ICS
CCS

团　体　标　准

T/CAAMTB XX-202X

汽车 SOA 架构设计与软件平台框架

Automotive SOA architecture design and software platform framework

（征求意见稿）

2023-XX-XX 发布　　XXXX-XX-XX 实施

发 布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本标准由****提出并归口。

本标准起草单位：****、****、****、****、****、****、****、****。

本标准主要起草人：**、**、**、**、**、**、**、**、**、**、**、**。

引　言

SOA(面向服务的架构)作为一种通用的组件模型，将应用程序的不同功能单元（称之为服务）进行拆分，并通过在这些服务之间定义良好的接口和协议将其联系起来。接口协议应采用中立的方式定义，独立于实现服务的硬件平台，操作系统和编程语言。

本文件针对汽车软件平台，规范了SOA架构设计的通用标准，包括汽车SOA软件平台设计规范、架构分层以及各层设计规范。提出了汽车SOA架构设计与软件平台框架定义，规范各层软件的接口设计，实现与硬件、操作系统及其他应用软件模块功能解耦，最终实现SOA服务的高内聚、低耦合的设计初衷。本文件将为整车SOA架构一致性提供技术要求及设计规范，填补国内汽车软件行业的空白。

汽车 SOA 架构设计与软件平台框架

1. 范围

本文件描述了汽车SOA软件平台的整体定义、分层结构及各层的功能定义。

本文件规定了汽车SOA软件平台的整体设计规范和分层设计规范。

本文件适用于支持汽车域控制单元或HPC的平台化软件开发。

2. 规范性引用文件

无

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

3. 术语和定义

3.1

硬件抽象 Hardware Abstraction

硬件抽象层通过操作系统、基础软件等，将硬件能力进行封装，对上层提供标准的接口，提供服务化的硬件抽象接口，供上层调度使用。

3.2

服务接口 Service Interface

服务接口（Service Interface）用于定义SOA软件平台三种服务之间通信接口（Event/Method/Field）消息类型和具体的命名空间。

3.3

E/E架构 Electronic Engineering Architecture

汽车电子电气架构又称EE架构是指整车电子电气系统的总布置方案，主要包含硬件架构、软件架构、通信架构三个方面。

3.4

域控制单元 Domain Control Unit

域控制单元是一种高速计算设备，具有强大的硬件算力和各种软件接口，可以使系统集成度更高，降低对功能感知和执行硬件的要求。

4. 缩略语

下列缩略语适用于本文件。

SOA	面向服务的结构	Service-Oriented Architecture
ASF	整车服务架构	AUTOSEMO Service Framework
IPC	进程间通信	Inter-Process Communication
HPC	高性能计算机	High performance computing
API	应用程序编程接口	Application Programming Interface
OTA	空中下载技术	Over-the-Air Technology
AI	人工智能	Artificial Intelligence
OS	操作系统	Operating System
RPC	远程过程调用	Remote Procedure Call
IDE	集成开发环境	Integrated Development Environment

QOS	服务质量	Quality of Service

5. SOA 软件平台概述及设计规范

5.1. SOA 软件平台概述及定义

SOA 软件平台是用于提供面向服务的软件架构能力的平台，其包括了所有提供 SOA 能力的**基础型服务软件和功能型服务软件**。用于对业务层提供 SOA 通讯能力，屏蔽硬件差异，提供整车业务需要的各类服务封装。

基础型服务软件包括：

硬件抽象：主要用于屏蔽硬件和操作系统差异，提供基础通讯、存储、日志、升级等节点级的最基础能力。

ASF：在基础软件基础上整合控制器级和整车级功能封装，强化高层级（控制器级、整车级）能力，弱化单节点（核或控制器）功能。提供控制器级服务和整车级服务，使开发者不再关注硬件，从而专注应用层开发。

功能型服务软件包括：

整车服务：提供支持整车的各种功能和业务需求，实现整车的统一协调，管理，调度，控制等服务

车云服务：提供车云通讯、车云服务治理、服务调用框架、服务定义/发布/分发平台等功能，将常用的车云通讯能力封装，在车端提供数据汇总功能统一传输至后台处理。

5.2. SOA 软件平台设计规范

5.2.1.SOA 软件平台架构设计规范

SOA 软件平台由硬件、OS、通用基础软件、通用服务框架中间件等节点能力提供基础型服务，通过调用基础型服务提供的标准服务接口组合汇总为整车级服务。跨域融合应用可以直接调用整车级服务实现业务逻辑，也可直接与云端标准服务接口联通。

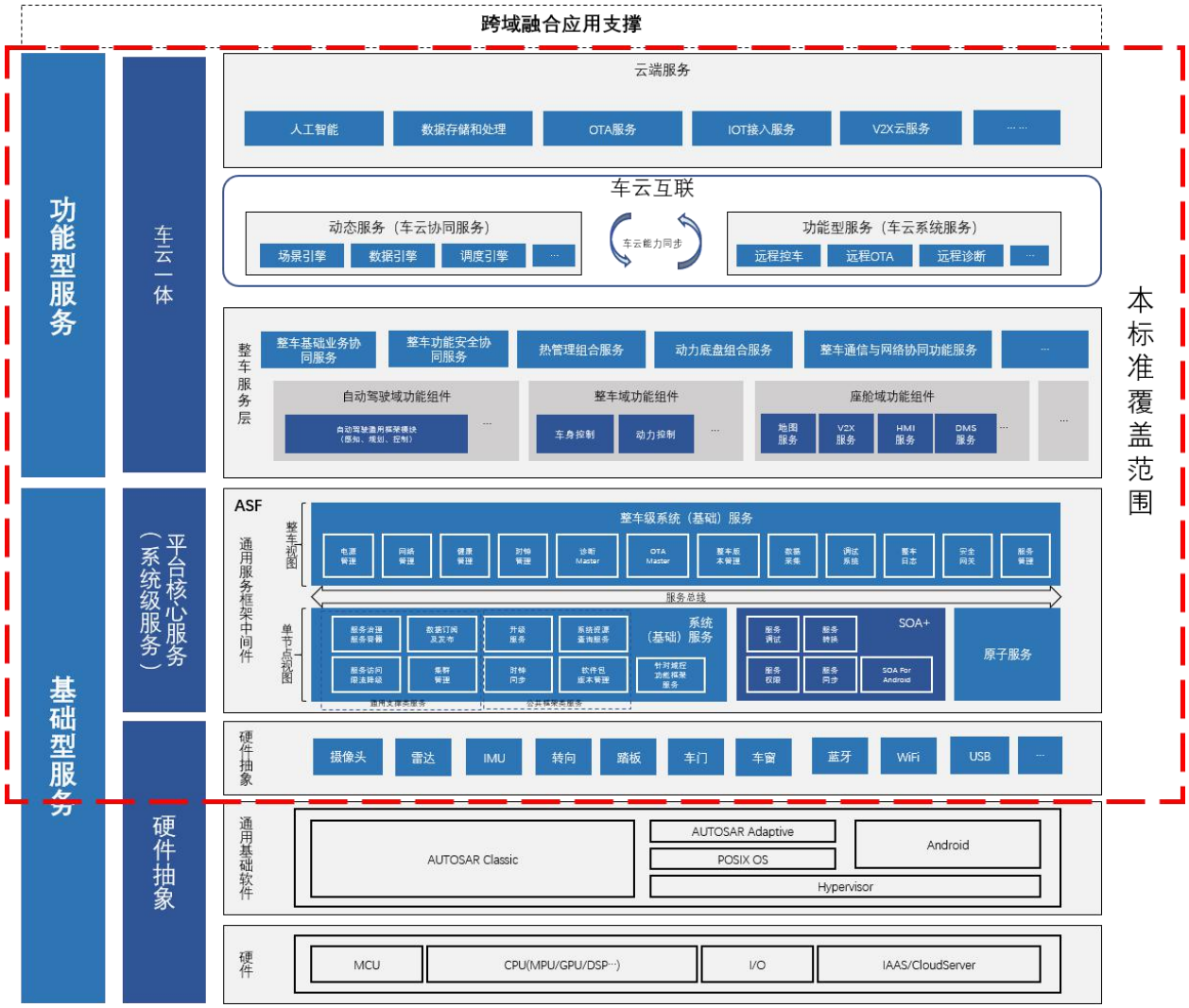

图 1. SOA 软件平台架构图

硬件抽象：通过基础软件将硬件能力进行封装，对上层提供标准的接口，提供服务化的硬件抽象接口，供上层调度使用。

ASF：通过通用框架中间件，从单节点角度将节点能力进行封装，对上层提供更多基于 SOA 开发需要的服务；

整车服务层：从整车角度，将各节点的服务进行汇总，为应用开发提供更便捷的接口。

车云一体：通过车云能力的服务化，实现车云无缝衔接和车云服务自由调度，为应用层智能化应用提供快速开发的能力支撑。

5.2.2. SOA 软件平台服务设计规范

SOA 软件平台的在设计服务时，需要首先识别服务的各项属性，包括：

- 服务的基础属性（如：服务编码，服务中英文名称，服务性质编码，服务功能描述等）；
- 服务的技术属性（如：版本号，注册时间，依赖服务，接口方法，接口协议，启用时间，停用时间，权限属性等）；
- 服务的安全属性（如：认证，加密，调度属性，权限属性等）；
- 服务的配置属性（如：IP，端口号，VLAN 号，QoS 等）。

针对上述属性，服务设计宜符合可重用、低耦合与安全性的总体原则。具体包括：

- 自治原则

在 SOA 架构下，服务被视作实现功能需求的最小单元，也是参与服务编排的基本单元。服务需要具备单一职责，具有独立且单一的业务边界和业务能力。

- 可复用原则

  服务设计需综合考虑一个服务会被多个业务上下文使用，明确服务的功能定义和逻辑设计，以确保所有业务场景下的复用性。

- 可组合原则

  要求软件在做服务拆分时，需考虑重新组装性，即多个服务重新组合为新的服务或应用。

- 低耦合原则

  服务需与业务流程脱离，服务设计需保证服务与业务的异步执行、分布式部署。降低服务间的横向耦合度、降低服务与操作系统的纵向耦合度

- 安全性原则：服务的安全性应从传输级安全性，消息级安全性，应用程序级安全性三个方面符合各类信息安全标准、规范要求：
  - 传输级安全性：传输通道点对点的完整性和机密性；
  - 消息级安全性：消息的完整性，机密性，不可否认性及消息身份验证；
  - 应用程序级安全性：应用程序的权限体系，用户凭证管理、安全日志等方面提供防护。

- 命名原则

  服务实例及指代定义的服务实现，后文若单独写服务，则意为服务实例。服务名采用首字母大写，全英文名称。服务名不能有下划线。服务名后缀需要以 Srv 结尾 eg. LedControlSrv。

- 抽象化原则

  服务需要对外隐藏不必要的元数据、程序逻辑等，确保服务的升级不受曾使用该服务的业务制约，便于服务演化。

- 无状态原则

  服务设计需抽离有状态逻辑，设计为无状态的数据处理、计算处理等服务。

同时，考虑到系统级的性能要求和可维护性要求，针对 SOA 软件平台整体服务的设计，宜遵循以下要求：

- 服务数量精简

  服务设计应遵循正交性原则，服务的数量可以控制在合理区间之内。在落实服务能力单一化设计原则的过程中，容易引起服务数量的膨胀以及服务功能的混淆重叠。因此服务设计宜依据合理粒度进行抽象、合并，在保证服务能力单一化的基础上，控制总体服务数量。即实现服务自治原则和服务数量精简原则的对立统一。

- 服务关系清晰

  由于功能需求可以由多个抽象服务、逻辑服务、原子服务，通过组合、编排实现，因此整车系统需要清晰的描述多个功能需求的服务间关系。

- 服务管控机制

  服务应具备动态发现或注册发现机制，使得在系统设计新服务时，可以综合评估所有可被发现的服务，管控服务发布内容，避免服务功能重复。

- 服务逻辑健壮

  服务应具备健壮性和容错机制，提供故障恢复、冗余通道等机制，支持使用者的异常使用场景，保证服务的长时间稳定运转。

5.2.3.SOA 软件平台接口设计规范

在 SOA 软件服务平台中，服务消费方须通过服务接口获得服务提供方的各项能力，而服务接口是服务提供方和服务消费方之间的桥梁。因此，在设计服务接口过程中，服务接口定义需要在实现功能需求的基础上宜满足以下原则：

服务接口定义共性要求：

- **职责明确：**是指接口应专注于“有限”工作。“有限”工作可以是某项具体的功能需求、也可以是经过仔细权衡、抽象出来的某类功能需求。职责明确，是降低复杂度、实现高内聚、低耦合的前置条件，也是提高服务可复用性的基础。须在接口设计阶段慎重考虑接口职责，推荐从功能需求的角度出发，评估接口必要性、确定接口职责。
- **保持稳定：**在 SOA 体系下，一个接口变更，至少影响两个服务（提供者、消费者）。低级别服务的接口变更影响范围会更大。服务接口应从以下两个方面考虑，尽量保持稳定：

（1）充分利用面向对象设计领域的设计原则&模式，赋予接口一定的“弹性”，可以在一定程度上吸收未来的需求变化；

（2）基于较为完善的接口依赖关系和版本管理体系，尽量保证接口向下兼容，如果不能做到向下兼容，也要对变更影响范围有明确认识。

- **文档完善：**服务接口一般使用 IDL 描述。同时，仍推荐使用专门文档对接口的设计背景、职责定义、参数、返回值进行详细说明，并提供接口输入、输出的 Sample 数据，以及记录接口演化履历。
- **表达清晰：**接口命名应与接口职责相符合，接口参数数量不宜过多，数据结构不易复杂。另外，服务接口应在命名、参数、返回值、版本定义等方面遵循统一、规范的规则。描述清晰、且符合统一规则的服务定义表达，有助于提高接口易用性、可维护性。命名规则详见下文服务接口通用命名原则和服务接口分类命名原则描述。

服务接口通用命名原则：

- 服务接口名采用首字母大写，全英文名称
- 服务接口名不能有下划线
- 服务接口名需要以 SrvIf 结尾 eg. LedControlSrvIf
- 尽量采用单词全称，名称过长后才使用缩写；
- 名称不包含特殊字符,.^~；
- 名称如果过长，需要参考常用单词缩写表中内容进行缩写；
- 考虑服务和参数的复用性，各类命名中不应带有拓补节点信息；
- 不同层级的各类名称不能重复；
- 命名中不能使用如下系统关键字：

关键字	依据
skeleton	Follow AUTOSAR 标准
proxy	Follow AUTOSAR 标准
internal	Follow AUTOSAR 标准
resources	Follow AUTOSAR 标准
method	Follow AUTOSAR 标准
event	Follow AUTOSAR 标准
field	Follow AUTOSAR 标准
input/output	Follow AUTOSAR 标准
amsr	Follow AUTOSAR 标准
ara	Follow AUTOSAR 标准
com	Follow AUTOSAR 标准
someip	Follow AUTOSAR 标准
base	Follow AUTOSAR 标准
vac	Follow AUTOSAR 标准

std	Follow AUTOSAR 标准
serialization	Follow AUTOSAR 标准

服务接口分类命名原则：

SOA 平台上服务之间通信接口有 Event、Method 和 Field 三种形式，服务接口（ServiceInterface）用于定义 Event/Method/Field 消息类型和具体的命名空间，与具体的通信协议无关。每种服务形式也会有特定的命名原则，具体的：

- **Event**

Event 接口表示实际传输的数据，以数据为操作对象，只要能够清晰的表达数据的含义即可，命名规范遵循基础规范，后缀要以 Evt 结尾。

eg. CurrentVleEvt : 电流值

- **Method**

Mehtod 接口表示某种控制，通讯方式采用 RPC 远程调用，通常有诸如控制，状态查询，传输，注册，设置的控制行为。其中 Method 又分为 F&F,与 R&R 两类，F&F （fire-and-forget）为单次调用，不需要反馈，R&R （request resoponse）则需要反馈。

接口名称需要表达清楚该方法的含义，推荐使用动名词进行命名，采用驼峰命名规范，基于如上 CURD/REST 参考，命名要以后缀 Mtd 作为结束，并设计如下基础命名范式:

get 获取状态

set 设置状态

report 传递信息

judge 判断事件，返回 Boolean

create 创建线程/进程/动态服务/文件/事件等

delete 删除线程/进程/文件等

eg. getSwithStateMtd : 获取开关状态

- **Field**

Field 表示一种属性，通常指状态值或某种信息，名称应该清楚的表达该属性的含义。

Field 包含如下三类信息：

getter : 只读接口，原型为 method，获取服务端信息

notifier : 只读接口，原型为 event，接收服务端的数据

setter : 写入接口，原型为 method，设置/修改服务端相关信息

eg. VehMoveFld : 车辆移动控制

5.2.4. SOA 软件平台评价原则和产出物定义

5.2.4.1. SOA 软件平台评价原则

汽车 SOA 软件平台评价原则如下

- 架构设计

  架构设计上需要参照本文档汽车 SOA 软件平台的层次结构进行设计，确保各层次即使独立进行开发，也能融合到一个系统中运行。

- 支持的协议和标准

  评估平台是否支持 SOA 软件平台架构设计中明确需要支持的通讯协议，以确保与不同系统和技术的互操作性。

- 服务设计

  服务设计需要满足平台设计服务规范中的要求，按照抽象化，单一化，业务解耦的原则进行服务设计，确保服务的标准性，可复用性。

- 接口设计

服务接口的设计，需要满足平台设计接口设计规范中的要求，按照命令规范，分类规范的要求进行服务接口设计。

- 性能和可伸缩性
  测试平台的性能，包括吞吐量、响应时间和负载容量。评估平台的可伸缩性，确定它是否能够处理日益增加的负载。
- 安全性和身份验证
  检查平台提供的安全功能，包括身份验证、授权、数据加密和漏洞管理。确保平台能够满足应用程序的安全性需求。
- 监控和日志记录
  评估平台的监控和日志记录功能，以便追踪和分析服务的性能和问题。确定平台是否支持实时监控和日志记录。
- 容错和故障处理
  检查平台是否提供容错和故障处理机制，以确保系统在故障情况下仍能提供服务。评估平台的高可用性功能。
- 开发支持和工具
  平台是否提供开发工具、集成开发环境（IDE）和文档，以便开发和测试服务。
- 集成能力
  考虑平台的集成能力，包括支持的数据格式、消息队列、第三方服务和现有系统的集成。
- 数据隐私和合规性
  确保平台符合数据隐私法规和政策，以及数据保护的最佳实践。

5.2.4.2. SOA 软件平台产出物定义及要求

在 SOA 软件平台的开发过程中，至少需要包含以下过程：软件需求分析，软件架构设计，软件详细设计，软件单元测试，软件集成测试。根据平台开发过程，定义了各过程下的产出物的要求（包括通用要求和各产出物的单独要求）如下表所示：

	章节名称	要求
通用要求	工作产品变更履历	文档新编时从 V0.1 开始，经过正式评审、批准后的首次发布为 V1.0； 后续有更新时，版本依次递增
	文档目的	描述本文档的目的，指定目标受众
软件需求规范	输入文档	描述本文档的输入文档，包括但不限于系统需求规范，系统架构设计
	需求背景	本章为具体定义在第 3 章中的需求提供背景信息，便于受众更好的理解，例如软件功能概述、与其他功能/系统的交互、约束/假设等
	接口信息	识别并描述相关的外部接口，包括接口名、类型、发送方/接收方、参数/信号定义等
	功能需求	功能性需求除了描述正常情况处理外，还需描述异常情况处理。处理流程包括对输入的处理，核心行为，对输出的处理等。对于异常处理包括错误处理和中断处理，如果有中断，需要描述中断的恢复情况。 如果涉及到标定数据，则需指明“可标定”， 并给出标定范围、默认值
	性能需求	如响应时间、精确度、内存使用程度等
	其他需求	可以从软件运行环境、法律法规标准、质量属性等方面考虑。质量属性包括可靠性、可用性、可维护性、安全性、可移植性、可扩展性等
软件架构设计	限制和约束	包括组织约束（如团队规模、开发时间）；技术约束（软件约束（如开发环境、操作系统限制、外部的 Lib 库

		等）；硬件约束（如微控制器）
	整体框架及分工	采用层次结构图的方式，定义软件架构，确定软件组件在软件架构中的位置，承担的架构职能。识别软件组件的类别（如既存、新开发、第三方提供等等）。如有功能安全要求，添加软件组件的 ASIL 等级信息。
	外部接口	识别或定义相关的外部接口，接口信息包含 Interface Name, Types, Units, Resolutions, Ranges , Default Values 等
	架构方案决策	说明哪些方面需要进行技术决策
	静态设计	详细定义每个软件组件，包括组件的职责及接口，细化整体框架的内容
	动态行为设计	软件动态行为是描述软件运行时的行为，包括但不限于：Task（任务）、 Interrupt（中断）、Schedule（调度）、处理流程（可使用 Sequence 图）、状态迁移（可使用状态迁移图）、 数据流（可使用数据流图）
	资源消耗设计	包括 CPU Load 和内存（如：ROM, RAM, EEPROM 等）。定义资源消耗目标(%)及资源的使用设计
软件详细设计	模块设计概览	功能概述，简要说明本模块实现的主要功能；说明模块的设计是否具有平台依赖性；包括系统层次图、接口交互图等
	模块描述	列出模块所包含的所有子模块和子功能 可以配以用例图，类图，Component 图等说明，如有功能安全要求，添加 ASIL 等级属性
	子功能模块描述	列出所有子模块和子功能 可以配以用例图，类图，Component 图等说明，如有功能安全要求，添加 ASIL 等级属性
	数据结构、全局变量、宏定义	定义类型应该包括以下几个要素： Name、Type、Description、Range、Remark 描述模块中定义和使用的数据，包括：简单数据，如模块级的全局变量、静态变量、枚举、常量、宏；复合数据，如模块内部的结构、联合。
	内外部接口	定义接口应该包括以下几个要素： ID、Service name、Syntax、Sync/Async、Reentrancy、Parameters (in)、Parameters (inout)、Parameters (out)、Return value、Description
软件单元测试报告	测试结果分析	包括测试数据统计分析、测试结论和质量分析、遗留缺陷和建议等
	执行情况	包括静态代码分析问题修正率、代码评审问题修正率、测试计划及用例执行情况、回归测试执行情况、缺陷统计
软件集成测试报告	测试对象	说明本次测试的测试对象。测试对象的识别，可根据本次开发的 scope 确定，如功能列表或服务列表的接口、通讯等。
	测试计划	此阶段的测试计划详细信息
	测试结果总览	包括：测试总结、测试结果分布、版本基本信息、测试 Bug 分析

表 1.SOA 软件平台产出物定义

6. SOA 软件平台各层设计规范

6.1. SOA 软件平台硬件抽象层设计规范

6.1.1. SOA 软件平台硬件抽象层概述

在 SOA 架构下，软件平台对软硬分离的需求更加强烈，硬件抽象层隔离硬件和软件，使上层应用可不依赖于硬件进行独立开发和演进。其通过对传感器（如摄像头、雷达、IMU、蓝牙、WiFi）、执行器（转向、踏板）、传统 ECU（车门、车窗、USB） 等硬件资源进行抽象，通过标准化接口向 上为服务提供设备访问能力，同时屏蔽硬件功能实现的细节和差异（硬件差异&厂家差异），减少定制化与重复劳动。

硬件抽象层需要为上层服务提供完备的设备访问能力，包括硬件的初始化、反初始化、硬件控制、更新硬件工作参数、获取/通知硬件故障状态，发送/接收硬件数据等；上层服务通过调用硬件抽象层接口来实现相应的逻辑并提供对应的硬件服务（原子服务），而无需了解底层实现，无需关心底层硬件功能的具体实现细节。

6.1.2. SOA 软件平台硬件抽象层技术要求

硬件抽象层（HAL）是软件平台系统的核心组件之一，它为应用程序和系统服务提供与底层硬件交互的接口。通过 HAL 层，系统可以将硬件驱动程序与系统内核解耦，提高硬件兼容性和软件可维护性。

硬件抽象层技术要求如下：

- 接口标准：HAL 需要提供标准化的硬件访问接口，遵循操作系统规定的接口规范。
- 可移植性：HAL 应该具备良好的跨平台可移植性，能够适应不同的硬件平台和芯片架构。
- 性能优化：HAL 需要针对底层硬件进行性能优化，以提高系统运行效率和响应速度。
- 稳定性：HAL 需要保证系统的稳定性，避免因硬件故障导致系统崩溃或数据丢失。
- HAL 必须是线程安全的，因为多个线程可能同时访问同一个硬件设备。
- HAL 应该提供合适的错误处理机制，以处理硬件设备操作失败的情况。
- HAL 应该遵循软件平台系统的安全规范，防止未经授权的访问和数据泄露。

6.1.3. SOA 软件平台硬件抽象层设计规范

硬件抽象层设计需遵循如下原则：

- 归一化原则

  硬件抽象层设计应遵循归一化原则。通过抽象和聚类，实现通用功能的规范化、标准化的同时，屏蔽差异性。

- 标准化原则

  硬件抽象层设计应遵循标准化原则。标准化原则的核心思想是一致性约束。主要体现为功能定义标准化、接口规范标准化、性能指标标准化等。

- 可配置化原则

硬件抽象层根据项目实际情况，需要适配不同厂家、不同硬件类型、不同硬件数量的硬件设备等，在部署及运行阶段，需充分考虑功能及接口的可配置化。

- 容错性设计原则

设计时需设计保护方案，关键功能增加容错处理，使功能免于故障。具体表现有：需预计功能会被误用，预计功能使用者会出现故障等，通过容错设计，保证硬件抽象层的健壮性。

硬件抽象层接口需遵循如下原则：

- 硬件接口一般是硬件（如传感器、执行器等）功能的抽象。
- 硬件接口一般以基础 API 的形式对外提供功能。
- 硬件接口应完成一个硬件完整的功能。
- 硬件接口需要是易于管理、易于客户使用。
- 除了通用接口外，硬件抽象层需根据不同的硬件类型，提供不同的硬件接口；如传感器，提供数据或状态获取接口等；执行器，提供数据设置或硬件控制接口等。

6.2. SOA 软件平台 ASF 设计规范

6.2.1. SOA 软件平台 ASF 层概述

ASF 位于 SOA 软件平台架构的基础软件平台（即基础操作系统和运行环境）和功能业务层之间。ASF 屏蔽整车 E/E 拓扑，屏蔽传感器执行器在控制器（或域控制器异构芯片）上的部署，为上层业务软件提供统一的整车级的业务能力标准访问接口，支撑业务层软件快速开发，缩短产品化周期。

ASF 主要分为三个层次分别为：基础系统服务层、服务总线、整车系统服务层。

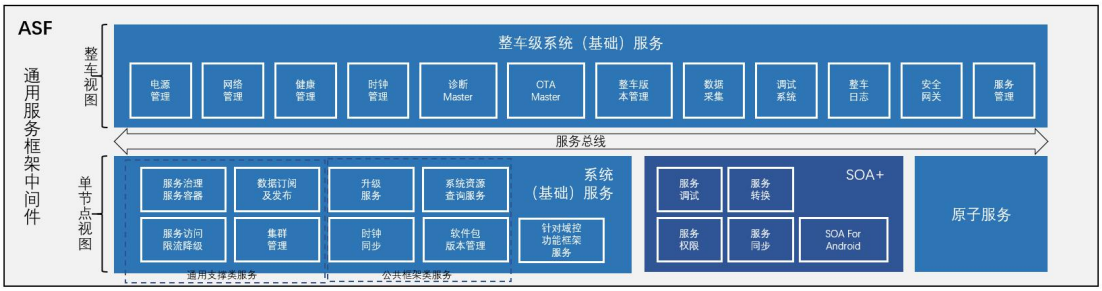

图 2 ASF 分层结构

基础系统服务层：主要是通过服务总线连通不同域控制器或域控制器异构芯片，为上层软件构建统一基础系统服务接口，上层软件开发可以独立于底层基础系统服务在控制器上的部署。主要功能模块包括时钟服务、安全服务、日志服务及升级服务等，这些服务来自于系统基础服务、SOA+、原子服务，提供单节点的基础系统服务。

服务总线：主要提供连接不同的服务，为不同的服务提供了一个统一的接口，服务只需要挂载在服务总线上。通过服务总线，可以通过适当的方式访问到不同的服务。

整车系统服务层：基于基础系统服务层做进一步开发封装，为上层软件提供整车级系统服务。该层主要功能模块包括整车诊断、整车 OTA、整车服务管理及整车大数据等。

6.2.2. SOA 软件平台 ASF 层技术要求

ASF 层是软件平台系统的核心组件之一，它向下使用标准化基础软件平台和操作系统提供的接口，提供更多整车业务层面需要的功能，并封装成基础系统服务与整车系统服务，为上层应用提供整车统一视图的服务功能及服务接口。

对于 ASF 层的定义和实现，主要有以下技术要求。

- 原子服务保持独立：在 ASF 层设计的原子服务因遵守平台服务设计规范，其设计应与硬件配置和实现无关，与上层功能服务层和下层的硬件驱动层解耦，完全独立。原子服务具有原子性：即设计的服务不可再拆分，作为服务的最小单位和执行实体，为功能服务提供最基础的执行或获取状态等服务。
- 灵活的接口标准支持：包括
  - API 接口（Application Programming Interface）：在行业内比较常用的有 AUTOSAR 的 ARXML，GENIVI 的 Franca-IDL （Interface Description Language），DDS 的 IDL 等；
  - 文件方式接口，如 INI、JSON、XML、Database 等。文件类服务接口需包括文件格式说明以及操作系统依赖的文件访问接口和访问权限。
  - 系统原生服务接口，如：娱乐系统上的窗口管理服务（wayland 接口），多媒体解码服务（gstreamer 接口）等。
  - IPC 接口，如 pipe，FIFO，互斥锁，条件变量，基于内存的信号量，读写锁，消息队列，socket，消息队列，命名信号量等。
  - 协议栈接口，即基于某种通信协议栈上提供服务对外的跨进程或跨平台的调用方式，包括：基础的网络协议如 TCP/UDP、AVB/TSN 等，应用协议如 HTTP、HTTPS、MQTT、gRPC 等，车载网络通信协议如 SOME/IP、DDS、DoIP、XCP 等。
- 多种服务类型支持：包括
  - 系统基础服务：即对基础软件平台（如 Adaptive AUTOSAR、Android 等）提供的通用化功能进行抽象，如健康管理服务、网络管理服务、时钟服务、电源管理服务等；
  - 功能服务：即提供具体业务功能相关的服务，包括与域控相关的专用中间件、应用层提供的功能。如 OTA 服务、安全服务，也可以对应用层提供功能，如支付服务、账户服务、导航服务等。
  - SOA 增强服务：SOA 增强服务具有通用性，即可为所有的应用服务提供通用功能，如数据存储、服务信号转换、服务调试等诸如此类的通用化功能。
  - 服务总线：把多种服务类型，挂载在服务总线上，通过总线上的服务治理，为应用和服务件提供服务访问管理功能。
  - 整车级系统服务：整车级系统服务具有全局性，即该类服务的设计更多关注是整车层面对整车内所有系统的通用化功能进行协同和管控，如整车健康管理服务、整车网络管理服务、整车时钟服务、整车电源管理服务等。

6.2.3. SOA 软件平台 ASF 层设计规范

6.2.3.1. 原子服务

原子服务是执行单一操作功能的服务，具有硬件功能上的不可拆分性。原子服务需要为上层提供完备的设备访问能力，而又隐藏硬件实现的细节，从而实现硬件逻辑和应用程序逻辑的分离。原子服务，应当具有以下的属性：

- 硬件控制
- 更改硬件工作参数

- 获取/通知硬件故障状态
- 故障自恢复
- 采集/发送硬件数据

通过调用原子服务，应用层可以实现硬件的控制功能、更改硬件的工作参数、获取硬件故障状态、收发硬件数据等功能，而无需关心硬件的初始化、反初始化、故障重启恢复等细节的具体实现。

6.2.3.2. 系统基础服务

系统基础服务描述车端各类域控及区域网关节点，基于通用基础软件（如 AUTOSAR AP/CP）提供的底层支持，进行相应的封装和扩展，实现各类通用化服务功能和框架及在此基础上形成的面向上层应用的各类服务接口（SDK 接口、API 接口、IPC 接口、RPC 接口等）。系统基础服务宜支持灵活组合、配置及部署，在不同的域控节点上部署不同的服务模块。

系统基础服务包括通用支撑类服务和公共框架类服务。其中通用支撑类服务宜包括服务治理（服务发布及发现）及服务容器、服务访问及限流降级、数据订阅及发布、集群管理等。公共框架类服务宜包括升级管理服务、健康管理服务、网络配置服务、资源管理服务、时钟同步服务、安全管理服务、电源管理服务、软件包管理服务、诊断服务等。

系统基础服务还包括针对具体域控节点的功能框架服务，如针对自动驾驶域控制器，提供自动驾驶融合感知模型和框架服务、规划控制模型和框架服务、决策执行模型及框架服务、定位服务等。针对智能座舱域控制器，提供手势识别服务、语音识别服务、仪表显示服务以及其他应用框架服务。

6.2.3.3. SOA 增强模块

SOA 增强服务是在 Adaptive AUTOSAR 基础平台进行服务框架扩展，封装通用化的基础功能。应用服务调用此类服务的接口方便完善其功能软件逻辑、便捷系统集成和敏捷测试。该类服务为一组服务集群，以 Lib 库的形式集成在应用服务中，并提供满足 Adaptive AUTOSAR 标准的服务接口，使接口标准完整统一。包含模块举例：服务调试、服务转换、服务权限、服务同步、SOA For Android 等。

6.2.3.4. 服务总线

服务总线是将各种服务和应用进行连接的桥梁，为不同的系统和域控制器提供服务的注册，发现功能，服务总线的主要设计规范如下：

协议转换：在面向服务的架构中，很有可能会出现各种协议，服务总线可以将来自不同服务的消息转换成统一的协议格式。这样就能避免不同协议之间的通信困难。

消息路由：服务总线可以帮助确定消息的目的地，它能够根据预定义的规则和条件选择正确的服务，并将消息传递给它。

消息转换：服务总线能够转换消息的格式，使得来自一个服务的消息可以被其他服务识别和使用。这能够降低服务之间的相关耦合度。

服务治理：服务总线能够确定服务的优先级，服务的访问范围，使得不同的应用可以访问到的服务范围有限，增强服务的安全性。

6.2.3.5. 整车级系统基础服务

整车级系统基础服务是将各控制器节点的能力，通过跨域、跨核组合成整车级别的业务功能，以对应用层提供整车级统一的调用。整车级基础服务包含：整车电源管理服务、整车健康管理服务、整车时钟服务、整车诊断 Master、整车版本管理服务、整车数据采集服务、整车日志管理服务等。

6.3. SOA 软件平台整车服务层设计规范

6.3.1. SOA 软件平台整车服务层概述

SOA 软件平台整车服务是指从整车的角度提取各 ECU 域控制器的通用和公共的部分，用于支持整车的各种功能和业务需求，实现整车的统一协调，管理，调度，控制等服务。

整车服务层是基于系统基础服务的实现，并对应用服务提供整车级通用服务的层级。整车服务层通常需要跨越多个功能域，实现多个功能域的组合复用性。

图 3 整车服务层结构

如图所示，整车服务主要包括了整车基础业务和功能安全协同服务、整车通信与网联协同功能服务、整车跨域协同服务、热管理组合服务、动力底盘组合服务等。主要是定义了车内软件系统中整车域需要实现和对上层提供的服务。

SOA 软件平台整车服务主要包含以下几个方面：

- 整车基础业务协同服务，定义整车基础和公共的服务，如整车级协同的电源管理，升级管理，日志管理，诊断管理等。
- 整车通信与网络协同服务，定义整车网络配置，网络调度，网络安全等与通信与网络相关的服务。
- 整车功能安全协同服务，定义整车功能安全相关的服务，如：健康监测，功能冗余，安全管理等。
- 整车跨域协同服务，定义需要在域间共享的服务，如自动驾驶整车服务，智能座舱整车服务，智能控制整车服务等。
- 其他整车服务，如热管理组合服务、动力底盘组合服务等。

6.3.2. SOA 软件平台整车服务层技术要求

对于整车级的服务的定义和实现，主要有以下技术要求：

- 标准化的通讯协议：整车服务层需要使用一种标准化的通讯协议，以便各个服务之间能够进行可靠的通讯和数据交换。常用的通讯协议包括 SOEMIP、DDS、RTSP 等。
- 可扩展性和灵活性：整车服务层需要具备良好的可扩展性和灵活性，以适应不断变化的业务需求和功能扩展。这包括支持动态添加和移除服务、动态调整服务的资源分配、动态配置和组合服务等能力。
- 可靠性和高可用性：整车服务层需要具备高可靠性和高可用性，以确保整车系统的稳定运行。这包括支持故障恢复、容错处理、负载均衡和故障切换等机制。
- 极致的性能：能够处理高并发和大数据的操作，满足整车系统的实时性和响应性。
- 可移植性和可维护性：良好的代码质量，统一的代码风格，编程规范，底层基于基础服务和 POSIX 操作系统接口，在 QNX、LINUX 等操作系统间提供较好的可移植性；有较好的故障码和错误日志定义。

- 整车级服务要具有较好的易用性，提供配套的设计与开发工具支持，提高应用层服务的集成开发效率。
- 安全性和权限控制：整车服务层需要具备强大的安全性和权限控制机制，以防止未经授权的访问和恶意攻击。这包括数据加密、身份认证、访问控制和审计日志等功能。
- 数据管理和分析：整车服务层需要具备强大的数据管理和分析能力，以便收集、存储和分析整车系统的各种数据，并为决策和优化提供支持。这包括数据存储、数据查询、数据挖掘和实时数据分析等功能。
- 故障诊断和远程管理：整车服务层需要提供故障诊断和远程管理功能，以便快速检测和解决整车系统中的故障，并通过远程管理接口进行远程配置和维护。
- 整车级服务的设计需满足 SOA 软件层级合理划分的要求，对下做到对系统服务的合理封装，对上需满足所有应用层服务的公共服务需求。
- 整车级服务的设计需要满足 SOA 的所有特性，包括高内聚松耦合特性、兼容特性、SOLID 原则、无状态原则、归一化原则等。
- 整车级服务的设计需满足系统无关性原则，整车服务往往需要应用于车内多个系统中，因此整车服务的设计与定义需要考虑到不同域、不同系统的差异性。

6.3.3.SOA 软件平台整车服务层设计规范

整车服务层的设计规范有助于确保系统的一致性、可用性和可维护性，同时也方便团队协作和项目的扩展。设计和定义整车级的服务，需要符合以下规范：

- 接口设计规范：定义清晰的接口规范，包括输入参数、输出结果、状态码和异常处理等。使用标准的命名规范和数据格式，以便不同服务之间能够进行无缝集成和交互。
- 服务命名规范：给每个服务指定一个唯一的、具有描述性的命名，以便开发人员和系统管理员能够清晰地理解其功能和用途。遵循一致的命名约定，提高代码的可读性和可维护性。
- 事件与方法定义规范：定义统一的事件和方法 ID。
- 异常处理规范：定义统一的异常处理机制，并规定不同异常类型的处理方式。例如，定义标准的错误码和错误信息，以便客户端能够正确地处理异常情况。
- 安全设计规范：在整车服务层中加入合适的安全机制，包括身份认证、权限控制和数据加密等。确保敏感数据的安全性和用户的合法访问。
- 性能优化规范：考虑到整车服务层需要处理大量的数据和请求，设计合理的缓存机制和查询优化策略，以提高系统的性能和响应速度。
- 文件和文档规范：为整车服务层编写清晰、易懂的代码注释，并准备好详细的文档和说明。描述服务的功能、接口和使用方法，以便其他开发人员和合作伙伴能够快速理解并正确使用服务。
- 整车级服务层的设计，数据的设计需满足一致性原则。整车服务会存在跨域交互和使用的情况，此时，数据的一致性要求才能保证不同域不同系统的功能符合预期且行为一致。
- 整车级服务层的设计，通信模块的设计需满足不同域不同通信协议的兼容性原则。由于整车服务面向的服务对象为车内不同域中的对象，因此整车服务的通信模块设计之初就必须考虑整车系统内所使用的所有通信协议的兼容性和同步性等问题，并且还需考虑通信协议的可扩展性。
- 整车级服务层的设计，对于服务访问的安全问题需要做特别的管理，由于整车级的服务往往关联了车内关键的功能和领域，因此服务访问的权限管理需在设计时重点考虑。
- 整车级服务层的设计，要通过访问基础服务层，设备抽象层，POSIX 接口函数集等，提高服务层的可移植性，尽量采用一套代码来适配不同的操作系统，提高服务的可维护性。
- 整车级服务层的设计，要考虑功能安全和信息安全需求，设计时需考虑： E2E 安全机制，配置和数据安全，必要的超时和异常监测处理，安全日志等。

6.4. SOA 软件平台车云一体层设计规范

6.4.1. SOA 软件平台车云一体层概述

软件平台车云一体层是指车端各 ECU 与云端生态能力之间，基于 SOA 服务化的设计思想，构建支撑整车智能化应用场景的平台化软件架构。车云一体是由面向服务的基础运行环境和面向服务的开发环境的一整套软件框架，包括动态服务（车云协同服务）和功能型服务（车云系统服务）两大类。它可以快速的将云端原子能力和车端原子能力通过服务化的方式进行组合的工具链构成。基于稳定、高效、安全运行的服务自由调度的运行环境和降低开发难度提升效率的工具链，可以实现从服务定义、编排到智能化场景运行支撑的全链路车云一体解决方案。

6.4.2. SOA 软件平台车云一体层技术要求

基于车云一体的架构设计，有以下关键技术要求：

1) 车云能力具备服务化技术设计思想：从用户出发，分析不同用户现实需求，挖掘用户潜在需要， 结合车辆未来的发展以及行业经验，集成云端的人工智能、大数据等云端技术，在抽象形成的车云软件组件 基础上进一步抽象，形成车云一体的服务。
2) 车云一体满足服务化通讯设计要求：对车端和云端的通信进行合理规划，对通信数据从大小、安 全性、QoS、时延、驹应要求等方面进行分类，按照不同的要求和应用建立不同的通信链路，提供通信 能力服务；
3) 车云一体服务化满足通信安全要求：对于车云通信链路进行安全防护。不仅仅需要考虑数据传输安全，还需要考虑车辆接入安全、通信链路安全、访问安全，也需要从进程防护、文件加固、流量监控、GNSS 欺骗防护、车内流量监控、侵入检测、恶意攻击防护等方面进行系统、全面的安全防护；
4) 车云能力具备同步技术：在车云基础通信服务之上，进行车端能力和云端能力的同步，将车端能力反馈到云端，将云端适合于某一个车辆的能力下发到车端，进行车云能力状态的管理和同步， 在车端形成其使用能力的合集，在云端形成车辆的影子设备。

6.4.3. SOA 软件平台车云一体层设计规范

车云一体的核心实现路径包括以下三点：

1) **横向——跨域**:针对不同的 E/E 架构（尤其在中央域控架构下），通过解决车内横向跨域的问题，构建整车集中的软件平台，是车云一体架构中车端面向云端能力融合的基础。
2) **纵向——跨车云**：基于车内功能统一整合和可开放的能力，通过与云端的打通实现车云之间数 据和服务的交互，从而充分利用云端的计算能力和生态服务能力提升智能化。
3) **关键技术**：基于 SOA 的设计思想将云端和车端能力以服务化的方式进行封装，进而简化应用的开发效率以及体现服务自由组合能力实现场景灵活可变的用户体验。

车云一体开发环境主要面向于普通开发者、专业开发者、第三方开发者，提供包括服务定义、服务开发、服务编排、仿真测试、宙批发布、数据运行的一整套工具链。

车云一体运行环境主要分布在车端软件平台和云端软件平台。针对不同的 E/E 架构，布置在定位为整车级管理的域控制器中，作为车内跨域融合与车云的交互中央交互单元，进行集中式管理。

车端部分主要包括动态服务引擎，以及为动态服务引擎提供车辆硬件能力的静态服务（原子/组合服务）和管理服务自由调度分配的服务调度管理平台。同时基于车内基础软件和中间件的系统和整车管理能力，与云端服务能力进行融合。

云端部分主要包括云端核心服务模块，提供动态可配置的全 ECU 的数据采集、OTA、诊断等远程

管理能力的车辆远程服务平台，以及提供云端生态能力的云服务（提供 CP/SP 服务、V2X、IOT 服务等）。

车云一体服务具备调度和协调的能力：车云能力虽然对使用者来讲没有车云分离概念，但是在事实上是存在云和车的区别。为了使用户和开发者对车云服务无感，首先将抽象的车云软件组件调度协调，形成车云一体化软件组件。将车端能力和云端能力进行有机整合，实现双向远程调用。

6.5. SOA 软件平台云端服务层设计规范

6.5.1. SOA 软件平台云端服务层概述

从服务的整体结构上讲，云端服务作为服务器端，车端以及车端控制设备(如手机等)作为客户端。服务端与客户端进行数据交互并管理客户端。云端服务定义 API 与客户端进行通信，主要关注使用的传输协议，消息格式，数据编码等逻辑。云端服务内部关注服务调度，数据存储和数据处理等逻辑。云端可以提供多种性质不同的服务，服务的设计实现可能需要依赖第三方软件，使用的开发环境也可能不尽相同，整合这些软件依赖以及适配不同的开发环境也是云端服务需要考虑的因素。云端服务的开发和运行环境包括 CPU 架构，操作系统，编译构建工具，编程语言等要素。典型的云端服务包括：人工智能服务（AI）、OTA 服务、大规模数据存储和处理、IOT 接入服务、V2X 云服务等。

6.5.2. SOA 软件平台云端服务层技术要求

云端服务的技术要求主要包括：

- 云端服务高可用性
  支持根据服务负载，进行服务实例的弹性伸缩。支持服务实例的失败恢复和快速部署。
- 云端连接管理多样性
  不同的服务可能使用的不同的网络层级。支持不同网络协议(例如 TCP, UDP 或者 HTTPs)上进行的连接管理。
- 云端通讯的安全性
  支持对传输数据的加密，完整性校验。对用户访问进行权限管理。
- 云端服务的实时性
  支持复杂逻辑和大算力的实时服务。例如特定逻辑的搜索，以及 AI 对话等。
- 云端服务的可调度
  通过云端服务的解耦，实现不同服务的调度和管理。

6.5.3. SOA 软件平台云端服务层设计规范

云端服务层的整体结构可以参考如下：

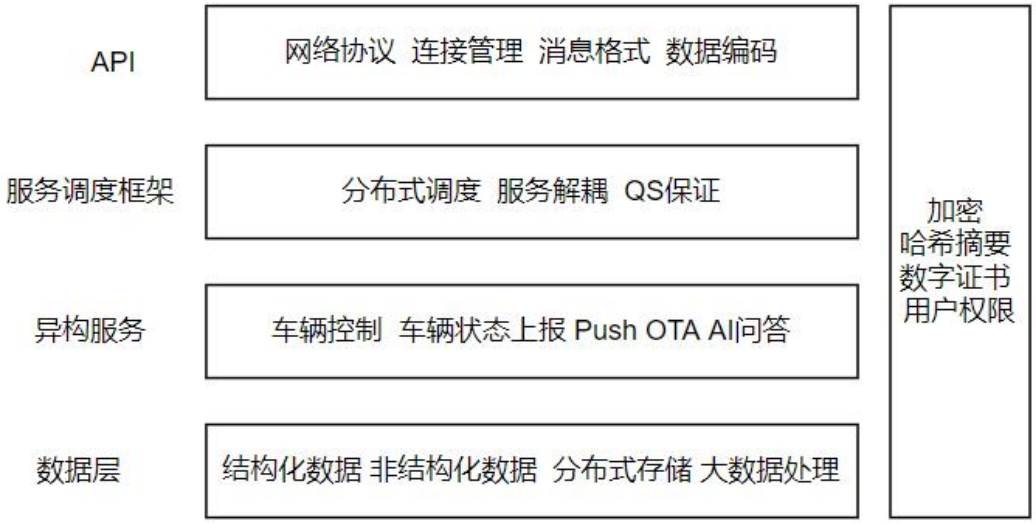

图 4 云端服务层结构

API 层是云端服务和客户端的接口，其主要的设计建议包括：

- 传输网络选择

  与移动设备进行通信时，适合选择较上层的传输协议，比如 HTTP 协议，可以比较好的兼顾开发效率。与车辆设备进行通信时，适合选择较底层的传输协议，比如 TCP 或者 UDP，可以较好的兼顾传输性能。

- 消息格式设计

  对于偏上层的业务消息或者传输量小的消息，可以考虑更好可读性的格式，如 JSON。对于偏底层的消息或者传输量大的消息，可以考虑传输性能更好的二进制格式。

- 传输效率

  影响数据在网络上的传输效率的因素很多，包括方式端和接收端的处理逻辑，缓存配置，传输网络的选择，消息格式设计，消息大小，消息的接收/方式模型等。

- 传输安全

  数据在网络中传输时，不能被第三方浏览内容和篡改内容，因此需要对数据进行加密和完整性校验。

- 服务安全

  需要有效的处理对服务的攻击和探测。如防止 DDoS 攻击，回放攻击等。

服务调度框架能无缝管理单机环境以及分布式环境中的服务。其主要内容包括，服务路由，服务实例运行状态监控，失败恢复，根据负载进行服务实例的横向扩展，以及服务结果聚合等。服务调度框架主要的设计建议包括：

- 微服务

  可以参考微服务的思想进行服务基础组件设计，包括服务网关，服务注册，服务发现等微服务组件，达到服务运行解耦，也可以提高前期的开发效率。

- 服务质量

  微服务中的网关组件可以根据负载进行熔断，降级等处理。调度框架支持不同的调度策略，如轮询调度，优先级调度，最少使用调度等机制。调度框架可以管理系统资源，如 CPU，GPU，内存等，为服务运行提供匹配的系统资源。

- 服务监控

  为了更好的管理云端服务，需要提供服务和系统资源的监控系统，可以可视化展示实时状态，并且支持管理员权限下的配置维护。

业务服务层包含各种不同服务的具体执行逻辑。不同的服务有不同的功能，以及可能的运行环境。

数据层主要包括数据存储和数据处理。不同的业务场景可能产生不同类型的数据，包括结构化数据，以及图片，音视频等非结构化数据。服务可以对输入的原始数据进行数据治理，在原始数据的基

础上产生不同的主题数据模块。数据层的设计建议包括：

- 数据存储

  数据存储系统需要考虑数据量大小，数据类型，数据安全性，存储系统的操作维护便利性，以及存储系统和数据处理框架和工具的匹配性等因素。

- 数据安全

  保证数据在存储设备失效时，不能丢失数据。需要存储系统根据数据安全等级提供相应的同步写入保证机制和副本机制。数据安全也包括数据访问权限管理，防止非法用户访问敏感数据。

- 数据复用

  服务对原始数据进行处理后，形成不同层次的主题数据，这些主题数据，可能是单独存储，也可能是对其他主题数据的引用。根据业务场景设计数据复用机制，数据复用会影响到存储空间的利用，以及软件层次的数据建模，如数据库的表关联等。

- 数据访问效率

  服务对其数据的访问效率可能有不同的要求，对于热点数据或者高时效性数据可能需要更高的访问效率。数据访问效率的设计需要考虑硬件和软件因素，例如可以使用 IOPS 更高的 SSD 存储热点数据，并且在软件层面，可以基于 redis 或者 memcached 等组件进行分级缓存设计。