**imageNeusoft Confidential**

File No. : ProjectNo.- D00-D01-T01-流水号

NeuSAR aCore

软件需求规格说明书

（Software requirement specification）

**(CM API)**

Neusoft Reach Automotive Technology Co., Ltd

Change Log

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Version** | **Contents Revised** | **Status** | **Reviser/**  **Date** | **Approve/Date** |
| 1 | 0.5 | 新建 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Contents**

[1 引言（Introduction） 1](#_Toc74502117)

[1.1 目的（Goal） 1](#_Toc74502118)

[1.2 范围（Scope） 1](#_Toc74502119)

[1.3 参考文档（Reference） 1](#_Toc74502120)

[1.4 术语及缩略语（Terms And Abbreviations） 1](#_Toc74502121)

[2 软件系统概述（Software System Overview） 2](#_Toc74502122)

[2.1 软件系统背景（Software System Background） 2](#_Toc74502123)

[2.2 软件系统目标（Software System Goal） 2](#_Toc74502124)

[2.3 外部关联 （External Association） 2](#_Toc74502125)

[3 功能需求（Functional Requirement） 2](#_Toc74502126)

[3.1 offer service(offer service) 2](#_Toc74502127)

[3.1.1 [SWRD\_CM\_CommunicationAPI\_00001]保证本机唯一offer service (Uniqueness of offered service on local machine) 2](#_Toc74502128)

[3.1.2 [SWRD\_CM\_CommunicationAPI\_00002] offer service的部署协议(Protocol where a service is offered) 2](#_Toc74502129)

[3.2 构造Service skeleton (Service skeleton creation) 3](#_Toc74502130)

[3.2.1 [SWRD\_CM\_CommunicationAPI\_00003] InstanceIdentifier 构造Service skeleton(InstanceIdentifier check during the creation of service skeleton) 3](#_Toc74502131)

[3.2.2 [SWRD\_CM\_CommunicationAPI\_00004] InstanceSpecifier 构造Service skeleton(InstanceSpecifier check during the creation of service skeleton) 3](#_Toc74502132)

[3.2.3 [SWRD\_CM\_CommunicationAPI\_00005] InstanceIdentifierContainer 构造Service skeleton(InstanceIdentifierContainer check during the creation of service skeleton) 4](#_Toc74502133)

[3.3 service methods处理 (Processing of service methods) 4](#_Toc74502134)

[3.3.1 [SWRD\_CM\_CommunicationAPI\_00006] Service methods 的处理方式(Service method processing modes) 4](#_Toc74502135)

[3.4 field注册get/set handler (Registering get/set handlers for fields) 5](#_Toc74502136)

[3.4.1 [SWRD\_CM\_CommunicationAPI\_00007] 调用GetHandlers (Invoking GetHandlers) 5](#_Toc74502137)

[3.4.2 [SWRD\_CM\_CommunicationAPI\_00008] 调用SetHandlers (Invoking SetHandlers) 5](#_Toc74502138)

[3.4.3 [SWRD\_CM\_CommunicationAPI\_00009] 调用SetHandlers之后Notify Field的值 (Notify the Field value after a call to the Set Handler function) 6](#_Toc74502139)

[3.4.4 [SWRD\_CM\_CommunicationAPI\_00010] 保证filed有初始值(Ensuring the existence of valid Field values) 6](#_Toc74502140)

[3.4.5 [SWRD\_CM\_CommunicationAPI\_00011] 保证filed有SetHandler (Ensuring the existence of SetHandler) 7](#_Toc74502141)

[3.5 find service (find service) 7](#_Toc74502142)

[3.5.1 [SWRD\_CM\_CommunicationAPI\_00012] 调用Find service handler (Find service handler invocation) 7](#_Toc74502143)

[3.5.2 [SWRD\_CM\_CommunicationAPI\_00013] 调用stop find service (Calling stop find service for already stopped finds) 8](#_Toc74502144)

[3.6 events接收(Receive events) 8](#_Toc74502145)

[3.6.1 [SWRD\_CM\_CommunicationAPI\_00014] event接收 (event receive) 8](#_Toc74502146)

[3.6.2 [SWRD\_CM\_CommunicationAPI\_00015] event接收回调 (Receive event by getting triggered) 9](#_Toc74502147)

[3.7 调用service method(call service method) 9](#_Toc74502148)

[3.7.1 [SWRD\_CM\_CommunicationAPI\_00016] method 调用 (method call) 9](#_Toc74502149)

[3.8 更新filed中的event(Update notification events for fields) 10](#_Toc74502150)

[3.8.1 [SWRD\_CM\_CommunicationAPI\_00017] 更新filed中的event (Update notification events for fields) 10](#_Toc74502151)

[3.9 Instance Specifier 解析(Instance Specifier Translation) 10](#_Toc74502152)

[3.9.1 [SWRD\_CM\_CommunicationAPI\_00018] InstanceSpecifier 转换 InstanceIdentifiers (InstanceSpecifier translation to InstanceIdentifiers) 10](#_Toc74502153)

[3.10 无效值 (Invalid Value) 11](#_Toc74502154)

[3.10.1 [SWRD\_CM\_CommunicationAPI\_00019] 无效值实现 (Implementation of invalidValue) 11](#_Toc74502155)

[3.11 API头文件（API Header files） 11](#_Toc74502156)

[3.11.1 [SWRD\_CM\_CommunicationAPI\_00020] 文件夹结构 (Folder structure) 11](#_Toc74502157)

[3.12 服务头文件（Service header files） 12](#_Toc74502158)

[3.12.1 [SWRD\_CM\_CommunicationAPI\_00021] Service头文件(Service header files) 12](#_Toc74502159)

[3.12.2 [SWRD\_CM\_CommunicationAPI\_00022] Common头文件 (Common header file) 12](#_Toc74502160)

[3.12.3 [SWRD\_CM\_CommunicationAPI\_00023] Service头文件命名空间 (Namespace of Service header files) 13](#_Toc74502161)

[3.12.4 [SWRD\_CM\_CommunicationAPI\_00024] Service类命名空间 (namespace for a specific service) 13](#_Toc74502162)

[3.13 Common头文件（Common header file） 14](#_Toc74502163)

[3.13.1 [SWRD\_CM\_CommunicationAPI\_00025] Common头文件描述 (Common header file) 14](#_Toc74502164)

[3.13.2 [SWRD\_CM\_CommunicationAPI\_00026] Common头文件中类型定义和命名空间(Type definitions and namespaces in common header files) 14](#_Toc74502165)

[3.14 Types头文件（Types header file） 15](#_Toc74502166)

[3.14.1 [SWRD\_CM\_CommunicationAPI\_00026] Types头文件中类型定义和命名空间(Type definitions and namespaces in types header files) 15](#_Toc74502167)

[3.15 Implementation头文件（Implementation Types header files） 16](#_Toc74502168)

[3.15.1 [SWRD\_CM\_CommunicationAPI\_00027] Implementation头文件中类型定义和命名空间(Type definitions and namespaces in implementation header files) 16](#_Toc74502169)

[3.16 服务标识符数据类型（Service Identifier Data Types） 16](#_Toc74502170)

[3.16.1 [SWRD\_CM\_CommunicationAPI\_00028] 服务ID，服务版本，服务合同(Service Identifier, Service Version Classes and Service Contract Version) 16](#_Toc74502171)

[3.16.2 [SWRD\_CM\_CommunicationAPI\_00029] InstanceIdentifier类(Instance Identifier Class) 17](#_Toc74502172)

[3.16.3 [SWRD\_CM\_CommunicationAPI\_00030] FindServiceHandle类(FindServiceHandle) 18](#_Toc74502173)

[3.16.4 [SWRD\_CM\_CommunicationAPI\_00031] Handle Type类(Handle Type Class) 19](#_Toc74502174)

[3.16.5 [SWRD\_CM\_CommunicationAPI\_00032] Handle Type类的构造和赋值函数(Constructors and assignment functions for the handle type class) 19](#_Toc74502175)

[3.16.6 [SWRD\_CM\_CommunicationAPI\_00033] Service处理容器(Service Handle Container) 20](#_Toc74502176)

[3.16.7 [SWRD\_CM\_CommunicationAPI\_00034] Find服务处理程序(Find Service Handler) 20](#_Toc74502177)

[3.17 与事件相关数据类型（Event Related Data Types） 21](#_Toc74502178)

[3.17.1 [SWRD\_CM\_CommunicationAPI\_00035] Sample指针(Sample Pointer) 21](#_Toc74502179)

[3.17.2 [SWRD\_CM\_CommunicationAPI\_00036] 数据类型(Data Types) 22](#_Toc74502180)

[3.18 与方法相关数据类型（Method Related Data Types） 23](#_Toc74502181)

[3.18.1 [SWRD\_CM\_CommunicationAPI\_00037] 方法调用处理方式(Method Call Processing Mode) 23](#_Toc74502182)

[3.19 错误类型（Error Types） 24](#_Toc74502183)

[3.19.1 [SWRD\_CM\_CommunicationAPI\_00038] CM ErrorDomain (CM ErrorDomain) 24](#_Toc74502184)

[3.20 E2E关联数据类型（E2E Related Data Types） 24](#_Toc74502185)

[3.20.1 [SWRD\_CM\_CommunicationAPI\_00039 方法调用处理方式(Method Call Processing Mode) 24](#_Toc74502186)

[3.21 API概述(API General) 25](#_Toc74502187)

[3.21.1 [SWRD\_CM\_CommunicationAPI\_00040] Skeleton (Skeleton) 25](#_Toc74502188)

[3.21.2 [SWRD\_CM\_CommunicationAPI\_00041] Proxy (Proxy) 26](#_Toc74502189)

[3.21.3 [SWRD\_CM\_CommunicationAPI\_00042] 可重入 (Re-entrancy - General) 26](#_Toc74502190)

[3.22 Named构造(Object Creation via Named Constructor Approach) 27](#_Toc74502191)

[3.22.1 [SWRD\_CM\_CommunicationAPI\_00043] Named构造 (Named Constructor Approach ) 27](#_Toc74502192)

[3.23 offer service方法 (offer service method) 27](#_Toc74502193)

[3.23.1 [SWRD\_CM\_CommunicationAPI\_00044]offer service方法 (offer service method) 27](#_Toc74502194)

[3.24 stop offer service方法 (stop offer service method) 28](#_Toc74502195)

[3.24.1 [SWRD\_CM\_CommunicationAPI\_00045]stop offer service方法 (stop offer service method) 28](#_Toc74502196)

[3.25 创建Service skeleton(Service skeleton creation) 28](#_Toc74502197)

[3.25.1 [SWRD\_CM\_CommunicationAPI\_00046]使用InstancID创建service skeleton (Creation of service skeleton using Instance ID) 28](#_Toc74502198)

[3.25.2 [SWRD\_CM\_CommunicationAPI\_00047]使用InstancID创建service skeleton (Creation of service skeleton using Instance ID) 29](#_Toc74502199)

[3.25.3 [SWRD\_CM\_CommunicationAPI\_00048使用InstancSpecifier建service skeleton (Creation of service skeleton using Instance Spec) 30](#_Toc74502200)

[3.25.4 [SWRD\_CM\_CommunicationAPI\_00050]使用InstancID Container创建service skeleton (Creation of service skeleton using Instance ID Container) 30](#_Toc74502201)

[3.25.5 [SWRD\_CM\_CommunicationAPI\_00049]使用InstancSpecifier创建service skeleton (Creation of service skeleton using Instance Spec) 30](#_Toc74502202)

[3.25.6 [SWRD\_CM\_CommunicationAPI\_00051]使用InstancID Container创建service skeleton (Creation of service skeleton using Instance ID Container) 31](#_Toc74502203)

[3.25.7 [SWRD\_CM\_CommunicationAPI\_00052]service skeleton的拷贝控制 (service skeleton copy and move) 32](#_Toc74502204)

[3.26 发送event(Send event) 32](#_Toc74502205)

[3.26.1 [SWRD\_CM\_CommunicationAPI\_00053]Event数据由application提供(Send event where application is responsible for the data) 32](#_Toc74502206)

[3.26.2 [SWRD\_CM\_CommunicationAPI\_00054]Event数据由CM提供(Send event where CM is responsible for the data) 33](#_Toc74502207)

[3.26.3 [SWRD\_CM\_CommunicationAPI\_00055]Event数据的申请(Allocating data for event transfer) 33](#_Toc74502208)

[3.27 service method (Provide a service method) 34](#_Toc74502209)

[3.27.1 [SWRD\_CM\_CommunicationAPI\_00056]method规定(Provision of method) 34](#_Toc74502210)

[3.27.2 [SWRD\_CM\_CommunicationAPI\_00057]Fire and Forget method规定(Provision of a Fire and Forget method) 35](#_Toc74502211)

[3.27.3 [SWRD\_CM\_CommunicationAPI\_00058]method实现的可重入(Re-entrancy - ServiceSkeleton method implementation) 35](#_Toc74502212)

[3.28 service method处理 (Processing of service methods) 36](#_Toc74502213)

[3.28.1 [SWRD\_CM\_CommunicationAPI\_00059]method处理模式(Set service method processing mode) 36](#_Toc74502214)

[3.28.2 [SWRD\_CM\_CommunicationAPI\_00060]method调用处理(Process Service method invocation) 36](#_Toc74502215)

[3.29 注册field的get handlers (Registering get handlers for fields) 37](#_Toc74502216)

[3.29.1 [SWRD\_CM\_CommunicationAPI\_00061]注册Getters(Registering Getters) 37](#_Toc74502217)

[3.29.2 [SWRD\_CM\_CommunicationAPI\_00062]提供Execution Context注册Getters(Execution Context for registering Getters) 37](#_Toc74502218)

[3.29.3 [SWRD\_CM\_CommunicationAPI\_00063] Getters通用(Getters General) 38](#_Toc74502219)

[3.30 注册field的set handlers (Registering set handlers for fields) 38](#_Toc74502220)

[3.30.1 [SWRD\_CM\_CommunicationAPI\_00064]注册Setters(Registering Setters) 38](#_Toc74502221)

[3.30.2 [SWRD\_CM\_CommunicationAPI\_00065]提供Execution Context注册Setters(Execution Context for registering Setters) 39](#_Toc74502222)

[3.30.3 [SWRD\_CM\_CommunicationAPI\_00066] Setters通用(Setters General) 39](#_Toc74502223)

[3.30.4 [SWRD\_CM\_CommunicationAPI\_00067] Update功能(Update Function) 40](#_Toc74502224)

[3.31 find service(Find service) 40](#_Toc74502225)

[3.31.1 [SWRD\_CM\_CommunicationAPI\_00068]find service立即返回，使用Instance ID(Find service with immediately returned request using Instance ID) 40](#_Toc74502226)

[3.31.2 [SWRD\_CM\_CommunicationAPI\_00069]find service立即返回，使用Instance Specifier(Find service with immediately returned request using Instance Specifier) 41](#_Toc74502227)

[3.31.3 [SWRD\_CM\_CommunicationAPI\_00070]find service可重入性(Re-entrancy - FindService) 42](#_Toc74502228)

[3.31.4 [SWRD\_CM\_CommunicationAPI\_00071]find service注册通知，使用Instance ID(Find service with handler registration using Instance ID) 42](#_Toc74502229)

[3.31.5 [SWRD\_CM\_CommunicationAPI\_00072]提供Execution Context find service注册通知，使用Instance ID(Execution Context for finding service with handler registration using Instance ID) 43](#_Toc74502230)

[3.31.6 [SWRD\_CM\_CommunicationAPI\_00073]find service注册通知，使用Instance Specifier(Find service with handler registration using Instance Specifier) 43](#_Toc74502231)

[3.31.7 [SWRD\_CM\_CommunicationAPI\_00074]start find service可重入性(Re-entrancy - StartFindService) 44](#_Toc74502232)

[3.31.1 [SWRD\_CM\_CommunicationAPI\_00075]stop find service (Stop find service) 44](#_Toc74502233)

[3.32 Service proxy构造(Service proxy creation) 45](#_Toc74502234)

[3.32.1 [SWRD\_CM\_CommunicationAPI\_00076]service proxy构造函数(Creation of service proxy) 45](#_Toc74502235)

[3.32.2 [SWRD\_CM\_CommunicationAPI\_00077]不抛异常构造service proxy(Exception-less creation of service proxy) 45](#_Toc74502236)

[3.32.3 [SWRD\_CM\_CommunicationAPI\_00078] GetHandle (GetHandle function to return the proxy instance creation handle) 46](#_Toc74502237)

[3.32.4 [SWRD\_CM\_CommunicationAPI\_00079]service proxy的拷贝控制 (service proxy copy and move) 46](#_Toc74502238)

[3.32.5 [SWRD\_CM\_CommunicationAPI\_00080]service proxy重新连接 (Re-establishing service connection) 47](#_Toc74502239)

[3.32.6 [SWRD\_CM\_CommunicationAPI\_00081]service proxy析构 (Destruction of service proxy) 47](#_Toc74502240)

[3.33 event订阅 (Service event subscription) 48](#_Toc74502241)

[3.33.1 [SWRD\_CM\_CommunicationAPI\_00082]event订阅方法(Method to subscribe to a service event) 48](#_Toc74502242)

[3.33.2 [SWRD\_CM\_CommunicationAPI\_00083]event取消订阅方法(Method to unsubscribe from a service event) 48](#_Toc74502243)

[3.33.3 [SWRD\_CM\_CommunicationAPI\_00084]查询订阅状态 (Query Subscription State) 49](#_Toc74502244)

[3.33.4 [SWRD\_CM\_CommunicationAPI\_00085]注册订阅状态变化通知 (Set Subscription State change handler) 49](#_Toc74502245)

[3.33.5 [SWRD\_CM\_CommunicationAPI\_00086] Execution Context注册订阅状态变化通知 (Execution Context for setting Subscription State change handler) 50](#_Toc74502246)

[3.33.6 [SWRD\_CM\_CommunicationAPI\_00087] SetSubscriptionStateChangeHandler 可重入性(Re-entrancy - SetSubscriptionStateChangeHandler) 50](#_Toc74502247)

[3.33.7 [SWRD\_CM\_CommunicationAPI\_00088]取消注册订阅状态变化通知 (Unset Subscription State change handler) 51](#_Toc74502248)

[3.33.8 [SWRD\_CM\_CommunicationAPI\_00089]调用订阅状态变化通知 (Call Subscription State change handler) 51](#_Toc74502249)

[3.34 event接收 (Receive event) 52](#_Toc74502250)

[3.34.1 [SWRD\_CM\_CommunicationAPI\_00090]更新event cache (Method to update the event cache) 52](#_Toc74502251)

[3.34.2 [SWRD\_CM\_CommunicationAPI\_00091] Execution Context更新event cache (Execution Context to update the event cache) 53](#_Toc74502252)

[3.34.3 [SWRD\_CM\_CommunicationAPI\_00092] GetNewSamples可重入性(Re-entrancy - GetNewSamples) 53](#_Toc74502253)

[3.34.4 [SWRD\_CM\_CommunicationAPI\_00093] 提供获取E2E Result方法(Provide E2E Result) 54](#_Toc74502254)

[3.34.5 [SWRD\_CM\_CommunicationAPI\_00094] 提供获取event缓冲区剩余空间方法(Query Free Sample Slots) 54](#_Toc74502255)

[3.34.6 [SWRD\_CM\_CommunicationAPI\_00095] 提供注册event接收通知方法(Enable service event trigger) 55](#_Toc74502256)

[3.34.7 [SWRD\_CM\_CommunicationAPI\_00096] Execution Context提供注册event接收通知方法(Execution Context for enabling service event trigger) 55](#_Toc74502257)

[3.34.1 [SWRD\_CM\_CommunicationAPI\_00097] SetReceiveHandler可重入性(Re-entrancy - SetReceiveHandler) 56](#_Toc74502258)

[3.34.2 [SWRD\_CM\_CommunicationAPI\_00098] 提供反注册event接收通知方法(Disable service event trigger) 56](#_Toc74502259)

[3.35 method调用 (Call a service method) 57](#_Toc74502260)

[3.35.1 [SWRD\_CM\_CommunicationAPI\_00099] method类定义 (Initiate a method call) 57](#_Toc74502261)

[3.35.2 [SWRD\_CM\_CommunicationAPI\_00100] Fire and Forget method类定义 (Initiate a Fire and Forget method call) 58](#_Toc74502262)

[3.36 field方法 (method for fields) 59](#_Toc74502263)

[3.36.1 [SWRD\_CM\_CommunicationAPI\_00101]field get方法(Get method for fields) 59](#_Toc74502264)

[3.36.2 [SWRD\_CM\_CommunicationAPI\_00102]field set方法(Set method for fields) 59](#_Toc74502265)

[3.37 Instance Specifier转换 (Instance Specifier Translation) 60](#_Toc74502266)

[3.37.1 [SWRD\_CM\_CommunicationAPI\_00103] Instance Specifier转换方法(Method Instance Specifier Translation) 60](#_Toc74502267)

[4 非功能需求(Non-Functional Requirements) 60](#_Toc74502268)

[4.1 制约（Constraint） 60](#_Toc74502269)

[4.1.1 [SWRD\_NF\_CM\_CommunicationAPI\_00001]异步回调(Non-Function Requirement No.1) 60](#_Toc74502270)

[5 接口说明（API） 61](#_Toc74502271)

[5.1 错误类型(Error types) 61](#_Toc74502272)

[5.1.1 [SWRD\_API\_CM\_CommunicationAPI\_00001]ComErrc枚举（ComErrc enum） 61](#_Toc74502273)

[5.1.2 [SWRD\_API\_CM\_CommunicationAPI\_00002]CM异常类（Com Exception） 61](#_Toc74502274)

[5.1.3 [SWRD\_API\_CM\_CommunicationAPI\_00003] ComException构造函数（ComException Construct） 62](#_Toc74502275)

[5.1.4 [SWRD\_API\_CM\_CommunicationAPI\_00004] ComErrorDomain类（Class ComErrorDomain） 62](#_Toc74502276)

[5.1.5 [SWRD\_API\_CM\_CommunicationAPI\_00005] Errc别名(using Errc) 63](#_Toc74502277)

[5.1.6 [SWRD\_API\_CM\_CommunicationAPI\_00006] ComErrorDomain构造函数（ComErrorDomain constructor） 63](#_Toc74502278)

[5.1.7 [SWRD\_API\_CM\_CommunicationAPI\_00007] Exception别名（using Exception） 63](#_Toc74502279)

[5.1.8 [SWRD\_API\_CM\_CommunicationAPI\_00008] name函数（name function） 64](#_Toc74502280)

[5.1.9 [SWRD\_API\_CM\_CommunicationAPI\_00009] message函数(message function) 64](#_Toc74502281)

[5.1.10 [SWRD\_API\_CM\_CommunicationAPI\_00010] ThrowAsException函数(ThrowAsException function) 65](#_Toc74502282)

[5.1.11 [SWRD\_API\_CM\_CommunicationAPI\_00011] GetComErrorDomain函数(GetComErrorDomain function) 65](#_Toc74502283)

[5.1.12 [SWRD\_API\_CM\_CommunicationAPI\_00012] MakeErrorCode函数(MakeErrorCode function) 66](#_Toc74502284)

[5.1.13 [SWRD\_API\_CM\_CommunicationAPI\_00013] 应用程序错误的定义(Definition of Application Errors) 67](#_Toc74502285)

[5.2 接口头文件（API Header files） 67](#_Toc74502286)

[5.3 接口共同数据类型（API Common Data Types） 67](#_Toc74502287)

[5.4 接口定义（API Reference） 67](#_Toc74502288)

[5.4.1 [SWRD\_ID]接口1（API 1） 67](#_Toc74502289)

[5.4.2 [SWRD\_ID]接口2（API 2） 68](#_Toc74502290)

[附录A- 信息定义 69](#_Toc74502291)

[附录B- 配置信息 71](#_Toc74502292)

# 引言（Introduction）

## 目的（Goal）

本文是对AUTOSAR ADAPTIVE CM模块CommunicationGroup具体需求。

## 范围（Scope）

使用于CM软件开发，测试，管理，人员。

## 参考文档（Reference）

|  |  |  |
| --- | --- | --- |
| **序号（No.）** | **文档名（Document Name）** | **版本名（Revision）** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 术语及缩略语（Terms And Abbreviations）

|  |  |
| --- | --- |
| **术语**  **Term/Abbreviation** | **描述（Description）** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 软件系统概述（Software System Overview）

## 软件系统背景（Software System Background）

NeuSAR aCore平台以面向服务的方式相互通信，CM负责实现面向服务的通信，提供统一的API接口定义，不依赖具体的通信协议部署。通信协议包括SOME/IP，DDS。

## 软件系统目标（Software System Goal）

符合AUTOSAR 21-11标准CM的规范。

## 外部关联 （External Association）

无

# 功能需求（Functional Requirement）

## 服务合约版本控制（Service contract versioning）

### [SWRD\_CM\_CommunicationAPI\_00001] 服务合约版本控制（Service contract versioning）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00001 |
| **Type** | Valid |
| **Description** | ServiceInterfaceDeployment 的版本应由服务发现根据用于服务连接的网络绑定在向后兼容性方面进行评估。 |
| **Upstream ID** | [SWS\_CM\_99003] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 不变 |

## offer service(offer service)

### [SWRD\_CM\_CommunicationAPI\_00002]保证本机唯一offer service (Uniqueness of offered service on local machine)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00002 |
| **Type** | Valid |
| **Description** | 保证本机内在offer service时，有唯一的serviceInstanceId, serviceInterfaceId and majorVersion。如果有重复的，不启动offer service，并log记录。 |
| **Upstream ID** | [SWS\_CM\_00102] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用offer service接口 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00003] offer service的部署协议(Protocol where a service is offered)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00003 |
| **Type** | Valid |
| **Description** | ServiceInterfaceDeployment配置的协议，与offer service发出的一致 |
| **Upstream ID** | [SWS\_CM\_00103] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用offer service接口，通过wireshark检查相应的部署协议 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00004] StopOfferService (StopOfferService)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00004 |
| **Type** | Valid |
| **Description** | 调用StopOfferService时，应检查所停止服务的网络绑定，此信息在ServiceInterfaceDeployment中配置。  如果是Someip，则someip binding应根据[SWS\_CM\_00204]描述处理该调用。  如果时dds，则dds binding应根据[SWS\_CM\_11005]描述处理该调用。  如果时user define，则应根据具体的实现处理该调用。 |
| **Upstream ID** | [SWS\_CM\_00104] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用StopOfferService接口，通过wireshark检查相应的部署协议 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## 构造Service skeleton (Service skeleton creation)

### [SWRD\_CM\_CommunicationAPI\_00005] InstanceIdentifier 构造Service skeleton(InstanceIdentifier check during the creation of service skeleton)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00005 |
| **Type** | Valid |
| **Description** | 提供InstanceIdentifier构造Service skeleton时，要检查InstanceIdentifie唯一，如果不唯一，返回violation |
| **Upstream ID** | [SWS\_CM\_10410] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用Service skeleton的构造函数 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00006] InstanceSpecifier 构造Service skeleton(InstanceSpecifier check during the creation of service skeleton)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00006 |
| **Type** | Valid |
| **Description** | 提供InstanceSpecifier构造Service skeleton时，要检查InstanceSpecifier唯一，如果不唯一，返回violation |
| **Upstream ID** | [SWS\_CM\_10450] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用Service skeleton的构造函数 |
| **Risk** | 需要实现InstanceSpecifier与InstanceIdentifier映射查找 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00007] InstanceIdentifierContainer 构造Service skeleton(InstanceIdentifierContainer check during the creation of service skeleton)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00007 |
| **Type** | Invalid[优先级低] |
| **Description** | 提供InstanceIdentifierContainer构造Service skeleton时，要检查InstanceIdentifierContainer   1. size大于0. 2. identifiers唯一 3. 所有的identifiers映射到相同的InstanceSpecifier 4. identifier之前没有创建Service skeleton Instance 5. 如果以上检查失败，返回fatal error |
| **Upstream ID** | [SWS\_CM\_10451] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用Service skeleton的构造函数 |
| **Risk** |  |
| **Change Type** | 变更 |

## service methods处理 (Processing of service methods)

### [SWRD\_CM\_CommunicationAPI\_00008] Service methods 的处理方式(Service method processing modes)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00008 |
| **Type** | Valid |
| **Description** | Service method的处理方式需要支持：   1. Polling：用户需要显示的调用一个接口，处理method调用 2. Event-driven，concurrent：默认模式，使用多线程，支持并发method调用 3. Event-driven，sequential：顺序调用，避免使用同步机制 |
| **Upstream ID** | [SWS\_CM\_10411] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用Service skeleton的构造函数 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## field注册get/set handler (Registering get/set handlers for fields)

### [SWRD\_CM\_CommunicationAPI\_00009] 调用GetHandlers (Invoking GetHandlers)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00009 |
| **Type** | Valid |
| **Description** | Proxy调用field.Get()，Skeleton注册的GetHandlers被调用 |
| **Upstream ID** | [SWS\_CM\_10412] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测注册的回调函数是否被调用 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00010] 调用SetHandlers (Invoking SetHandlers)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00010 |
| **Type** | Valid |
| **Description** | Proxy调用field.Set()，Skeleton注册的SetHandlers被调用。  在SetHandlers中，   1. 需要通过future返回field的新值。 2. 用户能够访问当前field值。 |
| **Upstream ID** | [SWS\_CM\_10413] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测注册的回调函数是否被调用 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00011] 调用SetHandlers之后Notify Field的值 (Notify the Field value after a call to the Set Handler function)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00011 |
| **Type** | Valid |
| **Description** | SetHandler通过返回值，将最新的field值，返回给调用者。并且要将最新的field值通知给其他订阅者 |
| **Upstream ID** | [SWS\_CM\_10415] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测其他订阅者是否得到通知 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00012] 保证filed有初始值(Ensuring the existence of valid Field values)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00012 |
| **Type** | Valid |
| **Description** | 如果filed：   1. hasNotifier = true 或者 2. hasGetter = true且GetHandler没注册   在调用offer\_service()之前，没有 调用field.Update()，需要返回  error code ：ComErrc::kFieldValueIsNotValid，并记录log |
| **Upstream ID** | [SWS\_CM\_00128] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测错误码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00013] 保证filed有SetHandler (Ensuring the existence of SetHandler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00013 |
| **Type** | Valid |
| **Description** | 如果filed，hasSetter = true且SetHandler没注册。  在调用offer\_service()，需要返回  error code ：ComErrc:: kSetHandlerNotSet，并记录log |
| **Upstream ID** | [SWS\_CM\_00129] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测错误码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## find service (find service)

### [SWRD\_CM\_CommunicationAPI\_00014] 调用Find service handler (Find service handler invocation)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00014 |
| **Type** | Valid |
| **Description** | 调用StartFindService后，有匹配的Service变成可用后，马上调用FindServiceHandler。之后又有匹配的Servcie变更可用，需要继续调用FindServiceHandler。 |
| **Upstream ID** | [SWS\_CM\_00124] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，检测 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00015] 调用stop find service (Calling stop find service for already stopped finds)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00015 |
| **Type** | Valid |
| **Description** | 在已经stop的FindServcieHandle调用StopFindServcie，忽略本次调用 |
| **Upstream ID** | [SWS\_CM\_10382] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数，不返回错误，不执行动作 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## events接收(Receive events)

### [SWRD\_CM\_CommunicationAPI\_00016] event接收 (event receive)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00016 |
| **Type** | Valid |
| **Description** | 1. 支持FIFO语义 2. 采用轮询方式时，event的发送端，在发送event时，不应当导致上下文切换到event的接收端。 |
| **Upstream ID** | [SWS\_CM\_00709], [SWS\_CM\_00710] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00017] event接收回调 (Receive event by getting triggered)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00017 |
| **Type** | Valid |
| **Description** | 1. 串行的调用注册的EventReceiveHandler 2. 当调用EventReceiveHandler后，接下来调用GetNewSamples()，如果GetFreeSampleCount()没有返回0，则应当至少返回一个event值 |
| **Upstream ID** | [SWS\_CM\_00182], [SWS\_CM\_00711] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## 调用service method(call service method)

### [SWRD\_CM\_CommunicationAPI\_00018] method 调用 (method call)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00018 |
| **Type** | Valid |
| **Description** | Proxy调用method，使用ara::core::Future返回给调用者，不可以引入不可控的上下文切换。  如果在method调用处理过程中，检测到错误，通过ara::core::Future::GetResult()/ara::core::Future::get()返回相应的ara::core::ErrorCode  Fire and Forget method不可以返回错误。 |
| **Upstream ID** | [SWS\_CM\_10414], [SWS\_CM\_10371] , [SWS\_CM\_90436] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用method接口 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## 更新filed中的event(Update notification events for fields)

### [SWRD\_CM\_CommunicationAPI\_00019] 更新filed中的event (Update notification events for fields)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00019 |
| **Type** | Valid |
| **Description** | 如果hasNotifier is true，需要有如下需求中定义的接口：  [SWS\_CM\_00141] Method to subscribe to a service event.  [SWS\_CM\_00151] Method to unsubscribe from a service event.  [SWS\_CM\_00316] Method to query the subscription state.  [SWS\_CM\_00701] Method to receive a service event using polling.  [SWS\_CM\_00181] Method to enable service event trigger.  [SWS\_CM\_00182] Event Receive Handler call serialization.  [SWS\_CM\_00183] Method to disable service event trigger.  [SWS\_CM\_00333] Method to set a subscription state change handler.  [SWS\_CM\_00334] Method to unset a subscription state change handler. |
| **Upstream ID** | [SWS\_CM\_00120] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 不变 |

## Instance Specifier 解析(Instance Specifier Translation)

### [SWRD\_CM\_CommunicationAPI\_00020] InstanceSpecifier 转换 InstanceIdentifiers (InstanceSpecifier translation to InstanceIdentifiers)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00020 |
| **Type** | Valid |
| **Description** | 将InstanceSpecifier转换成InstanceIdentifiers，转换的结果可以是0，1，或者多个。 |
| **Upstream ID** | [SWS\_CM\_10452] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的函数：ara::core::Result<ara::com::InstanceIdentifierContainer>  ara::com::runtime::ResolveInstanceIDs(ara::core::InstanceSpecifier modelName); |
| **Risk** | 需要生成manifest，查找映射 |
| **Change Type** | 不变 |

## 无效值 (Invalid Value)

### [SWRD\_CM\_CommunicationAPI\_00021] 无效值实现 (Implementation of invalidValue)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00021 |
| **Type** | Valid |
| **Description** | 在AUTOSAR中可以在定义数据类型时，定义无效值。  在类型定义的头文件中，使用如下形式定义无效值：  constexpr static <SourceDataType> kInvalidValue<DataType> = <InvalidValue>;  Note：无效值只适用于：  CppImplementationDataType of category TYPE\_REFERENCE and STRING |
| **Upstream ID** | [SWS\_CM\_10453] |
| **Dependencies** | TPS |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 变更 |

## API头文件（API Header files）

### [SWRD\_CM\_CommunicationAPI\_00022] 文件夹结构 (Folder structure)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00022 |
| **Type** | Valid |
| **Description** | Service头文件，common头文件，Implementation Type头文件，Implementation 头文件，都在命名空间为分隔符的目录结构。 |
| **Upstream ID** | [SWS\_CM\_01020] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00023] Implementation Types头文件目录结构 (Implementation types header files directory structure)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00023 |
| **Type** | Valid |
| **Description** | 通信管理希望根据[SWS\_CM\_12001]生成Implementation Type头文件，并且应根据[SWS\_LBAP\_00034]存放到相应的目录。 |
| **Upstream ID** | [SWS\_CM\_12000] |
| **Dependencies** | [SWS\_CM\_12001] [SWS\_LBAP\_00034] |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## 服务头文件（Service header files）

### [SWRD\_CM\_CommunicationAPI\_00024] Service头文件(Service header files)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00024 |
| **Type** | Valid |
| **Description** | 每个ServiceInterface将文件名<name>\_proxy.h用于Proxy头文件，将<name>\_skeleton.h用于Skeleton头文件，其中<name>是ServiceInterface.shortName转换为小写字母。 |
| **Upstream ID** | [SWS\_CM\_01002] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00025] Common头文件 (Common header file)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00025 |
| **Type** | Valid |
| **Description** | Proxy和Skeleton头文件应包括Common头文件。 |
| **Upstream ID** | [SWS\_CM\_01004] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00026] Service头文件命名空间 (Namespace of Service header files)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00026 |
| **Type** | Valid |
| **Description** | service头文件的C ++命名空间格式描述。  namespace <ServiceInterface.namespace[0].symbol> {  namespace <ServiceInterface.namespace[1].symbol> {  namespace <...> {  namespace <ServiceInterface.namespace[n].symbol> {  ...  } // namespace <ServiceInterface.namespace[n].symbol>  } // namespace <...>  } // namespace <ServiceInterface.namespace[1].symbol>  } // namespace <ServiceInterface.namespace[0].symbol> |
| **Upstream ID** | [SWS\_CM\_01005] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00027] Service类命名空间 (namespace for a specific service)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00027 |
| **Type** | Valid |
| **Description** | service skeleton，service proxy，service events，service methods，service fields命名空间为namespace name {...}，如skeleton：  namespace skeleton {...} |
| **Upstream ID** | [SWS\_CM\_01006] [SWS\_CM\_01007] [SWS\_CM\_01009]  [SWS\_CM\_01015] [SWS\_CM\_01031] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## Common头文件（Common header file）

### [SWRD\_CM\_CommunicationAPI\_00028] Common头文件描述 (Common header file)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00028 |
| **Type** | Valid |
| **Description** | Common头文件name<name>\_common.h，其中<name>是服务接口.shortName转换为小写字母。Common头文件应包括Types头文件:  #Include “ara / com/types.h”  Common头文件应包括实现类型头文件impl\_type\_ <symbol> .h。 |
| **Upstream ID** | [SWS\_CM\_01012] [SWS\_CM\_01001] [SWS\_CM\_10372] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00029] Common头文件中类型定义和命名空间(Type definitions and namespaces in common header files)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00029 |
| **Type** | Valid |
| **Description** | Common头文件应该包含ServiceInterface的ApApplicationErrors的所有ApApplicationErrorDomain的类定义。  Commo头文件应包含根据要求识别服务类型的信息[SWS\_CM\_01010]。  根据[SWS\_CM\_01017]的声明和定义应该位于由[SWS\_CM\_01005]定义的c++命名空间中，以匹配相关框架和代理头文件的命名空间。 |
| **Upstream ID** | [SWS\_CM\_10370] [SWS\_CM\_01017] [SWS\_CM\_01008] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## Types头文件（Types header file）

### [SWRD\_CM\_CommunicationAPI\_00030] Types头文件中类型定义和命名空间(Type definitions and namespaces in types header files)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00030 |
| **Type** | Valid |
| **Description** | Types头文件是types.h。其头文件命名空间：  namespace ara {  namespace com {  ...  } // namespace com  } // namespace ara  类型头文件应包括根据以下内容的数据类型定义：  [SWS\_CM\_00301],[SWS\_CM\_00302],[SWS\_CM\_00303], [SWS\_CM\_00304],[SWS\_CM\_00383],[SWS\_CM\_00306], [SWS\_CM\_00308],[SWS\_CM\_00309],[SWS\_CM\_00310]  和[SWS\_CM\_00311]。 |
| **Upstream ID** | [SWS\_CM\_01013] [SWS\_CM\_01018] [SWS\_CM\_01019] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## Implementation Types头文件（Implementation Types header file）

### [SWRD\_CM\_CommunicationAPI\_00031] C++ Implementation Data Types files (C++ Implementation Data Types files)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00031 |
| **Type** | Valid |
| **Description** | 通信管理应该使用根据下面规则生成的C++类型头文件  [SWS\_LBAP\_00032] (header file generation),  [SWS\_LBAP\_00033] (header file naming),  [SWS\_LBAP\_00034] (directory naming), |
| **Upstream ID** | [SWS\_CM\_12001] |
| **Dependencies** | [SWS\_LBAP\_00032] [SWS\_LBAP\_00033] [SWS\_LBAP\_00034] |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## 服务标识符数据类型（Service Identifier Data Types）

### [SWRD\_CM\_CommunicationAPI\_00032] 服务ID，服务版本，服务合同(Service Identifier, Service Version Classes and Service Contract Version)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00032 |
| **Type** | Valid |
| **Description** | CM应提供一个名为ServiceInterface.shortName的C ++类。该类至少包含一个完全限定的名称标识符，一个服务版本以及一个主要的服务合同和次要的版本号。并且存在一些派生关系和一些运算符计算，如下：  class <ServiceInterface.shortName> {  public:  static const serviceIdentifierType serviceIdentifier;  static const ServiceVersionType serviceVersion;  static std::uint32\_t serviceContractVersionMajor;  static std::uint32\_t serviceContractVersionMinor;  };  class serviceIdentifierType {  bool operator==(const serviceIdentifierType& other) const;  bool operator<(const serviceIdentifierType& other) const;  serviceIdentifierType& operator=(const serviceIdentifierType& other);  ara::core::StringView ToString() const;  };  class ServiceVersionType {  bool operator==(const ServiceVersionType& other) const;  bool operator<(const ServiceVersionType& other) const;  ServiceVersionType& operator=(const ServiceVersionType& other);  ara::core::StringView ToString() const;  }; |
| **Upstream ID** | [SWS\_CM\_01010] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00033] InstanceIdentifier类(Instance Identifier Class)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00033 |
| **Type** | Valid |
| **Description** | 通信管理应提供一个类 InstanceIdentifier。 它包含实例信息  以及有关服务类型的信息。  这将使 InstanceIdentifier 对于不同的实例是唯一的。：  class InstanceIdentifier {  public:  static ara::core::Result<InstanceIdentifier> Create(StringView  serializedFormat) noexcept;  explicit InstanceIdentifier( ara::core::StringView value);  ara::core::StringView ToString() const;  bool operator==(const InstanceIdentifier& other) const;  bool operator<(const InstanceIdentifier& other) const;  InstanceIdentifier& operator=(const InstanceIdentifier& other);  };  CM应该提供一个InstanceIdentifierContainer的别名  using InstanceIdentifierContainer = ara::core::Vector<InstanceIdentifier>;  如果提供给 Create() 的字符串表示的格式已损坏，或不符合软件提供者规范，则应在 Result 类型中返回错误代码 ComErrc::kInvalidInstanceIdentifierString。  如果提供的字符串格式损坏或不符合软件提供者规范，类构造函数应抛出 ComException |
| **Upstream ID** | [SWS\_CM\_00302] [SWS\_CM\_00319] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00034] FindServiceHandle类(FindServiceHandle)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00034 |
| **Type** | Valid |
| **Description** | CM提供一个FindServiceHandle，StartFindService方法能返回一个FindServiceHandle，该参数也可用作通过StopFindService取消此请求的参数，并且包含如下内容：  struct FindServiceHandle {  internal::ServiceId serviceIdentifier;  internal::InstanceId instanceIdentifier;  std::uint32\_t uid;  bool operator==(const FindServiceHandle& other) const;  bool operator<(const FindServiceHandle& other) const;  FindServiceHandle& operator=(const FindServiceHandle& other);  ...  }; |
| **Upstream ID** | [SWS\_CM\_00303] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00035] Handle Type类(Handle Type Class)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00035 |
| **Type** | Valid |
| **Description** | CM提供一个HandleType类，应包含创建ServiceProxy所需的信息，并且CM可以扩展HandleType的定义，必须包含如下内容：  class HandleType {  public:  bool operator==(const HandleType& other) const;  bool operator<(const HandleType& other) const;  const ara::com::InstanceIdentifier& GetInstanceId() const;  }; |
| **Upstream ID** | [SWS\_CM\_00312] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00036] Handle Type类的构造和赋值函数(Constructors and assignment functions for the handle type class)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00036 |
| **Type** | Valid |
| **Description** | Handle Type Class的拷贝构造，移动构造，拷贝赋值，移动赋值，析构  HandleType(const HandleType&);  HandleType& operator=(const HandleType&);  HandleType(HandleType &&);  HandleType& operator=(HandleType &&);  ~HandleType() noexcept; |
| **Upstream ID** | [SWS\_CM\_00317] [SWS\_CM\_00318] [SWS\_CM\_11371] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00037] Service处理容器(Service Handle Container)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00037 |
| **Type** | Valid |
| **Description** | CM应提供ServiceHandleContainer的定义，该容器包含服务句柄列表，并用作FindService方法的返回值。  template <typename T>  using ServiceHandleContainer = ara::core::Vector<T>; |
| **Upstream ID** | [SWS\_CM\_00304] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00038] Find服务处理程序(Find Service Handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00038 |
| **Type** | Valid |
| **Description** | FindServiceHandler的定义，它以handle容器作为输入参数 包含所有匹配服务实例的句柄以及一个FindServiceHandle。  template <typename T>  using FindServiceHandler =  std::function<void(ServiceHandleContainer<T>, FindServiceHandle)>; |
| **Upstream ID** | [SWS\_CM\_00383] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## 与事件相关数据类型（Event Related Data Types）

### [SWRD\_CM\_CommunicationAPI\_00039] Sample指针(Sample Pointer)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00039 |
| **Type** | Valid |
| **Description** | SamplePtr模板包含如下：  template< typename T >  class SamplePtr {  // Default constructor  constexpr SamplePtr() noexcept;  // semantically equivalent to Default constructor  constexpr SamplePtr(std::nullptr\_t) noexcept;  // Copy constructor is deleted  SamplePtr ( const SamplePtr& ) = delete;  // Move constructor  SamplePtr( SamplePtr&& ) noexcept;  // Destructor  ~SamplePtr() noexcept;  // Default copy assignment operator is deleted  SamplePtr& operator=( const SamplePtr& ) = delete;  // Assignment of nullptr\_t  SamplePtr& operator=(std::nullptr\_t) noexcept;  // Move assignment operator  SamplePtr& operator=( SamplePtr&& ) noexcept;  // Dereferences the stored pointer  T& operator\*() const noexcept;  T\* operator->() const noexcept;  //Checks if the stored pointer is null  explicit operator bool () const noexept;  // Swaps the managed object  void Swap ( SamplePtr& ) noexcept;  //Replaces the managed object  void Reset (std::nullptr\_t) noexcept;  //Returns the stored object  T\* Get () const noexcept;  // Returns the end 2 end protection check result  ara::com::e2e::ProfileCheckStatus GetProfileCheckStatus() const noexcept;  }; |
| **Upstream ID** | [SWS\_CM\_00306] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相关函数 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00040] 数据类型(Data Types)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00040 |
| **Type** | Valid |
| **Description** | CM应提供下面的数据类型：  ara::com::e2e::ProfileCheckStatus GetProfileCheckStatus() const noexcept;  template <typename T>  using SampleAllocateePtr = std::unique\_ptr<T>;  using EventReceiveHandler = std::function<void()>;  enum class SubscriptionState : std::uint8\_t {  kSubscribed,  kNotSubscribed,  kSubscriptionPending  };  using SubscriptionStateChangeHandler =  std::function<void(SubscriptionState)>; |
| **Upstream ID** | [SWS\_CM\_90420] [SWS\_CM\_00308] [SWS\_CM\_00309]  [SWS\_CM\_00310] [SWS\_CM\_00311] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 与Trigger相关数据类型（Trigger Related Data Types）

### [SWRD\_CM\_CommunicationAPI\_00041] Trigger接收句柄(Trigger Receive Handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00042 |
| **Type** | Valid |
| **Description** | Trigger接收句柄的定义与【SWS\_CM\_00309】定义相同。 |
| **Upstream ID** | [SWS\_CM\_00351] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## 与方法相关数据类型（Method Related Data Types）

### [SWRD\_CM\_CommunicationAPI\_00042] 方法调用处理方式(Method Call Processing Mode)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00042 |
| **Type** | Valid |
| **Description** | CM应提供一个MethodCallProcessingMode枚举，定义了service implementation的处理模式：  enum class MethodCallProcessingMode : std::uint8\_t {  kPoll,  kEvent,  kEventSingleThread  }; |
| **Upstream ID** | [SWS\_CM\_00301] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## Variant数据类型

### [SWRD\_CM\_CommunicationAPI\_00043]Variant类模板（Variant Class Template）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00043 |
| **Type** | Valid |
| **Description** | 通讯管理至少应提供一个类型保存的联合表示的 Variant 类模板。 |
| **Upstream ID** | [SWS\_CM\_01050] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00044] Variant 默认构造函数（Variant default constructor）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00044 |
| **Type** | Valid |
| **Description** | Variant的构造函数与std::variant的构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01051] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00045] Variant 移动构造函数（Variant move constructor）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00045 |
| **Type** | Valid |
| **Description** | Variant的移动构造函数与std::variant的移动构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01052] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00046] Variant 拷贝构造函数（Variant copy constructor）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00046 |
| **Type** | Valid |
| **Description** | Variant的移动构造函数与std::variant的移动构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01053] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00047] Variant 转换构造函数（Variant converting constructor）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00047 |
| **Type** | Valid |
| **Description** | Variant的转换构造函数与std::variant的转换构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01054] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00048] Variant 指定类型的显式转换构造函数（Variant explicit converting constructor with specified alternative）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00048 |
| **Type** | Valid |
| **Description** | Variant的指定类型的显式转换构造函数与std::variant的指定类型的显式转换构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01055] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00049] Variant 指定类型和初始化列表的显式转换构造函数（Variant explicit converting constructor with specified alternative and initializer list）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00049 |
| **Type** | Valid |
| **Description** | Variant的指定类型和初始化列表的显式转换构造函数与std::variant的指定类型和初始化列表的显式转换构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01056] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00050] Variant 通过Index指定类型的显式转换构造函数（Variant explicit converting constructor with specified alternative specified by index）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00050 |
| **Type** | Valid |
| **Description** | Variant通过Index指定类型的显式转换构造函数与std::variant通过Index指定类型的显式转换构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01057] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00051] Variant 通过Index指定类型和初始化列表的显式转换构造函数（Variant explicit converting constructor with specified alternative specified by index and initializer list）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00051 |
| **Type** | Valid |
| **Description** | Variant通过Index指定类型和初始化列表的显式转换构造函数与std::variant通过Index指定类型和初始化列表的显式转换构造函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01058] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00052] Variant析构函数（Variant destructor）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00052 |
| **Type** | Valid |
| **Description** | Variant的析构函数与std::variant带有noexcept关键字的析构函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01059] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00053] Variant移动赋值运算符（Variant move assignment operator）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00053 |
| **Type** | Valid |
| **Description** | Variant的移动赋值运算符与std::variant的移动赋值运算符行为一致。 |
| **Upstream ID** | [SWS\_CM\_01060] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00054] Variant拷贝赋值运算符（Variant copy assignment operator）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00054 |
| **Type** | Valid |
| **Description** | Variant的拷贝赋值运算符与std::variant的拷贝赋值运算符行为一致。 |
| **Upstream ID** | [SWS\_CM\_01061] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00055] Variant转换赋值运算符（Variant converting assignment operator）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00055 |
| **Type** | Valid |
| **Description** | Variant的转换赋值运算符与std::variant的转换赋值运算符行为一致。 |
| **Upstream ID** | [SWS\_CM\_01062] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00056] Variant返回可选项的0基下标函数（Variant function to return the zero-based index of the alternative）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00056 |
| **Type** | Valid |
| **Description** | Variant的返回可选项的0基下标函数与std::variant的返回可选项的0基下标函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01063] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00057] Variant检查是否非法状态函数（Variant function to check if the Variant is in invalid state）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00057 |
| **Type** | Valid |
| **Description** | Variant的检查是否非法状态函数与std::variant的检查是否非法状态函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01064] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00058] Variant原位构造新值的函数（Variant function to create a new value in-place, in an existing Variant object）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00058 |
| **Type** | Valid |
| **Description** | Variant的原位构造新值的函数与std::variant的原位构造新值的函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01066] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00059] Variant使用初始化列表原位构造新值的函数（Variant function to create a new value in-place, in an existing Variant object using an initializer list）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00059 |
| **Type** | Valid |
| **Description** | Variant的使用初始化列表原位构造新值的函数与std::variant的使用初始化列表原位构造新值的函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01067] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00060] Variant销毁所含值并原位构造新值并初始化的函数（Variant function to create a new value in-place, in an existing Variant object by destoying and initializing the contained value）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00060 |
| **Type** | Valid |
| **Description** | Variant的销毁所含值并原位构造新值并初始化的函数与std::variant的销毁所含值并原位构造新值并初始化的函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01068] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00061] Variant销毁所含值并使用初始化列表原位构造新值的函数（Variant function to create a new value in-place, in an existing Variant object by destoying and initializing the contained value using an initializer list）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00061 |
| **Type** | Valid |
| **Description** | Variant的销毁所含值并使用初始化列表原位构造新值的函数与std::variant的销毁所含值并使用初始化列表原位构造新值的函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01069] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00062] 交换两个variant对象的函数（Variant function to swap two Variants）

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00062 |
| **Type** | Valid |
| **Description** | 交换二个 variant 对象的函数与std::variant的交换二个 variant 对象函数行为一致。 |
| **Upstream ID** | [SWS\_CM\_01065] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## 错误类型（Error Types）

### [SWRD\_CM\_CommunicationAPI\_00063] CM ErrorDomain (CM ErrorDomain)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00063 |
| **Type** | Valid |
| **Description** | service接口的任何检查错误都应通过CoreTypes中指定的返回类型进行报告。  Errors定义ara::com::comErrorDomain里，其ErrorDomain id = 0x8000'0000'0000'1267。 |
| **Upstream ID** | [SWS\_CM\_11265] [SWS\_CM\_11264] [SWS\_CM\_11267] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## E2E关联数据类型（E2E Related Data Types）

### [SWRD\_CM\_CommunicationAPI\_00064 方法调用处理方式(Method Call Processing Mode)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00064 |
| **Type** | Valid |
| **Description** | ara::com::e2e::Profile枚举，表示single sample检查结果。  enum class ProfileCheckStatus : uint8\_t  {  kOk,  kRepeated,  kWrongSequence,  kError,  kNotAvailable,  kNoNewData,  kCheckDisabled  };  应提供一个枚举ara :: com：e2e :: SMState，该枚举表示在最近一次检查接收到的事件样本后，E2E监督处于什么状态。如下  enum class SMState : uint8\_t  {  kValid,  kNoData,  kInit,  kInvalid,  kStateMDisabled  }; |
| **Upstream ID** | [SWS\_CM\_90421] [SWS\_CM\_90422] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## API概述(API General)

### [SWRD\_CM\_CommunicationAPI\_00065] Skeleton (Skeleton)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00065 |
| **Type** | Valid |
| **Description** | Skeleton定义规则：   1. skeleton class   class UpperCamelCase(<ServiceInterface.shortName>)Skeleton {};   1. event class   class UpperCamelCase(<VariableDataPrototype.shortName>) {};   1. field class   class UpperCamelCase(<Field.shortName>) {};  其中shortName转换为首字母大写的驼峰命名规则 |
| **Upstream ID** | [SWS\_CM\_00002], [SWS\_CM\_00003], [SWS\_CM\_00007] |
| **Dependencies** | TPS |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码是否符合规则 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00066] Proxy (Proxy)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00066 |
| **Type** | Valid |
| **Description** | Proxy定义规则：   1. proxy class   class UpperCamelCase(<ServiceInterface.shortName>)Proxy {};   1. event class   class UpperCamelCase(<VariableDataPrototype.shortName>) {};   1. method class   class UpperCamelCase(<ClientServerOperation.shortName> {};   1. field class   class UpperCamelCase(<Field.shortName>) {};  其中shortName转换为首字母大写的驼峰命名规则 |
| **Upstream ID** | [SWS\_CM\_00004], [SWS\_CM\_00005], [SWS\_CM\_00006] , [SWS\_CM\_00008] |
| **Dependencies** | TPS |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码是否符合规则 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00067] API类型-通信和服务发现API (Types of APIs - Communication and Service Discovery APIs)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00067 |
| **Type** | Valid |
| **Description** | 有两类API：通信API和服务发现API  服务发现API（这些API用于服务发现阶段）：  \_ OfferService()  \_ StopOfferService()  \_ RegisterGetHandler()  \_ RegisterSetHandler()  \_ FindService()  \_ StartFindService()  \_ StopFindService()  \_ GetHandle()  \_ Subscribe()  \_ Unsubscribe()  \_ GetSubscriptionState()  \_ SetSubscriptionStateChangeHandler()  \_ UnsetSubscriptionStateChangeHandler()  \_ SetReceiveHandler()  \_ UnsetReceiveHandler()  通信API（这些API用于客户端和服务端通信）：  \_ Send()  \_ Allocate()  \_ Update()  \_ GetNewSamples()  \_ GetFreeSampleCount()  \_ Method call operator()  \_ Get()  \_ Set() |
| **Upstream ID** | [SWS\_CM\_99028] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00068] 可重入 (Re-entrancy - General)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00068 |
| **Type** | Valid |
| **Description** | skeleton，proxy类的不同成员函数要支持可重入。  相同/不同类对象的不同成员函数要支持并发调用。 |
| **Upstream ID** | [SWS\_CM\_00009] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

## Named构造(Object Creation via Named Constructor Approach)

### [SWRD\_CM\_CommunicationAPI\_00069] Named构造 (Named Constructor Approach )

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00069 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. 支持对象的创建过程exception-less（不抛出异常） 2. 方法原型：   static ara::core::Result<ClassToBeCreated> Create(/\* construction arguments \*/) noexcept;   1. 当构造过程失败的时候，通过ara::core::Result返回ara::core::ErrorCode |
| **Upstream ID** | [SWS\_CM\_11326] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用create方法 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## offer service方法 (offer service method)

### [SWRD\_CM\_CommunicationAPI\_00070]offer service方法 (offer service method)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00070 |
| **Type** | Valid |
| **Description** | 1. 提供ara::core::Result<void> OfferService();作为skeleton类的成员函数，实现offer service 2. 如果skeleton类中有field，且没有赋初值，调用OfferService，返回ComErrc::kFieldValueIsNotValid 3. 如果skeleton类中的field hasSetter=true，且没有支持SetHandler回调，调用OfferService，返回ComErrc::kSetHandlerNotSet 4. 当OfferService返回错误时，不执行service offer。 5. 支持可重入 |
| **Upstream ID** | [SWS\_CM\_00101], [SWS\_CM\_00010] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用offer service接口 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## stop offer service方法 (stop offer service method)

### [SWRD\_CM\_CommunicationAPI\_00071]stop offer service方法 (stop offer service method)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00071 |
| **Type** | Valid |
| **Description** | 1. 提供void StopOfferService();作为skeleton类的成员函数,实现stop offer service 2. 支持可重入 |
| **Upstream ID** | [SWS\_CM\_00111], [SWS\_CM\_00011] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用stop offer service接口 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 创建Service skeleton(Service skeleton creation)

### [SWRD\_CM\_CommunicationAPI\_00072]使用InstancID创建service skeleton (Creation of service skeleton using Instance ID)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00072 |
| **Type** | Valid |
| **Description** | 1. service skeleton构造方法原型：   ServiceSkeleton( ara::com::InstanceIdentifier instanceID,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent);   1. 参数mode有默认参数，kEvent |
| **Upstream ID** | [SWS\_CM\_00130] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00073]使用InstancID创建service skeleton (Creation of service skeleton using Instance ID)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00073 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. Exception-less构造方法原型：   static ara::core::Result<ServiceSkeleton> Create(  const ara::com::InstanceIdentifier &instanceID,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent) noexcept ;   1. 参数mode有默认参数，kEvent 2. 如果service使能了E2E，当mode=kEvent时，返回kWrongMethodCallProcessingMode错误码 3. 如果Grant enforcement failure，返回ComErrc::kGrantEnforcementError错误码 |
| **Upstream ID** | [SWS\_CM\_10435] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00074使用InstancSpecifier建service skeleton (Creation of service skeleton using Instance Spec)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00074 |
| **Type** | Valid |
| **Description** | 1. service skeleton构造方法原型：   ServiceSkeleton(ara::core::InstanceSpecifier instanceSpec,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent);   1. 参数mode有默认参数，kEvent |
| **Upstream ID** | [SWS\_CM\_00152] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | M |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | InstanceSpecifier映射实现 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00075]使用InstancID Container创建service skeleton (Creation of service skeleton using Instance ID Container)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00075 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. service skeleton构造方法原型：   ServiceSkeleton(ara::com::InstanceIdentifierContainer instanceIDs,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent);   1. 参数mode有默认参数，kEvent |
| **Upstream ID** | [SWS\_CM\_00153] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 需要支持多重绑定 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00076]使用InstancSpecifier创建service skeleton (Creation of service skeleton using Instance Spec)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00076 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. Exception-less构造方法原型：   static ara::core::Result<ServiceSkeleton> Create(  const ara::core::InstanceSpecifier &instanceSpec,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent) noexcept;   1. 参数mode有默认参数，kEvent 2. 如果service使能了E2E，当mode=kEvent时，返回kWrongMethodCallProcessingMode错误码 3. 如果Grant enforcement failure，返回ComErrc::kGrantEnforcementError错误码 |
| **Upstream ID** | [SWS\_CM\_10436] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00077]使用InstancID Container创建service skeleton (Creation of service skeleton using Instance ID Container)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00077 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. Exception-less构造方法原型：   static ara::core::Result<ServiceSkeleton> Create(  const ara::com::InstanceIdentifierContainer &instanceIDs,  ara::com::MethodCallProcessingMode mode =  ara::com::MethodCallProcessingMode::kEvent) noexcept;   1. 参数mode有默认参数，kEvent 2. 如果service使能了E2E，当mode=kEvent时，返回kWrongMethodCallProcessingMode错误码 3. 如果Grant enforcement failure，返回ComErrc::kGrantEnforcementError错误码 |
| **Upstream ID** | [SWS\_CM\_10437] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00078]service skeleton的拷贝控制 (service skeleton copy and move)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00078 |
| **Type** | Valid |
| **Description** | 1. 禁止拷贝构造和拷贝赋值：   ServiceSkeleton(const ServiceSkeleton&) = delete;  ServiceSkeleton& operator=(const ServiceSkeleton&) = delete;   1. 提供移动构造和移动赋值：   ServiceSkeleton(ServiceSkeleton &&);  ServiceSkeleton& operator=(ServiceSkeleton &&);   1. 提供析构函数：   ~ServiceSkeleton(); |
| **Upstream ID** | [SWS\_CM\_00134], [SWS\_CM\_00135]，[SWS\_CM\_11370] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用service skeleton创建方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 发送event(Send event)

### [SWRD\_CM\_CommunicationAPI\_00079]Event数据由application提供(Send event where application is responsible for the data)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00079 |
| **Type** | Valid |
| **Description** | 1. event Sent函数原型：   ara::core::Result<void> Event::Send(const SampleType &data);   1. CM使用data的拷贝 2. 数据发送失败，返回如下错误码：   ComErrc::kServiceNotOffered: Service not offered.  ComErrc::kCommunicationLinkError: Communication link is broken.  ComErrc::kCommunicationStackError: Communication Stack Error, e.g.  network stack, network binding, or communication framework reports an error.   1. 不同Event对象，Send方法支持可重入。相同Event对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00162], [SWS\_CM\_00012] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用event send方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00080]Event数据由CM提供(Send event where CM is responsible for the data)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00080 |
| **Type** | Valid |
| Description | 1. event Sent函数原型：   ara::core::Result<void> Event::Send(ara::com::SampleAllocateePtr  <SampleType> data);   1. CM管理data的生命周期，调用Send后，application不允许访问data 2. 在调用Send之前，application需要向CM申请data，使用Allocate(): 3. 数据发送失败，返回如下错误码：   ComErrc::kServiceNotOffered: Service not offered.  ComErrc::kCommunicationLinkError: Communication link is broken.  ComErrc::kCommunicationStackError: Communication Stack Error, e.g.  network stack, network binding, or communication framework reports an error.   1. 不同Event对象，Send方法支持可重入。相同Event对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_90437] , [SWS\_CM\_00012] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用event send方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00081]Event数据的申请(Allocating data for event transfer)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00081 |
| **Type** | Valid |
| Description | 1. event Allocate函数原型：   ara::core::Result<ara::com::SampleAllocateePtr<SampleType>>  Event::Allocate();   1. CM负责释放Allocate后的内存 2. 失败时返回的错误码：   ComErrc::kSampleAllocationFailure: If the allocation of the shared  memory fails (i.e., failure to retrieve/allocate a shared slot for a sample).  ComErrc::kIllegalUseOfAllocate: If the allocation is done via custom allocator(i.e., not via shared memory allocation).The error shall be logged.   1. 不同Event对象，Allocate方法支持可重入。相同Event对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_90438] , [SWS\_CM\_00013] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用event Allocate方法 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 发送Trigger (Send Trigger)

### [SWRD\_CM\_CommunicationAPI\_00082]发送Trigger (Send Trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00082 |
| **Type** | Valid |
| **Description** | 1. Trigger类的Send方法会发送trigger给所有订阅的应用程序。   如果未成功,Send方法会返回错误码：  \_ ComErrc::kServiceNotOffered: Service not offered.  \_ ComErrc::kCommunicationLinkError: Communication link is broken.  \_ ComErrc::kCommunicationStackError: Communication Stack Error, e.g.  network stack, network binding, or communication framework reports an error.   1. 对于不同的 Trigger 类实例，Send 应该是可重入的和线程安全的。 当在同一个 Trigger 类实例上可重入或同时调用时，行为是未定义的。 |
| **Upstream ID** | [SWS\_CM\_00721] [SWS\_CM\_00722] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## service method (Provide a service method)

### [SWRD\_CM\_CommunicationAPI\_00083]method规定(Provision of method)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00083 |
| **Type** | Valid |
| **Description** | 在ServiceSkeleton的类中：   1. 定义提供的method的纯虚函数 2. 定义input参数 3. 定义output参数，包括出参和返回值 4. 使用ara::core::Future返回output参数 5. 子类提供method的实现   例如：  struct Method1Output {  TypeOutputParameter1 output1;  TypeOutputParameter2 output2;  ...  TypeResult result;  };  virtual ara::core::Future <Method1Output> Method1(  TypeInputParameter1 input1,  TypeInputParameter2 input2,  ...  ) = 0; |
| **Upstream ID** | [SWS\_CM\_00191] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00084]Fire and Forget method规定(Provision of a Fire and Forget method)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00084 |
| **Type** | Valid |
| **Description** | 在ServiceSkeleton的类中：   1. 定义提供的Fire and Forget method的纯虚函数 2. 定义input参数 3. 没有返回值 4. 子类提供Fire and Forget method的实现   例如：  virtual void FireForgetMethod1(  TypeInputParameter1 input1,  TypeInputParameter2 input2,  ...) = 0; |
| **Upstream ID** | [SWS\_CM\_90434] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00085]method实现的可重入(Re-entrancy - ServiceSkeleton method implementation)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00085 |
| **Type** | Valid |
| **Description** | 1. 如果method可重入，可以使用kEvent 2. 如果mehtod不可重入，可以使用kPoll或kEventSigleThread |
| **Upstream ID** | [SWS\_CM\_00017] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | - |
| **Risk** | 无 |
| **Change Type** | 新增 |

## service method处理 (Processing of service methods)

### [SWRD\_CM\_CommunicationAPI\_00086]method处理模式(Set service method processing mode)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00086 |
| **Type** | Valid |
| **Description** | 1. 在ServiceSkeleton的构造函数的MethodProcessingMode指定 2. 处理模式表示ServiceSkeleton在处理method时的方式 3. 支持的模式有：   Polling (enumeration element kPoll)  Event-driven, concurrent (enumeration element kEvent)  Event-driven, sequential (enumeration element kEventSingleThread) |
| **Upstream ID** | [SWS\_CM\_00198] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00087]method调用处理(Process Service method invocation)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00087 |
| **Type** | Valid |
| **Description** | 当处理模式是Polling：   1. 对应用提供一个接口，来处理Proxy发起的method调用，原型为：   ara::core::Future<bool> ProcessNextMethodCall();   1. 通过ara::core::Future<bool>返回处理结果，如果是true，表示还有pending的method调用。如果是false，表示当前没有method调用 2. ara::core::Future(then)支持注册一个回调函数，当pending method调用处理完成后，调用这个注册的回调函数 3. 提供了一个重载函数，原型为：   template<typename ExecutorT>  ara::core::Future<bool> ProcessNextMethodCall(ExecutorT &&executor);   1. executor提供了一个executioner object，在executioner object中调用ProcessNextMethodCall 2. 如果在executor中不能支持ProcessNextMethodCall调用，返回ComErrc::kCouldNotExecute错误码 3. 应用可以通过ara::core::Promise设置ApApplicationError，通过ara::core::Future返回给调用方 |
| **Upstream ID** | [SWS\_CM\_00199],[SWS\_CM\_11350], [SWS\_CM\_11351], [SWS\_CM\_11362] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 注册field的get handlers (Registering get handlers for fields)

### [SWRD\_CM\_CommunicationAPI\_00088]注册Getters(Registering Getters)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00088 |
| **Type** | Valid |
| **Description** | 1. 提供Field的getter的注册接口，原型：   ara::core::Result<void> RegisterGetHandler(  std::function<ara::core::Future<FieldType>()> getHandler); |
| **Upstream ID** | [SWS\_CM\_00114] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00089]提供Execution Context注册Getters(Execution Context for registering Getters)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00089 |
| **Type** | Valid |
| **Description** | 1. 提供Field的getter的注册接口，原型：   template<typename ExecutorT>  ara::core::Result<void> RegisterGetHandler(  std::function<ara::core::Future<FieldType>()> getHandler, ExecutorT&& executor);   1. 如果在executor中不能支持RegisterGetHandler调用，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11360], [SWS\_CM\_11361] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00090] Getters通用(Getters General)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00090 |
| **Type** | Valid |
| **Description** | 1. Field.hasGetter控制是否需要RegisterGetHandler 2. 不同Field对象，RegisterGetHandler方法支持可重入。相同Field对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00115], [SWS\_CM\_00014] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## 注册field的set handlers (Registering set handlers for fields)

### [SWRD\_CM\_CommunicationAPI\_00091]注册Setters(Registering Setters)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00091 |
| **Type** | Valid |
| **Description** | 提供Field的setter的注册接口，原型：  ara::core::Result<void> RegisterSetHandler(  std::function<ara::core::Future<FieldType>(  const FieldType& value)> setHandler); |
| **Upstream ID** | [SWS\_CM\_00116] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00092]提供Execution Context注册Setters(Execution Context for registering Setters)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00092 |
| **Type** | Valid |
| **Description** | 1. 提供Field的setter的注册接口，原型：   template<typename ExecutorT>  ara::core::Result<void> RegisterSetHandler(  std::function<ara::core::Future<FieldType>(  const FieldType& value)> setHandler, ExecutorT&& executor);   1. 如果在executor中不能支持RegisterSetHandler调用，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11362], [SWS\_CM\_11363] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00093] Setters通用(Setters General)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00093 |
| **Type** | Valid |
| **Description** | 1. Field.hasSetter控制是否需要RegisterSetHandler 2. 不同Field对象，RegisterSetHandler方法支持可重入。相同Field对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00117], [SWS\_CM\_00015] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00094] Update功能(Update Function)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00094 |
| **Type** | Valid |
| **Description** | 1. 提供Field的update的接口，原型：   ara::core::Result<void> Field::Update(const FieldType &value);   1. 如果update失败，返回错误码：   kServiceNotOffered: Service not offered.  kCommunicationLinkError: Communication link is broken.  kNoClients: No clients connected.  kCommunicationStackError: Communication Stack Error, e.g. network stack, network binding, or communication framework reports an error   1. 如果Field.hasGetter=true或者没有注册Getter，update接口应该更新field的内部值 2. 如果Field.hasNotification=true，update notification应该被发送 3. 不同Field对象，Update方法支持可重入。相同Field对象，并发调用，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00119], [SWS\_CM\_00016] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## find service(Find service)

### [SWRD\_CM\_CommunicationAPI\_00095]find service立即返回，使用Instance ID(Find service with immediately returned request using Instance ID)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00095 |
| **Type** | Valid |
| **Description** | 1. 提供FindService方法，原型：   static ara::core::Result<ara::com::ServiceHandleContainer<  <ProxyClassName>::HandleType>>  FindService(ara::com::InstanceIdentifier instance);   1. 通过ara::core::Result返回所以符合的Instance。 2. 返回的错误码：   kNetworkBindingFailure: Local failure has been detected by the network  binding.  kGrantEnforcementError: Request was refused by Grant enforcement  layer.  kPeerIsUnreachable: Transport Layer Security handshake failed.   1. 如果InstanceIdentifier验证失败或者ServiceHandleContainer内存申请失败，按照ara::core定义的Violation处理 |
| **Upstream ID** | [SWS\_CM\_00122] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00096]find service立即返回，使用Instance Specifier(Find service with immediately returned request using Instance Specifier)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00096 |
| **Type** | Valid |
| **Description** | 1. 提供FindService方法，原型：   static ara::core::Result<ara::com::ServiceHandleContainer<  <ProxyClassName>::HandleType>>  FindService(ara::core::InstanceSpecifier instance);   1. 通过ara::core::Result返回所以符合的Instance。 2. 返回的错误码：   kNetworkBindingFailure: Local failure has been detected by the network  binding.  kGrantEnforcementError: Request was refused by Grant enforcement  layer.  kPeerIsUnreachable: Transport Layer Security handshake failed.   1. 如果InstanceSpecifier验证失败或者ServiceHandleContainer内存申请失败，按照ara::core定义的Violation处理 |
| **Upstream ID** | [SWS\_CM\_00622] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | M |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 依赖InstanceSpecifier映射实现 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00097]find service可重入性(Re-entrancy - FindService)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00097 |
| **Type** | Valid |
| **Description** | FindService不可重入，当并发调用时，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00018] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00098]find service注册通知，使用Instance ID(Find service with handler registration using Instance ID)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00098 |
| **Type** | Valid |
| **Description** | 1. 提供StartFindService方法，原型：   static ara::core::Result<ara::com::FindServiceHandle>  StartFindService(ara::com::FindServiceHandler<<ProxyClassName>::HandleType> handler, ara::com::InstanceIdentifier instance);   1. 通过ara::core::Result返回FindServiceHandle。FindServiceHandle同时在stop find service中使用 2. 当检测到匹配的service和instance ID，通过调用FindServiceHandler通知用户 3. 返回的错误码：   kNetworkBindingFailure: Local failure has been detected by the network  binding.  kGrantEnforcementError: Request was refused by Grant enforcement  layer.  kPeerIsUnreachable: Transport Layer Security handshake failed.   1. 如果InstanceIdentifier验证失败或者ServiceHandleContainer内存申请失败，按照ara::core定义的Violation处理 |
| **Upstream ID** | [SWS\_CM\_00123] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00099]提供Execution Context find service注册通知，使用Instance ID(Execution Context for finding service with handler registration using Instance ID)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00099 |
| **Type** | Valid |
| **Description** | 1. 提供StartFindService方法，原型：   template<typename ExecutorT>  static ara::com::FindServiceHandle StartFindService(  ara::com::FindServiceHandler<<ProxyClassName>::HandleType> handler, ara::com::InstanceIdentifier instance, ExecutorT&& executor);   1. 提供了一个executioner object参数，作为FindServiceHandler的执行上下文 2. 当executioner object不能执行时，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11352], [SWS\_CM\_11353] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00100]find service注册通知，使用Instance Specifier(Find service with handler registration using Instance Specifier)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00100 |
| **Type** | Valid |
| **Description** | 1. 提供StartFindService方法，原型：   static ara::core::Result<ara::com::FindServiceHandle> StartFindService(  ara::com::FindServiceHandler<<ProxyClassName>::HandleType> handler, ara::core::InstanceSpecifier instance);   1. 通过ara::core::Result返回FindServiceHandle。FindServiceHandle同时在stop find service中使用 2. 当检测到匹配的service和instance Specifier指定的Abstact Network Binding instance，通过调用FindServiceHandler通知用户 3. 返回的错误码：   kNetworkBindingFailure: Local failure has been detected by the network  binding.  kGrantEnforcementError: Request was refused by Grant enforcement  layer.  kPeerIsUnreachable: Transport Layer Security handshake failed.   1. 如果InstanceIdentifier验证失败或者ServiceHandleContainer内存申请失败，按照ara::core定义的Violation处理 |
| **Upstream ID** | [SWS\_CM\_00623] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 依赖InstanceSpecifier映射实现 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00101]start find service可重入性(Re-entrancy - StartFindService)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00101 |
| **Type** | Valid |
| **Description** | StartFindService不可重入，当并发调用时，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00019] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00102]stop find service (Stop find service)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00102 |
| **Type** | Valid |
| **Description** | 1. 提供StopFindService方法，原型：   void StopFindService(ara::com::FindServiceHandle handle)   1. 不同ara::com::FindServiceHandles的StopFindService支持可重入，相同ara::com::FindServiceHandles并发调用StopFindService，其行为未定义 |
| **Upstream ID** | [SWS\_CM\_00125], [SWS\_CM\_00020] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## Service proxy构造(Service proxy creation)

### [SWRD\_CM\_CommunicationAPI\_00103]service proxy构造函数(Creation of service proxy)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00103 |
| **Type** | Valid |
| **Description** | 1. 提供ServicePorxy的构造函数，原型：   explicit ServiceProxy::ServiceProxy(const HandleType &handle);   1. handle参数由FindService方法提供 |
| **Upstream ID** | [SWS\_CM\_00131] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00104]不抛异常构造service proxy(Exception-less creation of service proxy)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00104 |
| **Type** | Invalid[优先级低] |
| **Description** | 1. 提供ServicePorxy不抛异常的构造方法，原型：   static ara::core::Result<ServiceProxy> Create(  const HandleType &handle) noexcept;   1. handle参数由FindService方法提供 2. 失败返回错误码：   kErroneousFileHandle  ComErrc::kGrantEnforcementError |
| **Upstream ID** | [SWS\_CM\_10438] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | L |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00105] GetHandle (GetHandle function to return the proxy instance creation handle)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00105 |
| **Type** | Valid |
| **Description** | 1. 提供GetHandle方法，原型：   HandleType ServiceProxy::GetHandle() const;   1. 实现为可重入方法 |
| **Upstream ID** | [SWS\_CM\_10383],[SWS\_CM\_00021] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00106]service proxy的拷贝控制 (service proxy copy and move)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00106 |
| **Type** | Valid |
| **Description** | 1. 禁止拷贝构造和拷贝赋值：   ServiceProxy(const ServiceProxy&) = delete;  ServiceProxy& operator=(const ServiceProxy&) = delete;   1. 提供移动构造和移动赋值：   ServiceProxy(ServiceProxy &&);  ServiceProxy& operator=(ServiceProxy &&); |
| **Upstream ID** | [SWS\_CM\_00136], [SWS\_CM\_00137] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用service proxy创建方法 |
| **Risk** | 无 |
| **Change Type** | 不变 |

### [SWRD\_CM\_CommunicationAPI\_00107]service proxy重新连接 (Re-establishing service connection)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00107 |
| **Type** | Valid |
| **Description** | 如果出现连接异常，记录日志，当重新收到offer service消息后，重新建立连接 |
| **Upstream ID** | [SWS\_CM\_10491] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 断开Skeleton连接，再重新建立Skeleton连接 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00108]service proxy析构 (Destruction of service proxy)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00108 |
| **Type** | Valid |
| **Description** | proxy析构时：   1. 销毁method或Field的Set，Get调用返回的Promise和对应的Future对象 2. 调用Promise的析构函数，使Future，返回错误码broken\_promise |
| **Upstream ID** | [SWS\_CM\_10446] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 析构service proxy |
| **Risk** | 需要实现Promise析构的时候，它返回的future对象得到通知，当调用future.get()时，返回broken\_promise错误码。  这个需求需要确认std::promise和std::future是否能够支持。 |
| **Change Type** | 不变 |

## event订阅 (Service event subscription)

### [SWRD\_CM\_CommunicationAPI\_00109]event订阅方法(Method to subscribe to a service event)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00109 |
| **Type** | Valid |
| **Description** | 1. 提供event订阅函数，原型：   ara::core:Result<void> Event::Subscribe(size\_t maxSampleCount);   1. 参数maxSampleCount表明event的缓冲区大小 2. 当已经订阅的event再次调用Subscribe，如果maxSampleCount与之前相同，直接返回。如果maxSampleCount与之前不同，返回错误码ComErrc::kMaxSampleCountNotRealizable 3. 当Subscribe正常返回，maxSampleCount指定大小的缓冲区必须已经申请好 4. 不同event对象的Subscribe方法可以重入，相同event对象并发调用Subscribe方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00141], [SWS\_CM\_00700] , [SWS\_CM\_00022] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00110]event取消订阅方法(Method to unsubscribe from a service event)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00110 |
| **Type** | Valid |
| **Description** | 1. 提供event订阅函数，原型：   void Event::Unsubscribe();   1. 当event没有被订阅，调用Unsubscribe方法默认返回 2. 不同event对象的Unsubscribe方法可以重入，相同event对象并发调用Unsubscribe方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00151], [SWS\_CM\_00023] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00111]查询订阅状态 (Query Subscription State)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00111 |
| **Type** | Valid |
| **Description** | 1. 提供查询event订阅状态函数，原型：   ara::com::SubscriptionState GetSubscriptionState() const;   1. GetSubscriptionState返回的订阅状态与SubscriptionStateChangeHandler要一致 2. 不同event对象的GetSubscriptionState方法可以重入，相同event对象并发调用GetSubscriptionState方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00316], [SWS\_CM\_00024] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00112]注册订阅状态变化通知 (Set Subscription State change handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00112 |
| **Type** | Valid |
| **Description** | 1. 提供注册订阅状态变化通知函数，原型：   ara::core::Result<void> SetSubscriptionStateChangeHandler(ara::com::  SubscriptionStateChangeHandler handler);   1. event的订阅状态发生变化，需要马上调用SubscriptionStateChangeHandler 2. SubscriptionStateChangeHandler可以在运行时，动态的被改写 |
| **Upstream ID** | [SWS\_CM\_00333] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00113] Execution Context注册订阅状态变化通知 (Execution Context for setting Subscription State change handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00113 |
| **Type** | Valid |
| **Description** | 1. 提供注册订阅状态变化通知函数（Execution Context重载），原型：   template<typename ExecutorT>  ara::core::Result<void> SetSubscriptionStateChangeHandler(  ara::com::SubscriptionStateChangeHandler handler, ExecutorT&& executor);   1. 提供了一个executioner object参数，作为SubscriptionStateChangeHandler的执行上下文 2. 如果在executor中不能支持SubscriptionStateChangeHandler调用，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11354], [SWS\_CM\_11355] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00114] SetSubscriptionStateChangeHandler 可重入性(Re-entrancy - SetSubscriptionStateChangeHandler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00114 |
| **Type** | Valid |
| **Description** | 不同event对象的SetSubscriptionStateChangeHandler方法可以重入，相同event对象并发调用SetSubscriptionStateChangeHandler方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00025] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00115]取消注册订阅状态变化通知 (Unset Subscription State change handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00115 |
| **Type** | Valid |
| **Description** | 1. 提供取消注册订阅状态变化通知函数，原型：   void UnsetSubscriptionStateChangeHandler();   1. 不同event对象的UnsetSubscriptionStateChangeHandler方法可以重入，相同event对象并发调用UnsetSubscriptionStateChangeHandler方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00334], [SWS\_CM\_00026] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00116]调用订阅状态变化通知 (Call Subscription State change handler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00116 |
| **Type** | Valid |
| **Description** | kSubscriptionPending状态通知条件：   1. client订阅一个event，实际的订阅消息没有发出 2. client已经订阅了一个event，但是服务端的instance暂时不可用   kSubscribed状态通知条件：   1. client订阅一个event，实际的订阅关系已经建立成功 2. client已经订阅了一个event，但是event的订阅连接经历了断开又重新连接   在服务不可用又恢复可用后，订阅连接的需要自动重新连接。这种自动连接的行为，不依赖于network binding。且支持连接的属性更新后的自动连接。 |
| **Upstream ID** | [SWS\_CM\_00313], [SWS\_CM\_00314] , [SWS\_CM\_00315] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码，并调用 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## event接收 (Receive event)

### [SWRD\_CM\_CommunicationAPI\_00117]更新event cache (Method to update the event cache)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00117 |
| **Type** | Valid |
| Description | 1. 提供更新event cache同时接收event数据的函数，原型：   template <typename F>  ara::core::Result<std::size\_t> GetNewSamples(F&& f,  std::size\_t maxNumberOfSamples = std::numeric\_limits<std::size\_t>::max());   1. 输入参数maxNumberOfSamples指定本次调用最多接收的event数量 2. 输入参数可调用对象f，需要符合如下函数签名：   void(ara::com::SamplePtr<SampleType const>)   1. GetNewSamples操作流： 2. 从event（订阅时maxSampleCount指定）缓冲区获取event数据 3. 如果需要，反序列化数据 4. 将反序列化后的数据，放到局部缓存中 5. 将局部缓存中的数据使用SamplePtr（包括ProfileCheckStatus），通过调用用对象f，返回给用户 6. 当获取的event数量= maxNumberOfSamples，或SamplePtrs的数量达到maxSampleCount，或缓存区为空。结束本次调用 7. 达到maxSampleCount时，应用程序持有服务端提供的最多SamplePtr。 8. size\_t表示调用上下文中的样本数。 9. 返回值ara::core::Result<size\_t>可以是event数据通过f，回调的次数或错误码kMaxSamplesReached |
| **Upstream ID** | [SWS\_CM\_00701], [SWS\_CM\_00702] , [SWS\_CM\_00703] , [SWS\_CM\_00704] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00118] Execution Context更新event cache (Execution Context to update the event cache)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00118 |
| **Type** | Valid |
| Description | 1. 提供更新event cache同时接收event数据（Execution Context重载）的函数，原型：   template<typename ExecutorT>  ara::core::Result<size\_t> GetNewSamples(F&& f, size\_t maxNumberOfSamples = std::numeric\_limits<size\_t>::max(),  ExecutorT&& executor);   1. 提供了一个executioner object参数，作为GetNewSamples的执行上下文 2. 如果在executor中不能支持GetNewSamples调用，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11358], [SWS\_CM\_11359] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00119] GetNewSamples可重入性(Re-entrancy - GetNewSamples)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00119 |
| **Type** | Valid |
| **Description** | 不同event对象的GetNewSamples方法可以重入，相同event对象并发调用GetNewSamples方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00714] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00120] 提供获取E2E Result方法(Provide E2E Result)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00120 |
| **Type** | Valid |
| Description | 提供获取E2E结果的函数，原型：  const ara::com::e2e::Result GetResult() const; |
| **Upstream ID** | [SWS\_CM\_90424] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00121] 提供获取event缓冲区剩余空间方法(Query Free Sample Slots)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00121 |
| **Type** | Valid |
| Description | 1. 提供获取event缓冲区剩余空间的函数，原型：   size\_t GetFreeSampleCount() const noexcept;   1. 返回值为event缓冲区剩余空间的数量 2. 计算event缓冲区剩余空间： 3. 在调用Subscribe，maxSampleCount=N后，GetNewSamples之前，GetFreeSampleCount返回N 4. GetNewSamples上下文中每创建一个SamplePtr，N减1 5. SamplePtr使用完后，SamplePtr=std::nullptr\_t或SamplePtr析构，event缓冲区剩余空间的数量加1 6. GetFreeSampleCount需要实现为可重入 |
| **Upstream ID** | [SWS\_CM\_00705], [SWS\_CM\_00706] , [SWS\_CM\_00707] , [SWS\_CM\_00027] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00122] 提供注册event接收通知方法(Enable service event trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00122 |
| **Type** | Valid |
| Description | 1. 提供注册event接收通知的函数，原型：   ara::core::Result<void> Event::SetReceiveHandler(  ara::com::EventReceiveHandler handler);   1. 参数EventReceiveHandler签名std::function<void()> 2. 如果注册失败，返回ComErrc::kSetHandlerNotSet错误码 |
| **Upstream ID** | [SWS\_CM\_00181] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00123] Execution Context提供注册event接收通知方法(Execution Context for enabling service event trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00123 |
| **Type** | Valid |
| Description | 1. 提供注册event接收通知（Execution Context重载）的函数，原型：   template<typename ExecutorT>  ara::core::Result<void> Event::SetReceiveHandler(  ara::com::EventReceiveHandler handler, ExecutorT&& executor);   1. 提供了一个executioner object参数，作为SetReceiveHandler的执行上下文 2. 如果在executor中不能支持SetReceiveHandler调用，返回ComErrc::kCouldNotExecute错误码 |
| **Upstream ID** | [SWS\_CM\_11356], [SWS\_CM\_11357] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00124] SetReceiveHandler可重入性(Re-entrancy - SetReceiveHandler)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00124 |
| **Type** | Valid |
| **Description** | 不同event对象的SetReceiveHandler方法可以重入，相同event对象并发调用SetReceiveHandler方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00028] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码并调用 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00125] 提供反注册event接收通知方法(Disable service event trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00125 |
| **Type** | Valid |
| Description | 1. 提供反注册event接收通知的函数，原型：   ara::core::Result<void> Event::UnsetReceiveHandler();   1. 如果反注册失败，返回ComErrc::kUnsetFailure错误码 2. 不同event对象的UnsetReceiveHandler方法可以重入，相同event对象并发调用UnsetReceiveHandler方法，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00183], [SWS\_CM\_00029] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## trigger订阅（Service Trigger subscription）

### [SWRD\_CM\_CommunicationAPI\_00126]trigger订阅方法 (Method to subscribe to a service trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00126 |
| **Type** | Valid |
| Description | 1. 特定ServiceProxy 类的特定Trigger 类中，应提供Subscribe 方法来启动相应Trigger的订阅。 2. 如果调用时，Trigger是已订阅状态，那么该接口直接返回，不做其他处理。 3. 对于不同的Trigger实例，订阅应该是可重入的和线程安全的。   在同一个 Trigger 类实例上重入或同时调用时，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00723] [SWS\_CM\_00724] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00127]trigger取消订阅方法 (Method to unsubscribe to a service trigger)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00127 |
| **Type** | Valid |
| Description | 1. 特定ServiceProxy 类的特定Trigger 类中，应提供UnSubscribe 方法来取消相应Trigger的订阅。 2. 如果调用时，Trigger是未订阅状态，那么该接口直接返回，不做其他处理。 3. 对于不同的Trigger实例，订阅应该是可重入的和线程安全的。   在同一个 Trigger 类实例上重入或同时调用时，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00810] [SWS\_CM\_00035] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## trigger接收

### [SWRD\_CM\_CommunicationAPI\_00128] 更新Trigger计数的方法(Method to update the trigger counter)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00128 |
| **Type** | Valid |
| Description | 1. 通信管理应该提供GetNewTriggers方法来更新Trigger计数。 2. 在调用GetNewTriggers的上下文中通信管理应该能取得自上次调用GetNewTriggers以来的所有trigger触发的次数。 3. 返回值size\_t标示了自上次调用GetNewTriggers依赖trigger触发的次数。（0代表没收到新的trigger） 4. 对于不同的Trigger实例，GetNewTriggers应该是可重入的和线程安全的。在同一个 Trigger 类实例上重入或同时调用时，行为未定义 |
| **Upstream ID** | [SWS\_CM\_00226][SWS\_CM\_00227] [SWS\_CM\_00228] [SWS\_CM\_11251] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 新增 |

### [SWRD\_CM\_CommunicationAPI\_00129] trigger接收回调 (Receive trigger by getting triggered)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00129 |
| **Type** | Valid |
| **Description** | 1. 为了使应用能接收到ServiceProxy中对应的trigger触发时的通知。应该提供SetReceiveHandler来注册trigger触发时调用的回调函数。 2. TriggerReceiveHandler构成一个没有参数的函数，并且必须使用对应trigger类中GetNewTriggers方法访问收到的trigger计数。 3. 如果调用SetReceiveHandler()失败，应该返回ComErrc::kSetHandlerNotSet。 |
| **Upstream ID** | [SWS\_CM\_00182], [SWS\_CM\_00711] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 调用相应的接口函数 |
| **Risk** | 无 |
| **Change Type** | 新增 |

## method调用 (Call a service method)

### [SWRD\_CM\_CommunicationAPI\_00130] method类定义 (Initiate a method call)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00130 |
| **Type** | Valid |
| **Description** | 1. method类命名约定：   ServiceInterface.method with ClientServerOperation.fireAndForget=false  ServiceInterface.method.shortName为method类名   1. method输出参数约定：   (ClientServerOperation.arguments with ArgumentDataPrototype.  direction set to out or inout) and optional return values shall be provided.   1. 提供operator()操作符约定：   输入参数为ClientServerOperation.arguments with ArgumentDataPrototype.direction set to in or inout  返回值为ara::core::Future包裹的输出参数类型   1. 例如：   class Method {  struct Output {  TypeOutputParameter1 output1;  TypeOutputParameter2 output2;  ...  TypeResult result; // return value (optional)  };  ara::core::Future<Output> operator()(  TypeInputParameter1 input1,  TypeInputParameter2 input2,  ...  );  };   1. method调用马上返回，调用者通过ara::core::Future选择同步或者异步方式获取method调用的结果（包括错误） 2. operator()操作符需要实现为可重入 3. 取消method调用：   返回的ara::core::Future对象被销毁，会导致取消method调用，需要释放相关的资源，不把结果返回给调用者   1. 使用ara::core::Future.GetResult获取method的调用结果。如果调用结果还没有接收，则GetResult会阻塞，直到接收到调用结果。 2. 作为替代，可以使用ara::core::Future.get获取GetResult里面的值或异常 3. 同步方法调用：   使用ara::core::Future的get(), wait(), wait\_for(), wait\_until().  允许在调用上述方法阻塞的时候，CM进行上下文切换   1. 如果CM检测到错误，使用ara::core::Promise.SetError，将kNetworkBinding-Failure错误码返回给对应的Future对象 2. 异步方法调用：   polling：使用ara::core::Future.is\_ready查询，如果is\_ready返回true，调用get方法会立即返回  notification：使用ara::core::Future.then方法注册通知函数 |
| **Upstream ID** | [SWS\_CM\_00196], [SWS\_CM\_00032], [SWS\_CM\_00194] , [SWS\_CM\_00195], [SWS\_CM\_00192] , [SWS\_CM\_10440] , [SWS\_CM\_00193] , [SWS\_CM\_00197] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 析构ara::core::Future对象时，需要取消method调用。  需要实现析构ara::core::Future对象，对应的ara::core::Promise对象也得到通知 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00131] Fire and Forget method类定义 (Initiate a Fire and Forget method call)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00131 |
| **Type** | Valid |
| **Description** | 1. fire and forget method类命名约定：   ServiceInterface.method with ClientServerOperation.fireAndForget=true  ServiceInterface.method.shortName为method类名   1. 提供operator()操作符约定：   输入参数为ClientServerOperation.arguments with ArgumentDataPrototype.direction set to in  返回值为void   1. 例如：   class FireAndForgetMethod {  void operator()(  TypeInputParameter1 input1,  TypeInputParameter2 input2,  ...  );  }; |
| **Upstream ID** | [SWS\_CM\_90435] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 不变 |

## field方法 (method for fields)

### [SWRD\_CM\_CommunicationAPI\_00132]field get方法(Get method for fields)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00132 |
| **Type** | Valid |
| Description | 1. 提供field的get函数，原型：   ara::core::Future<FieldType> Get();   1. Field.hasGetter决定是否有get方法 2. get方法需要实现为可重入 |
| **Upstream ID** | [SWS\_CM\_00112], [SWS\_CM\_00132] , [SWS\_CM\_00030] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

### [SWRD\_CM\_CommunicationAPI\_00133]field set方法(Set method for fields)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00133 |
| **Type** | Valid |
| Description | 1. 提供field的set函数，原型：   ara::core::Future<FieldType> Set(const FieldType& value);   1. Field.hasSetter决定是否有set方法 2. set方法需要实现为可重入 |
| **Upstream ID** | [SWS\_CM\_00113], [SWS\_CM\_00133] , [SWS\_CM\_00031] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 无 |
| **Change Type** | 变更 |

## Instance Specifier转换 (Instance Specifier Translation)

### [SWRD\_CM\_CommunicationAPI\_00134] Instance Specifier转换方法(Method Instance Specifier Translation)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_CM\_CommunicationAPI\_00134 |
| **Type** | Valid |
| Description | 1. 提供InstanceSpecifier转换函数，原型：   ara::core::Result<ara::com::InstanceIdentifierContainer>  ara::com::runtime::ResolveInstanceIDs(ara::core::InstanceSpecifier modelName);   1. 返回的InstanceIdentifierContainer数量可以是0,1,N 2. 如果失败，返回ComErrc::kInstanceIDCouldNotBeResolved错误码 |
| **Upstream ID** | [SWS\_CM\_00118] |
| **Dependencies** | - |
| **ASIL** | QM |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 检查生成代码 |
| **Risk** | 依赖InstanceSpecifier的映射实现 |
| **Change Type** | 变更 |

# 非功能需求(Non-Functional Requirements)

## 制约（Constraint）

### [SWRD\_NF\_CM\_CommunicationAPI\_00001]异步回调(Non-Function Requirement No.1)

|  |  |
| --- | --- |
| **SWRD\_ID** | SWRD\_NF\_CM\_CommunicationAPI\_00001 |
| **Type** | Valid |
| **Description** | CM中提供的注册接口，在异步回调的时，能够保证回调上下文的有效性 |
| **Upstream ID** | - |
| **Dependencies** | - |
| **ASIL** | - |
| **Status** | [In review] |
| **Priority** | H |
| **Verification Criteria** | 异步调用 |
| **Risk** | - |
| **Change Type** | 新增 |

# 接口说明（API）

## 错误类型(Error types)

### [SWRD\_API\_CM\_CommunicationAPI\_00001]ComErrc枚举（ComErrc enum）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00001 | |
| **Upstream ID** | [SWS\_CM\_10432] | |
| **Consistency** | Yes | |
| **Change Type** | 变更 | |
| ***Kind:*** | enumeration | |
| **Symbol:** | ComErrc | |
| **Scope:** | namespace ara::com | |
| **Syntax:** | ara::core::ErrorDomain::CodeType  Syntax: enum class ComErrc : ara::core::ErrorDomain::CodeType {...}; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | ComErrc枚举定义了ComErrorDomain的错误码。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00002]CM异常类（Com Exception）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00002 | |
| **Upstream ID** | [SWS\_CM\_11327] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | class | |
| **Symbol:** | ComException | |
| **Scope:** | namespace ara::com | |
| **Syntax:** | class ComException : public Exception {...}; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 为通信api抛出的异常定义一个类。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00003] ComException构造函数（ComException Construct）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00003 | |
| **Upstream ID** | [SWS\_CM\_11328] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | ComException(ara::core::ErrorCode errorCode) | |
| **Scope:** | class ara::com::ComException | |
| **Syntax:** | explicit ComException (ara::core::ErrorCode errorCode) noexcept; | |
| **Parameters (in):** | errorCode |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | ComException的构造函数。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00004] ComErrorDomain类（Class ComErrorDomain）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00004 | |
| **Upstream ID** | [SWS\_CM\_11329] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | class | |
| **Symbol:** | ComErrorDomain | |
| **Scope:** | namespace ara::com | |
| **Syntax:** | class ComErrorDomain final : public ErrorDomain {...}; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 定义一个表示通信错误域的类。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00005] Errc别名(using Errc)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00005 | |
| **Upstream ID** | [SWS\_CM\_11336] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | type alias | |
| **Symbol:** | Errc | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | using Errc = ComErrc; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 错误码值枚举的别名。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00006] ComErrorDomain构造函数（ComErrorDomain constructor）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00006 | |
| **Upstream ID** | [SWS\_CM\_11330] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | ComErrorDomain() | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | constexpr ComErrorDomain () noexcept; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | ComErrorDomain构造函数。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00007] Exception别名（using Exception）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00007 | |
| **Upstream ID** | [SWS\_CM\_11337] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | type alias | |
| **Symbol:** | Exception | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | using Exception = ComException; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | Exception的别名。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00008] name函数（name function）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00008 | |
| **Upstream ID** | [SWS\_CM\_11331] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | Name() | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | const char\* Name () const noexcept override; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 返回一个与ComErrorDomain相关的字符串常量。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00009] message函数(message function)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00009 | |
| **Upstream ID** | [SWS\_CM\_11332] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | Message(CodeType errorCode) | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | const char\* Message (CodeType errorCode) const noexcept override; | |
| **Parameters (in):** | errorCode | 错误码 |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 返回与errorCode相关联的消息。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00010] ThrowAsException函数(ThrowAsException function)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00010 | |
| **Upstream ID** | [SWS\_CM\_11333] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | ThrowAsException(const ara::core::ErrorCode &errorCode) | |
| **Scope:** | class ara::com::ComErrorDomain | |
| **Syntax:** | void ThrowAsException (const ara::core::ErrorCode &errorCode) const  noexcept(false) override; | |
| **Parameters (in):** | errorCode | 抛出的错误。 |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 描述:从errorCode中创建一个新的ComException实例，并将其作为c++异常抛出。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00011] GetComErrorDomain函数(GetComErrorDomain function)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00011 | |
| **Upstream ID** | [SWS\_CM\_11334] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | GetComErrorDomain() | |
| **Scope:** | namespace ara::com | |
| **Syntax:** | constexpr ara::core::ErrorDomain& GetComErrorDomain () noexcept; | |
| **Parameters (in):** |  |  |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 返回对全局ComErrorDomain对象的引用。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00012] MakeErrorCode函数(MakeErrorCode function)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00012 | |
| **Upstream ID** | [SWS\_CM\_11335] | |
| **Consistency** | Yes | |
| **Change Type** | 新增 | |
| ***Kind:*** | function | |
| **Symbol:** | MakeErrorCode(ara::com::ComErrc code, ara::core::ErrorDomain::SupportDataType data) | |
| **Scope:** | namespace ara::com | |
| **Syntax:** | constexpr ara::core::ErrorCode MakeErrorCode(ComErrc code,  ara::core::ErrorDomain::SupportDataType data  r) noexcept; | |
| **Parameters (in):** | code | 错误号。 |
| data | 供应商定义的与错误相关的数据。 |
| **Header file:** | #include "ara/com/com\_error\_domain.h" | |
| **Description:** | 返回对全局ComErrorDomain对象的引用。 | |
| **Additional:** | 无 | |

### [SWRD\_API\_CM\_CommunicationAPI\_00013] 应用程序错误的定义(Definition of Application Errors)

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | SWRD\_API\_CM\_CommunicationAPI\_00013 | |
| **Upstream ID** | [SWS\_CM\_11266] | |
| **Consistency** | No | |
| **Change Type** | 不变 | |
| ***Kind:*** | enum | |
| **Symbol:** |  | |
| **Scope:** |  | |
| **Syntax:** | enum class <ApApplicationErrorDomain.SN>Errc : ara::core::ErrorDomain::  CodeType  {  <ApApplicationError.SN> = <ApApplicationError.errorCode>,  }; | |
| **Parameters (in):** |  |  |
| **Header file:** |  | |
| **Description:** | 每个ApApplicationError都引用一个ApApplicationErrorDomain。ApApplicationErrorDomain的枚举定义。 | |
| **Additional:** | 无 | |

## 接口头文件（API Header files）

*[记录AutoSar中接口头文件信息]*

## 接口共同数据类型（API Common Data Types）

*[记录AutoSar中共同数据类型信息]*

## 接口定义（API Reference）

### [SWRD\_ID]接口1（API 1）

*[表格中前四行信息必须保留原有样式，后续内容可以根据自己需求增减内容，原则上AutoSar标准文档中包含的字段必须保留]*

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** | *SWRD\_ID* | |
| **Upstream ID** | *AutoSAR的SWS-ID* | |
| **Consistency** | *Yes/No，Yes:与AutoSar标准一致；No:非标* | |
| **Change Type** | *新增/修改/删除/不变* | |
| ***Kind:*** | *function* | |
| **Symbol:** | *ara::phm::SupervisedEntity::SupervisedEntity(InstanceSpecifier const &instance)* | |
| **Scope:** | *class ara::phm::SupervisedEntity* | |
| **Syntax:** | *explicit SupervisedEntity (InstanceSpecifier const &instance);* | |
| **Parameters (in):** | *instance* | *instance specifier of the supervised entity.* |
| **Header file:** | *#include "ara/phm/supervised\_entity.h"* | |
| **Description:** | *Creation of a SupervisedEntity.* | |
| **Additional:** | *在配置工具上进行配置，配置工具生成代码，生成instance，使用生成的instance。* | |

### [SWRD\_ID]接口2（API 2）

|  |  |  |
| --- | --- | --- |
| **SWRD\_ID*:*** |  | |
| **Upstream ID** |  | |
| **Consistency** |  | |
| **Change Type** |  | |
| ***Kind:*** |  | |
| **Symbol:** |  | |
| **Scope:** |  | |
| **Syntax:** |  | |
| **Parameters (in):** |  |  |
| **Header file:** |  | |
| **Description:** |  | |
| **Additional:** |  | |

附录A- 信息定义

|  |  |  |
| --- | --- | --- |
| 类别 | 结构 | 备注 |
| 需求 | SWRD\_{需求类型}\_{功能简称}\_流水号  功能简称：参见下面功能简称列表  需求类型：功能需求为空，非功能需求为NF,接口为API  流水号：从00001开始的5位自然数 | *例：*  *SWRD\_Nvm\_00001*  *SWRD\_NF\_Nvm\_00001*  *SWRD\_API\_Nvm\_00001* |

|  |  |
| --- | --- |
| 功能简称列表（aCore） | 说明 |
| DM\_DEM | Diagnostics management模块的诊断事件管理 |
| DM\_DCM | Diagnostics management模块的诊断通信管理 |
| DM\_DCM\_DOIP | Diagnostics management模块的DO/IP相关功能 |
| CoreTypes | 核心数据类型 |
| CM\_SOMEIP | Communication management模块的SOME/IP相关功能 |
| CM\_DDS | Communication management模块的DDS相关功能 |
| CM\_CommunicationGroup | Communication management模块的通信组相关功能 |
| CM\_SHM | Communication management模块的共享内存相关功能 |
| CM\_IPC | Communication management模块的IPC相关功能 |
| CM\_Raw | Communication management模块的raw data streaming相关功能 |
| CM\_TLS | Communication management模块的TLS相关功能 |
| CM\_S2S | Communication management模块的S2S相关功能 |
| CM\_E2E | Communication management模块的E2E相关功能 |
| UCM\_Master | Update and config management模块的主站相关功能 |
| UCM\_Server | Update and config management模块的从站相关功能 |
| LT | Log and trace模块相关功能 |
| PHM | Platform health management模块相关功能 |
| Per | Persistency模块相关功能 |
| SM | State management模块相关功能 |
| Crypto | Cryptography模块相关功能 |
| EM | Execution mangement模块相关功能 |
| NM | Network management模块相关功能 |
| TS | Time synchronization模块相关功能 |

说明：根据项目情况可自己定义，增加功能简称

|  |  |  |
| --- | --- | --- |
| 安全等级(ASIL) | 解释说明 | 备注 |
| ASIL A | 根据S – Severity(严重度)  E – Exposure（暴露度） C – Controllability（可控性） 排定功能安全等级。详细理解可以参考26262标准文件。 | *如果有关于ASIL等级的特殊解释说明，请记录在此* |
| ASIL B |  |
| ASIL C |  |
| ASIL D |  |
| QM(A) | 从ASIL A到ASIL D 中拆分出来，拆分的标准，参考功能安全体系文件《功能安全需求分解指南\_FS.pdf》 |  |
| QM(B) |  |
| QM(C) |  |
| QM(D) |  |
| ASIL A(A) |  |
| ASIL A(B) |  |
| ASIL A(C) |  |
| ASIL A(D) |  |
| ASIL B(B) |  |
| ASIL B(C) |  |
| ASIL B(D) |  |
| ASIL C(C) |  |
| ASIL C(D） |  |
| ASIL D(D) |  |

|  |  |  |
| --- | --- | --- |
| 优先级（Priority） | 解释说明 | 备注 |
| H | 高优先级 | *例：被依赖的需求优先级设置为H级别* |
| M | 中优先级 | *例：* |
| L | 低优先级 | *例：其余功能均设置为L级别* |

|  |  |  |
| --- | --- | --- |
| 状态  （Status） | 状态说明 | 备注 |
| Draft | 草稿 | *例：表示新建* |
| In Review | 评审中 | *例：表示处于评审中* |
| Approved | 批准 | *例：表示通过评审* |
| Released | 发布 | *例：表示通过客户确认* |
| Modified | 修改 | *例：表示正在检讨修改中* |

|  |  |  |
| --- | --- | --- |
| 类型  （Type） | 状态说明 | 备注 |
| Valid | 有效 | *例：表示需要对应* |
| InValid | 不适用 | *例：表示不做对应* |
| TBD | 检讨中 | *例：表示正在检讨中* |

说明：根据项目情况可自己定义，但需要明确

|  |  |  |
| --- | --- | --- |
| 变更类型  (Change Type) | 解释说明 | 备注 |
| 新增 | 相对已建立的第一版需求基线（含Base项目的需求基线），如果是新增的需求，选择此项 | 如果有关于每个变更类型的特殊解释说明，请记录在此 |
| 修改 | 相对已建立的第一版需求基线（含Base项目的需求基线），发生了修改的需求 |  |
| 不变 | 相对已建立的base项目的需求基线，复用了base项目的需求，填此类型，如没有Base项目需求基线，不应填此类型。 |  |
| 删除 | 相对已建立的第一版需求基线（含Base项目的需求基线），如果是删除的需求，选择此项。 |  |

说明：根据项目情况可自己定义，但需要明确

附录B- 配置信息

|  |  |  |  |
| --- | --- | --- | --- |
| 配置信息 | 说明 | 范围 | 备注 |
| *API configuration class* |  | *1、2、3* |  |
|  |  |  |  |
|  |  |  |  |