，多种类型的流程都是这些业务活动的不同组装方式。
- 面向服务企业集成中流程服务就是通过将这些流程间共享的业务活动抽象为可重用的服务，并通过流程服务提供的流程编排的能力将它们组成各种大同小异的流程类型，来降低流程集成成本，加快流程集成开发效率的。以服务为中心的企业集成通过服务建模过程发现这些可重用的服务，并通过流程模型将这些服务组装在一起。

服务建模

- 使用组件业务建模(Component Business Model)和面向服务的建模和架构(Service-Oriented Model and Architecture)两种方法学建立业务的组件模型，服务模型和流程模型。
- 服务模型是服务建模的主要结果。

服务建模

- 两个业务组件：

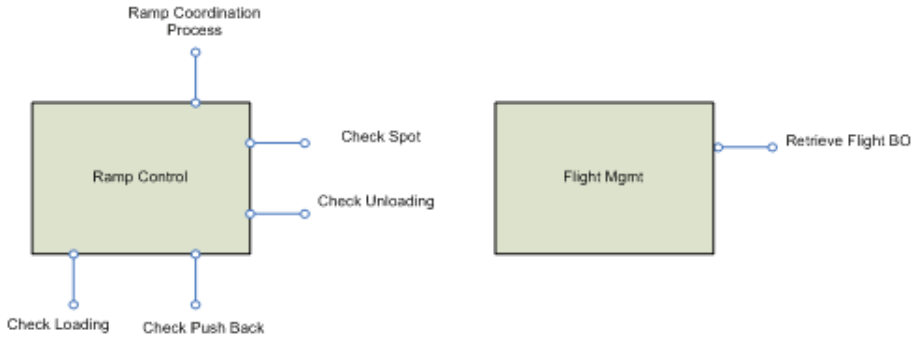


图 4. Ramp Coordination 相关的服务模型

  - Ramp Control：负责Ramp Control相关各种业务活动
    - 服务Ramp Coordination，用于Ramp Coordination流程的编排
    - 服务Check Spot，用于检测机位安全信息
    - 服务Check Unloading，用于检查卸货状况
    - 服务Check Loading，用于检查装货状况
    - 服务Check Push Back，用于检查关门动作
  - Flight Management：负责航班相关信息的管理，包括航班日程，乘客信息等
    - 服务Retrieve Flight BO，用于提取和航班相关的数据信息

服务建模

- 在服务建模确定系统相关的服务输出后，还需要确定服务在当前环境下的实现方式。
  - Retrieve Flight BO被实现为信息服务，Ramp Coordination被实现为流程服务，通过BPEL4WS方式实现；其他四个服务都是Staff Service。
  - 需要注意的是，因为环境的不同，和随着系统的演化，我们可能会改变服务的实现方式，比如Check Push Back现在通过Staff Service即人工服务实现。将来随着自动化程度的增强，Check Push Back完全可能通过自动化的系统实现。到那时，我们只需重新实现这个服务，而无需改变整个流程。这是服务的可替换性的一个典型实例。

IT环境分析

- IT环境分析是调查现有应用技术特点的重要手段。IT环境分析主要用于调查现有应用，为决定服务模型中服务的实现方式提供技术依据。同时，它也是架构设计的重要依据。
- 针对本案例，在构建Ramp Control系统之前，该航空公司已经有大量IT系统。现有IT环境调研描绘了和Ramp Control相关的IT系统的状况，包括周围应用和应用提供的接口，这些应用和Ramp Control交互的类型和数据格式。

IT环境分析

- Ramp Coordination流程需要四种类型的外围应用交互：
  - 从乘务人员管理系统提取航班乘务员的信息；
  - 从订票系统中提取乘客信息；
  - 从机务人员管理系统中提取机务人员信息；
  - 接收来自航班调度系统的航班到达事件；

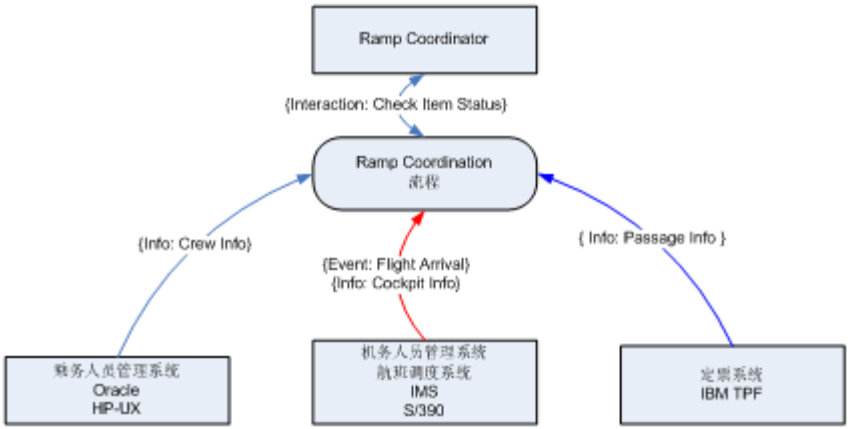


图 5： Ramp coordination 流程系统交互图

IT环境分析

- 集成存储在IMS和TPF等主机系统中的航班调度信息和定票信息的方法
  - 把这些主机上的数据进行了集中，并通过信息服务的形式输出给开放系统使用。
  - 来自IMS的航班信息和机务人员信息，来自Oracle的乘务人员信息和来自TPF的乘客信息都被汇集为一个业务对象Flight BO。Retrieve Flight BO服务提供访问这种业务对象的手段。Retrieve Flight BO服务隔离了底层实现的复杂性。

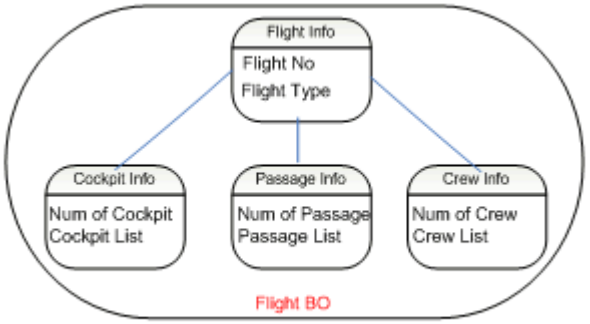


图 6： Flight BO

高层架构设计

- 根据需求和设计阶段的业务模型和现有IT环境调研结果，再结合传统的IT应用开发方法，设计了Ramp Coordination系统的高层架构。

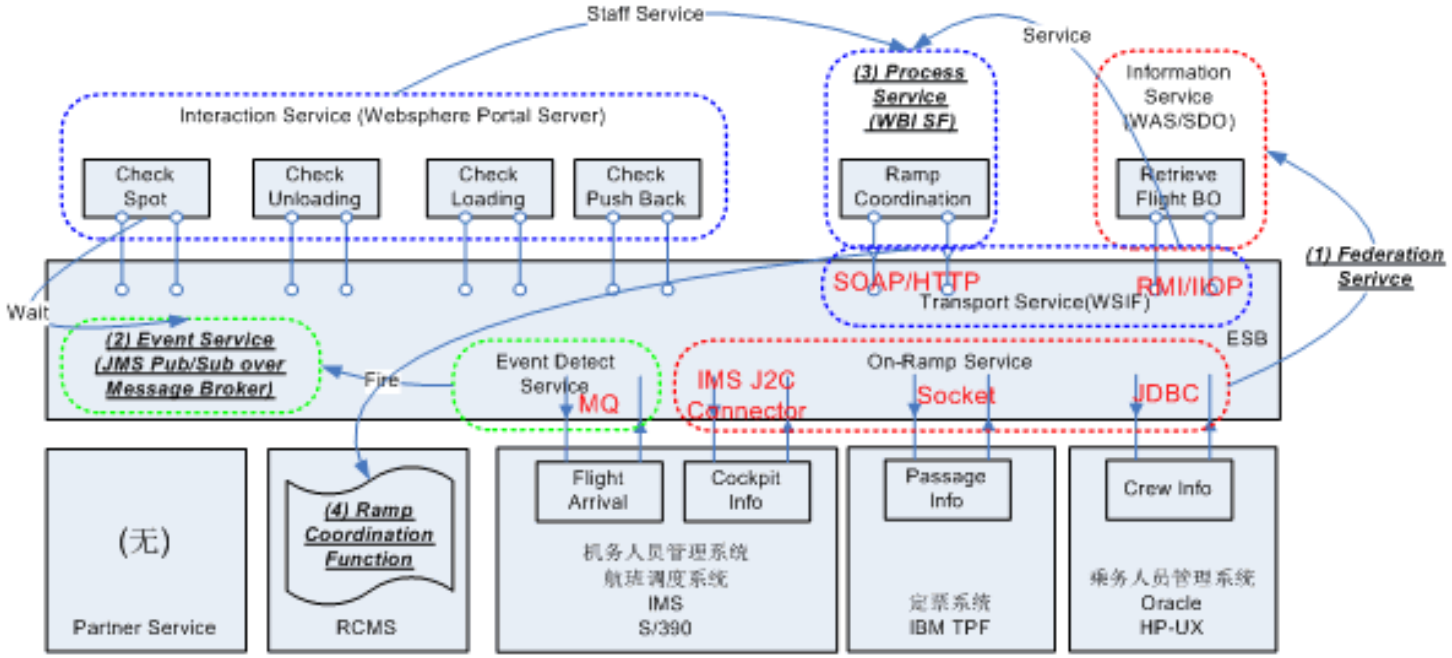


图 7 ： Ramp Coordination 系统架构

高层架构设计

- 主要架构元素以及它们间的工作关系
  - 信息服务—Federation Service: Ramp Coordination流程中需要从已有系统中提取四类信息，在Service建模阶段这四类信息被聚合为Flight BO（Business Object）。如上文所述，Retrieve Flight BO服务用于从已有系统中提取Flight BO。它实际上是一个Federation Service，将来自乘务人员管理系统、机务人员管理系统和订票系统中的信息聚合在一起。从这三个已有系统来的Crew Info, Cockpit Info和Passage Info是在已有系统中已经存在的业务逻辑或业务数据，它们属于可接入服务(on-ramp service)，接入的协议分别为JDBC, IMS J2C Connector和socket。乘务人员管理系统基于Oracle数据库，Crew Info可以直接通过JDBC获得。机务人员管理系统基于S/390上的IMS, IBM已经提供了IMS的J2C Connector， 所以Cockpit Info可以通过J2C connector获得。订票系统构建在IBM TPF之上，由于实时性的要求，socket是比较好的接入方法。Retrieve Flight BO被实现为一个EJB，外部访问通过RMI/IIOP绑定访问这个服务。在Retrieve Flight BO内部，Flight BO以SDO来表示。

高层架构设计

- 主要架构元素以及它们间的工作关系
  - 企业服务总线中的事件服务— Event Service: 在检查机务环境安全(Check Spot)前，Ramp Coordiator需要被通知航班已经到达。这个业务事件由航班调度系统激发，Flight Arrival是典型事件发现服务(Event Detect Service)，它通过MQ将事件传递给Message Broker，通过JMS的Pub/Sub，这个事件被分发给Check Spot。这里的Event Service是本例中ESB的重要组成部分。通过ESB上的通用事件服务，现有Information Hub的缺陷得到了克服。应用程序间的事件集成不再需要点到点的方式，而是通过ESB的事件服务完成订阅发布，应用程序间的耦合性得到了极大的缓解。

高层架构设计

- 主要架构元素以及它们间的工作关系
  - 流程服务－ Process Service: Ramp Coordination被实现为一个Process Service，它被WBI SF的BPEL4WS容器执行，BPEL4WS容器提供Choreograph Service, Transaction Service和Staff Service支持。Ramp Coordination通过RMI/IIOP协议调用，在BPEL4WS容器中WSIF被用于通过各种协议调用服务，它成为ESB中Transport Service的一部分。Ramp Coordination中的人工动作被实现为Staff Service而集成到流程中。这里Staff Service通过Portlet实现，运行在Websphere Portal Server上。Portal Service实现部分Delivery Service支持PDA设备，Ramp Coordinator通过PDA设备访问系统。
  - 企业服务总线中的传输服务－RCMS是即将新建系统用于提供包括Ramp Coordination在内的Ramp Control的功能。RCMS通过由WSIF实现的Transport Service以SOAP/HTTP调用Ramp Coordination服务。

高层架构设计

- 主要架构元素以及它们间的工作关系
  - 信息服务—Federation Service: Ramp Coordination流程中需要从已有系统中提取四类信息，在Service建模阶段这四类信息被聚合为Flight BO（Business Object）。如上文所述，Retrieve Flight BO服务用于从已有系统中提取Flight BO。它实际上是一个Federation Service，将来自乘务人员管理系统、机务人员管理系统和订票系统中的信息聚合在一起。从这三个已有系统来的Crew Info, Cockpit Info和Passage Info是在已有系统中已经存在的业务逻辑或业务数据，它们属于可接入服务(on-ramp service)，接入的协议分别为JDBC, IMS J2C Connector和socket。乘务人员管理系统基于Oracle数据库，Crew Info可以直接通过JDBC获得。机务人员管理系统基于S/390上的IMS, IBM已经提供了IMS的J2C Connector， 所以Cockpit Info可以通过J2C connector获得。订票系统构建在IBM TPF之上，由于实时性的要求，socket是比较好的接入方法。Retrieve Flight BO被实现为一个EJB，外部访问通过RMI/IIOP绑定访问这个服务。在Retrieve Flight BO内部，Flight BO以SDO来表示。

开发过程

- 尽管以服务为中心的企业集成在开发阶段和普通的应用开发并没有本质的区别，但是它在角色，职责、工具和方法还是有不少自己的特色。
- 下图汇总了案例中使用的开发角色，职责，开发方法和工具。

角色	开发方法或工具	任务
Business Analyst	WBI Modeler CBM	进行流程建模
Architect	Rational Software Architect SOMA	利用 SOMA 进行服务建模 系统高层设计和建模
Integration Developer A	WSAD-IE	服务和流程开发
Integration Developer B	Websphere Portal Server Toolkit	Portlet 开发
Integration Developer C	WSAD-EE	已有系统集成
Integration Developer D	WSAD-IE	ESB 相关功能开发

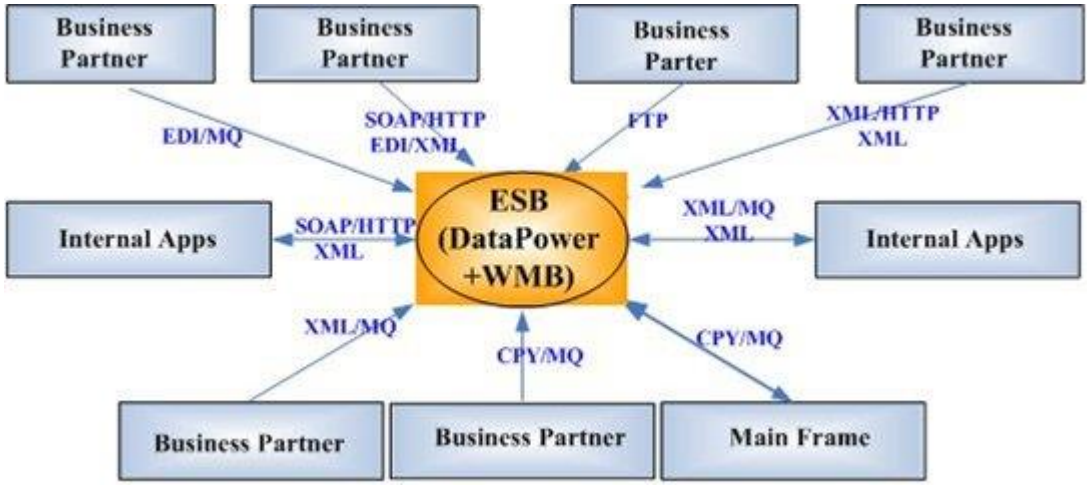
表 2：角色划分和工具支持

本章内容

1. 基于ESB的企业SOA项目需求分析
2. 基于ESB的企业SOA项目方案设计
3. 航空公司SOA案例分析
4. 制造企业SOA案例分析

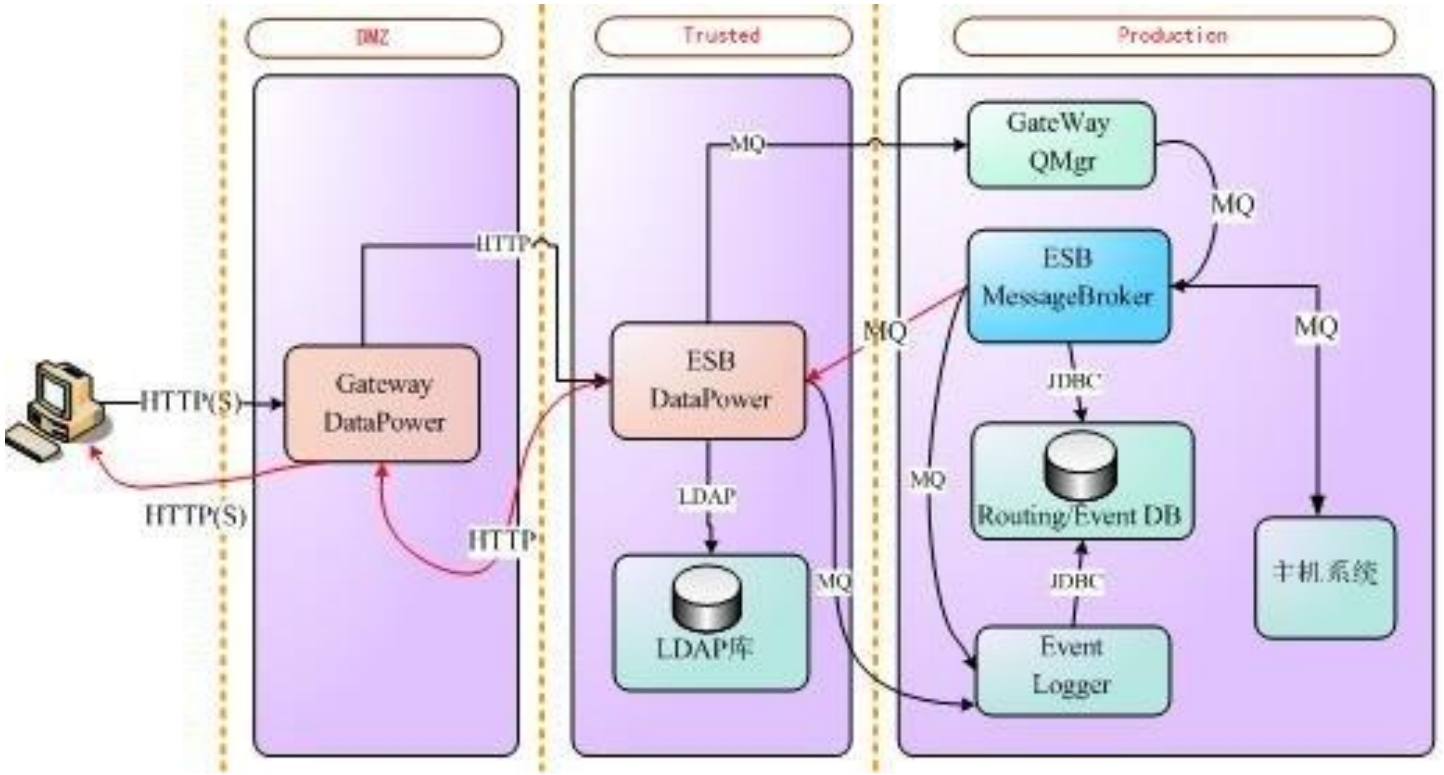
项目背景

- 其合作伙伴以多种方式调用该企业内部的后台服务，包括标准的 SOAP/HTTP 方式、XML/MQ 的方式、EDI/MQ 的方式甚至 FTP 的方式等；
- 该企业内部的一个主要核心业务系统运行在 IBM 大型主机上，对外的接口方式采用 WebSphere MQ，数据格式采用传统的 COBOL CopyBook 的方式，除了该主机系统之外，该企业的其他一些内部应用会以 SOAP/HTTP 和 XML/MQ 的方式与外界交互。



DataPower/Message Broker联邦ESB解决方案

- 使用的IBM产品主要包括 Data Power XI50 和 WebSphere Message Broker V6.0.2，DB2 V9 以及 WebSphere Application Server V6.1。



联邦ESB解决方案的组件构成

- Gateway DataPower：
  - 由 DataPower 作为企业对外的 ESB 入口，即担任 B2B 网关的角色，侧重提供其它企业的接入服务以及安全控制方面的工作；而采用 Message Broker 作为企业内部的 ESB 总线，实现企业内部各种应用系统，包括传统应用系统之间的集成平台。
  - 采用了两台 DataPower，其中 Gateway DataPower 主要负责外围的接入，接入的通信协议包括 MQ，HTTP(S)，JMS，FTP 四种。Gateway DataPower 位于网络架构中的 DMZ 区，它负责 XML threat protection，信息属性传递（ 例如 IP 地址的传递 ）和 SSL 传输层安全控制。

联邦ESB解决方案的组件构成

- ESB DataPower：这台设备位于网络架构中的受信区，对输入的消息进行更进一步的处理，包括：
  - 校验 Inbound/Outbound XML 类型消息的 Schema；
  - 实现非 MQ 通信协议和 MQ 通信协议之间的转换，在 ESB DataPower 和 ESB WMB 之间采用 MQ 的通讯协议，因此在ESB DataPower上要将所有的协议转换为MQ；
  - 信息头的处理：将创建我们为应用设定的消息头并且插入到接收到的数据包头或 SOAP 头中。
  - 认证 / 授权：通过 LDAP 进行发送者的认证和授权；
  - 日志：为了审计目的，通过日志服务（Logging Service）组件对输入信息进行日志处理。

联邦ESB解决方案的组件构成

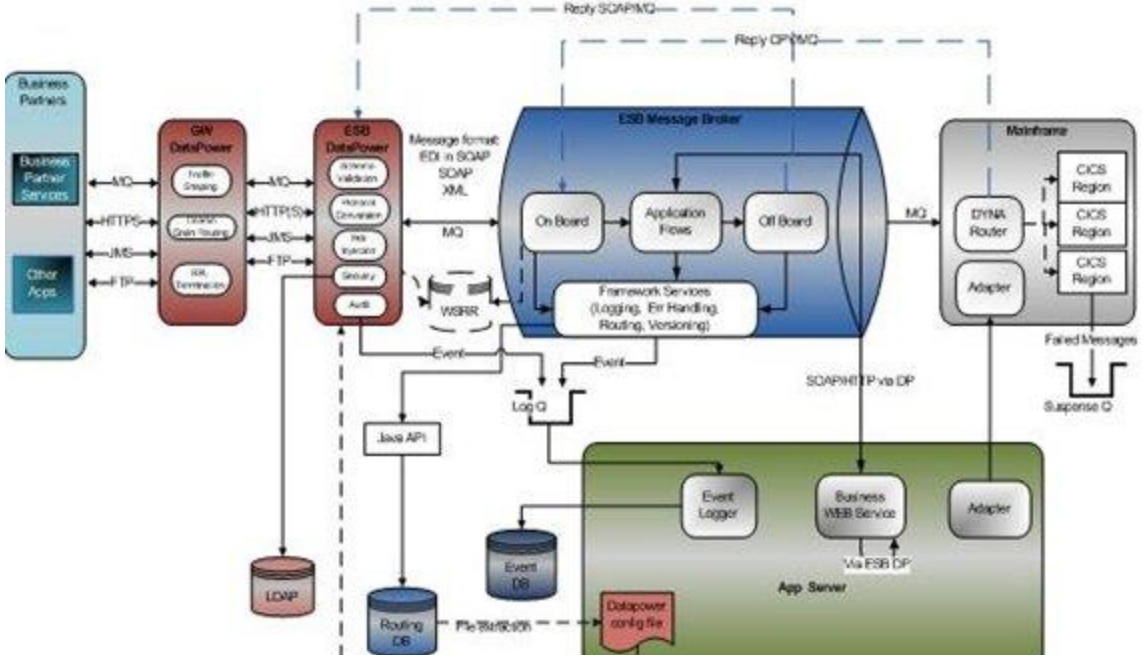
- ESB Message Broker：由于后台核心系统是位于IBM 大型主机上的应用系统，需要的数据格式是COBOL Copybook 的格式，因此在Message Broker上将实现XML到Copybook的转换。在 Message Broker上，设计了3种典型的Message Flow：
  - OnBoard Flow: 通过查询数据库设置消息的环境树(Environment Tree)，把输入消息转换为标准的 XML格式；
  - Main Application Flow：用于应用逻辑处理；
  - OffBoard Flow: 依据消息的 Environment Tree，将消息路由到特定的 Service Endpoint（服务端点），并且将数据转换为后台系统需要的格式。

联邦ESB解决方案的组件构成

- Application Server 组件：该组件接收和响应 Web Services 请求，并且 Logging 组件也运行在该组件上。
- 后台主机系统：负责后台的业务逻辑处理。
- LDAP 组件：存储认证/授权相关的信息。
- Routing Database（路由表）：路由表中存储了服务路由信息、服务版本信息等，其中包含了特定于应用程序的配置信息，通过Java API对其进行访问来决定服务的路由和绑定。
- Event Database/ Event Logger（事件数据库/事件日志处理器）：Event Database中存储了各种日志和错误信息，Event Logger是一个Java应用，用于将各种日志和错误信息存储到事件数据库中。

DataPower/Message Broker 联邦 ESB 的关键服务组件设计

- 从服务的角度，联邦ESB主要由以下服务组件构成
  - Gateway Services（网关服务）
  - Schema Validation Services（模式校验服务）
  - Security Services（安全服务）
  - Routing Services（路由服务）
  - Transformation Services（转换服务）
  - Logging, Error Handling and Notification Services（日志服务）



Gateway Services（网关服务）

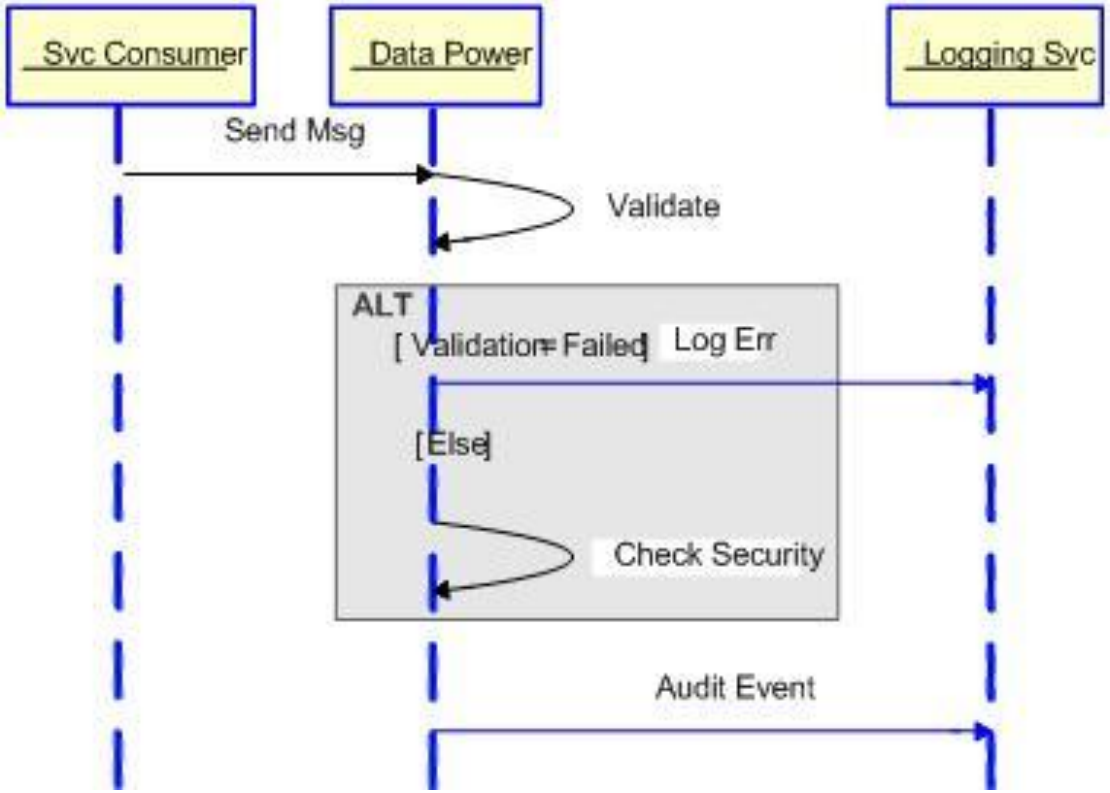
- 这个组件的所有功能都是通过对 DataPower 的配置实现的。它是 Internet 和 Intranet 之间的一个网关，从 ESB 的角度，它就像第一道防火墙，起到对企业外部服务调用的安全控制和协议转换的作用。同时，它根据来源数据的格式，例如 SOAP，XML，文本或 CopyBook 等，进行粗粒度的路由。

Schema Validation Services（模式校验服务）

- 这个组件的所有功能也都是通过对 DataPower 的配置实现的，开发者可以在部署时，在 DataPower 设备上配置有关的 Schema，对 XML Schema 的高速校验是 DataPower 的一个非常重要的功能。需要注意的是，在我们的案例中，我们的联邦 ESB 除了 XML 格式之外，还必须支持 Copybook 的主机格式，这是 DataPower 不擅长的，对这种格式的解析，我们将交给后面的 Message Broker 来实现，这正是体现了在这个联邦 ESB 解决方案中 DataPower 与 Message Broker 的各自定位以及无缝配合。

Schema Validation Services（模式校验服务）

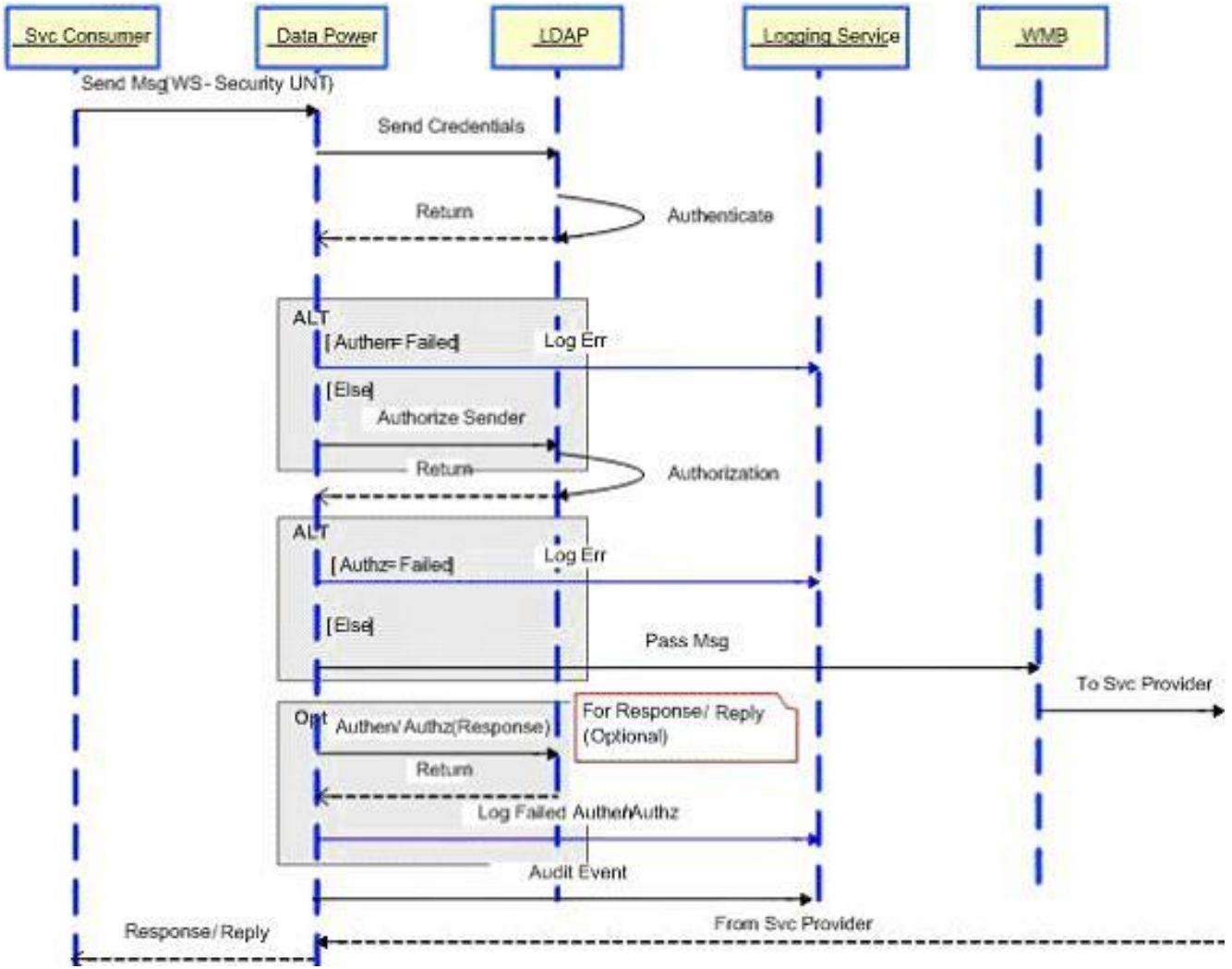
- 模式验证服务主要由ESB DataPower实现，它负责对SOAP/XML消息模式的校验，如果失败，则调用日志服务进行错误处理。



Security Services（安全服务）

- 安全服务将负责认证、授权和审计 (3A)。所有这些功能也都是通过对 DataPower 的配置实现的，采用 SSL 作为传输级安全控制，对于 MQ 的消息使用 MQRFH2 消息头来实现消息级安全控制，对于 SOAP 消息采用 WS-Security 标准来进行安全控制。

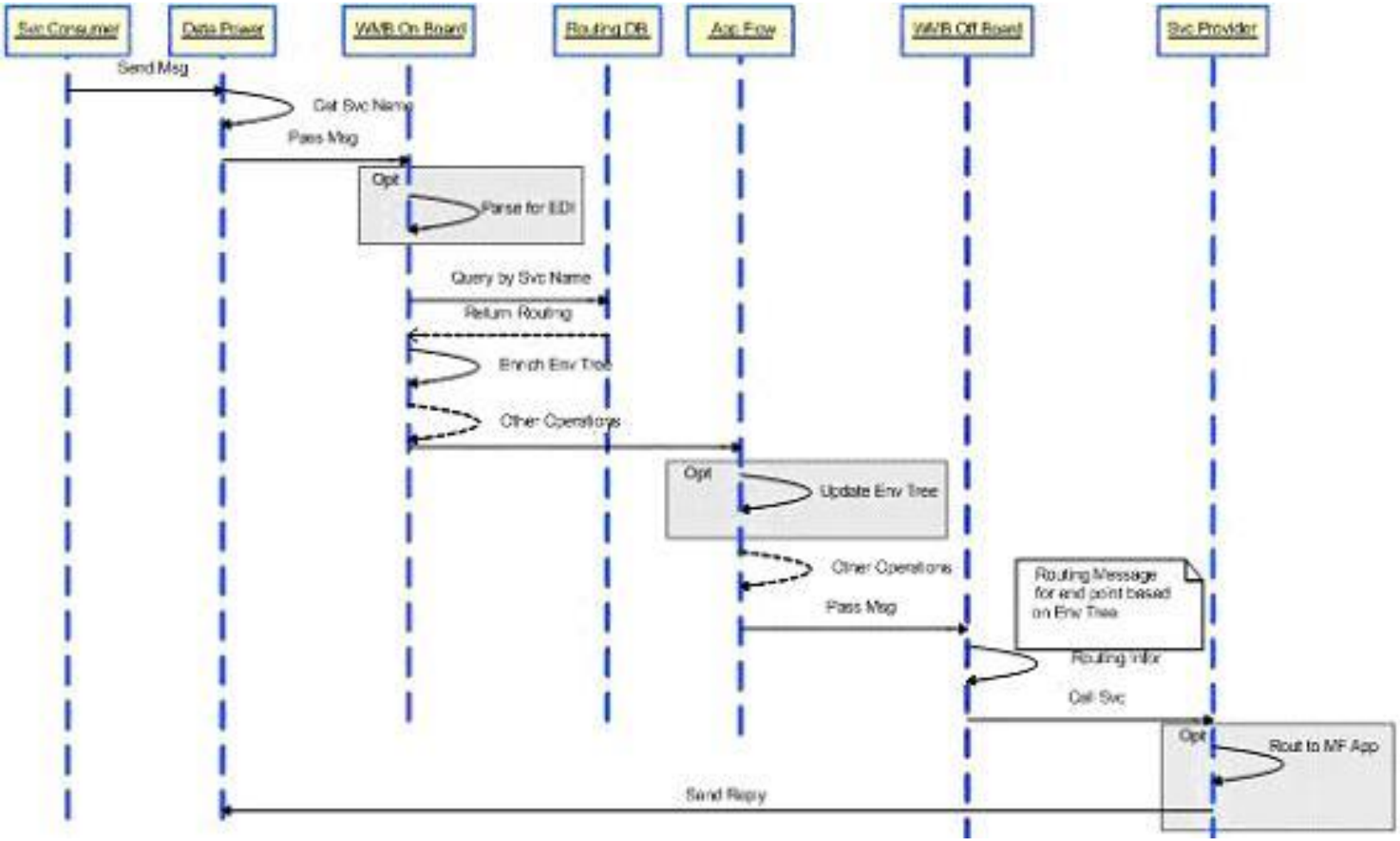
SOAP/XML 消息认证授权序列图



Routing Services（路由服务）

- 整个系统的路由服务将分为两层，第一层是介于ESB DataPower和ESB Message Broker之间，第二层是在Message Broker内部。
  - 第一层的路由是一层粒度较粗的路由，我们使用请求者的URI来决定其调用的服务名称，然后我们将服务名称和MQ队列的名称一一对应，这样我们就把请求数据路由到后面的Message Broker上对应的队列中了，当然这是一种简单的处理方式，从理想的角度而言，我们可以使用WebSphere Service Registry&Repository这样的服务目录和注册库产品来实现。
  - 第二层路由将由Message Broker来实现，它通过访问路由表(Routing DataBase)来获取路由信息，

路由服务序列图


Svc Consumer
Data Power
WMB On Board
Routing DB
App Flow
WMB Off Board
Svc Provider
Send Msg
Get Svc Name
Pass Msg
Opt
Parse for EDI
Query by Svc Name
Return Routing
Enrich Env Tree
Other Operations
Opt
Update Env Tree
Other Operations
Routing Message for end point based on Env Tree
Pass Msg
Routing Info
Call Svc
Opt
Rout to MF App
Send Reply

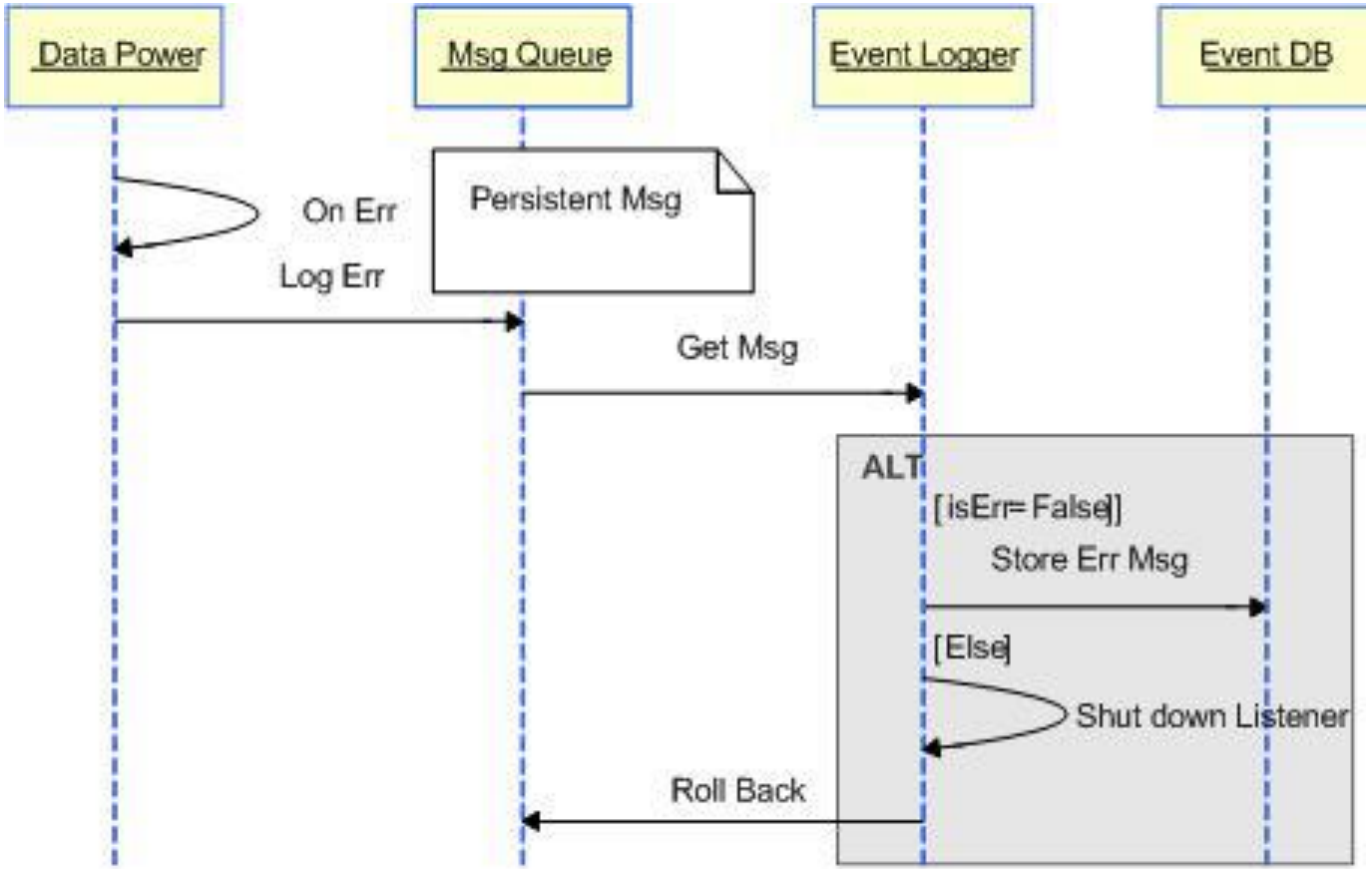
Transformation Services（转换服务）

- 转换服务由 DataPower 和 Message Broker 分别完成，其中 DataPower 实现 XML 到 XML 的转换，Message Broker 实现 XML 和非 XML，如 XML 与 COBOL Copybook 之间的转换。

Logging Services（日志服务）

- 日志服务由3个组件构成，分别是 Event Logger（日志处理器），Event DB（日志数据库）以及 Log Viewer（日志查看器）。
  - 日志处理器是一个 J2EE 应用，它通过 Message Driven Bean 读取 Log Queue（日志队列），然后将日志信息写入日志数据库。
  - 日志查看器是一个 GUI 应用，用来查看日志信息。从方案设计的角度，整个系统应该采用一致的日志和错误处理策略，在此我们仅列举 Data Power 的日志设计作为参考，其他的日志设计雷同。

日志服务序列图


Data Power
Msg Queue
Event Logger
Event DB
On Err
Log Err
Persistent Msg
Get Msg
ALT
[isErr= False]
Store Err Msg
[Else]
Shut down Listener
Roll Back