**基于 Scrum 敏捷开发的 EDI管理链式微服务实践**

**摘要**

为解决制造业企业 EDI 数据交互难题，提升供应链协同效率，公司于 2020 年 5 月启动 EDI 管理系统项目。笔者作为技术负责人，主要负责整体方案设计和落地。在项目推进过程中，笔者发现，随着 EDI 系统接入的贸易伙伴不断增多、业务场景日益复杂，在云原生应用中服务的管理和消息传递变得愈发复杂。而服务网格可以通过提供服务熔断、重试、负载均衡、熔断降级等功能，来管理那些必须运行在复杂环境中的服务。在该项目中，笔者通过深入研究服务网格技术的原理与特性，结合EDI 数据交互实时性强、准确性要求高的实际需求，充分利用服务网格服务发现、流量管理等功能，使得项目在预期时间内高质量交付，各项性能指标均达到或优于预定标准。该平台在经历一系列的业务优化升级变更，版本的并行切换后，仍然高效稳定运行至今

**项目背景**

我司作为全球领先的企业资源管理解决方案服务商，自成立以来，始终致力于为制造业企业提供高效、稳定的 ERP 解决方案，凭借深厚的技术沉淀与对市场趋势的敏锐洞察，已成功为众多企业实现了生产、采购、财务等核心业务的数字化管理升级。随着全球制造业供应链体系加速变革，企业间数据交互效率成为制约产业协同发展的关键因素。尽管 EDI 技术广泛应用于供应链数据自动化交互，但 ANSI X12、EDIFACT 等国际主流 EDI 标准与企业个性化业务需求之间存在适配难题。据行业数据显示，传统模式下企业 EDI 数据处理错误率高达 15%，平均订单处理周期长达 3 - 5 天，严重影响供应链响应速度与企业运营成本控制。

2020 年5月，公司管理层基于对制造业数字化转型趋势的战略研判，深刻意识到构建一套自主可控、高度适配企业业务需求的 EDI 管理系统，是突破当前供应链协同困境、强化企业核心竞争力的关键。为实现数据协议全流程自动化管理，打破企业间数据交互壁垒，公司正式启动 EDI 管理系统研发项目。旨在以标准化与定制化结合的方式，满足不同企业的多样化业务场景需求，从而助力客户提升供应链协同效率，在激烈的市场竞争中占据优势地位。

该项目从规模和复杂度来看，属于中大型企业级应用项目，聚焦 EDI 数据交互核心需求，打造三大核心功能组件：订单管理组件实现全流程数字化管控，覆盖订单全生命周期；数据格式映射组件支持多标准数据格式双向转换，保障数据传输精准高效；EDI 结果监控及展示组件实时追踪交互状态，智能预警异常并可视化呈现处理结果。项目初期组建了12人的专业团队，成员覆盖业务分析、软件开发、测试等多个领域。随着项目推进及业务需求深化，后期团队规模扩展至18人，进一步充实了系统优化、功能拓展及运维保障等方面的力量，确保系统稳定运行与持续迭代升级。

**技术方法说明**

云原生应用中的服务管理和消息传递始终是一个比较棘手的问题。以某个新服务升级为例，传统架构下，如果要做新产品上线的灰度发布，需要由业务代码中来判断哪些用户请求可以导向新的服务版本，哪些导向旧版本。这种硬编码方式有很多缺点，诸如导致代码臃肿，策略变更不灵活等等。经过深入调研，笔者发现可以使用服务网格来解决这些问题。从本质上来看，服务网格是一个专用的基础设施层，用于管理服务之间的通信。相比传统的分布式解决方案，服务网格不管从配置上还是效果上都有明显优势。在笔者所在的项目中，主要使用服务网格的三大功能，分别是服务发现和注册，流量管理和可观测性。从服务发现与注册来看，服务网格能够自动感知服务实例的状态变化。它通过数据平面代理（如Istio 中的Envoy）来实时监控服务实例的上线和下线情况。在流量管理方面，服务网格可以通过设置特定的策略，实现自定义的流量分配，为我们实现诸如灰度发布，多版本路由提供了可能。在可观测性方面，服务网格能够收集三大类数据——Metric、Trace 和 Access Log，可以让我们轻松实现服务的调用依赖与性能分析、服务访问审计和错误排查工作，保障了云原生应用在复杂环境下的稳定、高效运行。

**主要内容**

通过笔者在项目前期的技术调研，发现 Istio 与 Kubernetes 紧密耦合，是为云原生应用构建在 Kubernetes 之上的理想服务网格解决方案。鉴于 Kubernetes 已成为团队在企业级应用开发中的容器编排标准，最终选定 Istio 作为项目的服务网格底座。以下将从服务注册发现、流量控制和监控告警三个方面展开阐述。

在服务注册发现环节，传统方式要么依赖 DNS 和负载均衡器，要么借助特定框架的服务注册中心，如团队此前常用 Eureka 管理 EDI 系统中的服务注册。各服务将自身信息（服务名称、IP 地址、端口等）注册到 Eureka 服务器，服务消费者据此获取服务提供者列表并进行调用。相比之下，Istio 显著简化了服务注册与发现流程。当 EDI 系统中的服务启动时，Envoy 代理会自动向控制平面注册，上报服务实例信息。控制平面维护全局服务注册表，记录所有服务地址与端口。其他服务调用时，Envoy 代理向控制平面查询目标服务地址并建立连接。在 Kubernetes 环境下，Istio 通过集成 Kubernetes API，自动获取 Pod 和服务信息并注册，无论是新接入的 EDI 数据转换服务，还是下线的旧数据传输服务，Istio 都能实时更新注册表，保障服务间通信顺畅。

流量控制环节，团队在 EDI 系统中常需处理多版本路由场景。例如，企业与贸易伙伴升级 EDI 订单传输协议时，旧版 order 服务由 v1 deployment 提供，遵循原有传输规则；新开发的 order 服务 v2 版本适配了新协议。借助 Istio，通过判断请求 header 中的协议标识信息进行路由 —— 携带旧协议标识的请求导向 order v1，新协议标识的请求导向 order v2。Istio 利用 VirtualService 和 DestinationRule 核心资源对象实现路由分发：通过 VirtualService 定义请求 header 中协议标识与服务版本的映射规则，当标识为旧协议时路由至 order 服务的 v1 subset，新协议时路由至 v2 subset，精准实现新旧协议切换的流量控制。

在监控告警方面，传统微服务监控方法主要关注服务自身的运行状态，对于服务之间的交互和整体系统的运行情况掌握程度有限。虽然笔者也尝试过通过一些分布式追踪工具（如Zipkin）来跟踪请求在多个微服务之间的请求，但这种方法仍然无法完全掌握服务间的网络连接质量、服务间通信的详细协议等信息。笔者团队通过 Prometheus 收集Envoy 代理上报的Metric 数据，提供网格拓扑、服务拓扑、服务监控（请求数量、请求状态码分布、请求耗时、请求大小），然后把 Grafana 作为可视化工具，通过定制监控面板，将 Prometheus 收集的数据以图表的形式展现出来，实时监控服务网格的关键指标。比如笔者比较关注服务请求量，服务响应时间，服务间流量分布，通过掌握这些指标，及时发现并处理数据传输延迟、协议转换异常等潜在问题。

**结尾**

得益于服务网格的实施，本项目在运维管理的成本上得到了极大改善。虽经历了多次的需求变更，大大小小数十次迭代，但该项目至今仍然稳定运行，取得了设计之初的预期结果。作为技术负责人，笔者也深切感受到，服务网格能否成功应用，核心在于精准分析项目实际需求与预期收益，切忌盲目跟风采用新技术。服务网格并非万能，它在为项目带来诸如流量治理，流量路由、熔断和限流能力的同时，也对团队的技术能力和运维经验提出了更高要求。在技术引入的时候，技术负责人不仅要充分考虑团队的技术能力和运维经验，同时也要考虑技术的成熟性和稳定性。只有在项目规划阶段充分权衡利弊，综合考虑各方面因素，并在项目实施过程中持续优化细节、及时解决问题，才能确保项目取得圆满成功，让新技术真正为项目开发赋能。