修订记录

| 日期 | 修订版本 | CR号 | 修改章节 | 修改描述 | 作者 |
| --- | --- | --- | --- | --- | --- |
| 2001-06-08 | A |  |  | 初稿完成 | 姓名+工号 |
| yyyy-mm-dd | B | xxx | x.x.x; y.y.y | 修改XXX   1. Xxx 2. Xxx 3. ... | 姓名+工号 |
| xxx | x.x.x; y.y.y | 修改XXX   1. Xxx 2. Xxx 3. ... |
|  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |
|  |  |  |

目 录

1 简介 7

1.1 目的 7

1.2 范围 7

1.2.1 软件名称 7

1.2.2 软件功能 7

2 需求分析 7

2.1 需求描述 7

2.1.1 需求编号 7

2.2 竞品分析 7

2.2.1 竞品表现 7

2.2.2 竞品实现方案 7

2.2.3 竞品指标目标 7

2.3 设计约束 7

2.3.1 方案依赖项 7

2.3.2 需要遵守的全局规范和方案 7

2.3.3 性能指标 7

2.3.4 功耗指标 8

2.3.5 内存指标 8

2.3.6 外部存储空间使用 8

2.3.7 外部依赖技术 8

2.3.8 Google规范遵从性 8

3 方案选型分析 8

4 第零层设计描述 8

4.1 软件系统上下文定义 8

4.2 设计约束 8

4.2.1 遵循标准 9

4.2.2 硬件限制 9

4.2.3 技术限制 9

4.2.4 系统配置 9

4.3 设计思路(可选) 9

5 第一层设计描述 9

5.1 总体结构 10

5.1.1 模块划分 10

5.2 分解描述 10

5.2.1 模块/子系统分解 10

5.2.2 并发进程处理分解 10

5.2.3 数据分解 10

5.3 依赖性描述 11

5.3.1 运行设计 11

5.3.2 数据依赖关系 11

5.4 接口描述 11

5.4.1 模块/子系统接口 11

5.4.2 进程接口 12

6 第二层设计描述 13

6.1 模块1名称 13

6.1.1 模块1总体结构 13

6.1.2 模块分解描述 13

6.1.3 数据实体描述 13

6.1.4 依赖性描述 13

6.1.5 接口描述 13

7 引擎jni 接口 13

7.1 数据描述 14

7.1.1 简单数据描述； 14

7.1.2 结构1 14

7.1.3 结构2 14

7.2 函数描述 16

7.2.1 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_initAsr(JNIEnv\* env, jobject jobj, jstring jpath, jstring jparams) 16

int32\_t InitASR(const string& path, const std::string& para) 16

7.2.2 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_releaseAsr(JNIEnv\* env, jobject jobj) 17

int32\_t RealseASR() 17

7.2.3 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_startRecognition(JNIEnv\* env, jobject jobj, jstring taskId, jstring words, jstring jparams) 18

int32\_t InitSession(const string& sessionId, const std::string& sessPara, const std::string& userWords) 18

7.2.4 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_importWords(JNIEnv\* env, jobject jobj, jstring type, jstring words, jstring version, jstring deviceName) 20

int32\_t ImportWordSession(const std::string& wordsChar, const std::string& type, uint32\_t weight, const std::string& deviceName) 20

7.2.5 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_sendAudio(JNIEnv\* env, jobject jobj, jstring taskId, jbyteArray frame, jint streamLen) 21

int32\_t SendAudio(const string& sessionId, char\* audioStream, uint32\_t length) 21

是否多次调用 21

7.2.6 JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_finishRecognition(JNIEnv\* env, jobject jobj, jstring taskId) 22

int32\_t FinishSession(const string& sessionId) 22

8 模块2详细设计 26

8.1 26

9 非功能设计 26

9.1 DFX设计 26

9.1.1 可靠性设计 26

9.1.2 可扩展性(模块化)隔离设计 26

9.1.3 可配置性设计 26

9.1.4 可维护性设计 26

9.1.5 可生产性设计 26

9.1.6 安全性设计 26

9.1.7 兼容性设计 26

9.1.8 可测试性设计 26

9.2 性能设计 26

9.2.1 开机内存基线遵从性说明 27

9.2.2 动态内存基线遵从性说明 27

9.2.3 CPU基线遵从性说明 27

9.2.4 广播基线遵从性说明 27

9.2.5 静态性能基线遵从性说明 27

9.2.6 组件动态性能基线遵从性说明 27

9.2.7 组件静态性能基线遵从性说明 27

9.2.8 竞品比拼-一生快基线遵从性说明 27

9.2.9 竞品比拼-核心场景基线遵从性说明 27

9.3 功耗设计 27

9.3.1 应用场景功耗基线遵从性说明 27

9.3.2 应用服务后台功耗基线遵从性说明 27

9.4 飞马架构设计 27

9.4.1 可移植设计（解耦、升级） 27

9.4.2 ROM占用设计 27

9.5 全球化设计 27

9.6 可扩展性设计 28

9.7 UI一致性设计自检 28

9.8 设计IPR自检 28

10 性能指标 28

11 部署 28

表目录

Table1 \*\*表 **错误！未定义书签。**

表1 \*\*表 **错误！未定义书签。**

图目录

Figure 1 \*\*图 **错误！未定义书签。**

图 1 \*\*图 **错误！未定义书签。**

XX 软件设计说明书Software design specification

*(Title must be in both Chinese and English to facilitate English speaking users' search为方便英语用户检索，此处必须提供中英文版)*

Keywords关键词：*Provide Keywords in both Chinese and English to facilitate English speaking users' search. 为方便英语用户检索，此处必须提供中英文版*

Abstract摘 要：*Provide Abstract in both Chinese and English to facilitate English speaking users' search. 为方便英语用户检索，此处必须提供中英文版*

缩略语清单：

<对本文所用缩略语进行说明，要求提供每个缩略语的英文全名和中文解释。>

| 缩略语 | 英文全名 | 中文解释 |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 简介

## 目的

这部分要描述文档的目的。应该指明读者。

## 范围

### 软件名称

对软件命名

### 软件功能

解释软件产品将完成或不完成的功能（可以直接描述也可以参考相关文档）

# 需求分析

## 需求描述

### 需求编号

|  |  |
| --- | --- |
| SR编号 | SR描述 |
|  |  |

## 竞品分析

### 竞品表现

### 竞品实现方案

### 竞品指标目标

## 设计约束

### 方案依赖项

### 需要遵守的全局规范和方案

### 性能指标

### 功耗指标

### 内存指标

### 外部存储空间使用

### 外部依赖技术

### Google规范遵从性

# 方案选型分析

方案选项要说明为什么这么选, 比如和友商比咱们的优势是什么

# 第零层设计描述

## 软件系统上下文定义

首先描述本软件系统在实际环境的组网方式（一般用图描述）

描述本系统如何与外部实体一起组成一个功能性的实体（一般用图描述）

本节还要描述待开发软件系统与外部实体的关系.，可以使用系统架构图、顶层数据流图、事务图等来描述系统结构和交互关系。

所有系统外围实体(在设计规格流程中有定义)，这些实体与软件系统发生交互作用或以某种方式影响软件系统，比如，操作系统，驱动程序，硬件，中断，其它软件系统等.

外围实体和本系统相互作用的性质(比如，消息，库调用)和类型(比如，OS服务，通信服务，系统工具等)。描述外部实体的特性时要注意该描述一定要有助本软件的设计和描述。 为了本文档内容的完整性，可以采用对于描述相关设计实体文档索引的方式：如对于操作系统编程手册的索引

注：文档中出现的所有的表号都应该放在表格的上方，所有的图号都应该放在图形的下方

## 设计约束

列出对软件设计有重要影响的系统内外部约束和限制，例如：

软件环境，包括操作系统和相关软件的设置，软件总体框图

用户环境

系统和相关资源

遵循的标准

界面和接口需求

数据存储和分发需求

内存或其它资源限制

安全和可靠性需求

性能需求

测试和可维护性需求

上面列举的例子并不是充分或必须的，你可以根据自己的需要增加或删除内容。

例子（下面的章节目录可根据需要增删）：

### 遵循标准

### 硬件限制

### 技术限制

### 系统配置

## 设计思路(可选)

从以下几个方面进行考虑：

设计原则：设计的目标/原则/优先级等

设计方法：使用的设计模式（层模式，微内核模式，代理模式等），软硬件平台对设计思路的影响

设计方案：对本软件系统的几种设计方案进行分析、比较，并确定所采用的方案

软件复用：对已有软件模块的复用

可扩展性/兼容性：业务能力扩展的设计思路，对以前版本的兼容，平滑升级

性能：延时，吞吐量，在既定硬件环境的约束下采用的性能实现思路

可靠性：故障检测，故障隔离，故障恢复，容错，冗余，备份

多进程/多线程：同步，并发，保护，防死锁

其它需要后续开发/维护人员知道的内容，如本软件特别的地方（编程语言，数据库，库文件等）

# 第一层设计描述

总体方案描述。需要描述各功能模块的输入输出。

本章节描述系统功能如何被分解到模块或子系统中。此处不用描述每个模块的具体实现细节（在后续的子章节中有描述）。本章节的目的是描述清楚系统为什么和怎样被分解，各个模块如何协调工作实现系统功能（必要时可以描述其它系统分解方法为何没有被采纳）

注意：本章节（包括子章节）仅仅适用于系统中新开发的模块。如果在原有系统上进行扩展，则除了新模块外，旧模块中仅需要描述相关的地方。描述的原则是，读者能够理解你进行的修改或扩展。

## 总体结构

这里要描述软件系统的总体结构（包括资源/内存分配，Flash资源/文件分配，配置）。总体结构描述可以使用结构图，层次分解图，数据流图，自然语言。

例子（下面的章节目录可根据需要增删）：

### 模块划分

## 分解描述

### 模块/子系统分解

本节描述系统中的子系统和模块。

下面的每一个子章节按照以下格式描述（可以根据需要增加或者删除）：

标识，实体的名称。任意两个实体不能重名。实体的名称要尽量反映实体的特征

类型，描述实体种类，分类方法要通篇保持一致。

目的，描述实体为什么存在。

功能列表，说明实体做什么。

处理，描述实体为了实现其功能所采用的方法，它是对功能的细化

要实现的需求ID(可选):

子模块(可选)： 如果本模块还要继续划分，列出下属子模块

#### 模块/子系统1描述

#### 模块/子系统2描述

### 并发进程处理分解

本节描述系统中的并发进程，仿照“模块/子系统分解”的格式写作。

#### 进程1描述

描述本进程中要完成的功能，如功能已经在前面的模块分解中有描述，则应重点描述这些模块的哪些功能是在本进程中实现的。

### 数据分解

本节描述系统中的数据结构。

直接访问的数据，被各分解模块用来在接口中使用的结构。描述数据时要注意该数

外部数据实体不必描述。

本章节描述本系统中用到的全局数据结构，其中至少包括： 各分解模块据是否真正为本级模块分解的公用数据。

#### 数据实体1描述

按照以下格式描述：

标识：

类型：

目的：

定义：使用表格或编程语言定义该数据结构

## 依赖性描述

本节描述系统中的子系统，数据结构，模块，进程等设计实体间的关系。

依赖关系描述可以使用结构图（交互）事务图，消息序列图, ER图。

### 运行设计

本章节描述本层设计的各个模块和进程是如何配合完成系统的功能需求. 可以用消息序列图, (交互)事务图, 结构图等进行描述, 可以按照处理过程进行分节描述, 要注意的是描述的内容应该涵盖所有SRS文档的功能需求, 对于二级设计来讲, 就是要涵盖分解对象模块的功能规格

### 数据依赖关系

本节描述数据结构间的关系。

依赖关系描述可以使用ER图。

## 接口描述

本节描述软件系统中设计实体(如子系统，模块，进程）的接口.

接口描述可以使用接口文件，参数表。

对于外部实体只有同被描述软件相关的接口才需描述。

本章节涉及的数据类型应该在前面的“数据分解”中描述类型定义

### 模块/子系统接口

本节描述模块/子系统的接口。

#### 模块/子系统1提供的接口

对于接口函数，可以采用以下形式描述：

接口1：

形式：

输入：

输出：

返回：描述返回什么。

描述：总体描述该接口实现的目的。

### 进程接口

本节描述进程的接口。

#### 进程1描述

进程间的接口主要是描述一些共享内存，协议数据，消息等。

按照以下格式描述：

标识：

类型：

目的：

功能列表：

# 第二层设计描述

L1中定义的每个实体的进一步设计在下面的章节进行描述。如果某个模块不需要某章节，比如没有进一步分解成子模块，则可直接删除相关章节。

对每个模块重复使用下述的格式。

## 模块1名称

不要直接写“模块1名称”，用简短的词语命名模块.

### 模块1总体结构

这里要描述本模块的总体结构。本章节的目的是描述清楚模块为什么和怎样被分解，各个子模块如何协调工作实现系统功能

例子：

本模块被分解成3个子模块，如下图所示：

……

### 模块分解描述

描述模块的分解，并描述每个子模块的功能等属性。

下面的每一个子章节按照以下格式描述（可以根据需要增加或者删除）：

标识，实体的名称。任意两个实体不能重名。实体的名称要尽量反映实体的特征

类型，描述实体种类，分类方法要通篇保持一致。

目的，描述实体为什么存在。

功能列表，说明实体做什么。

处理，描述实体为了实现其功能所采用的方法，它是对功能的细化

### 数据实体描述

按照以下格式描述模块内部使用的多个数据实体：

标识：

类型及组成结构（必要的话）：

目的：

### 依赖性描述

描述模块内部的依赖关系，如两个子模块的依赖关系等。

### 接口描述

描述模块内部的接口关系，如两个子模块间的接口等。

# 引擎jni 接口

描述模块中定义和使用的数据，包括：

简单数据，如模块级的全局变量、常量、宏；

复合数据，如模块内部的结构、联合...；

## 数据描述

在1中描述本模块中的简单变量、常量、宏；

从第2部分起描述本模块中的复合数据，如结构、联合等；

### 简单数据描述；

描述在本模块中定义和使用的简单变量、常量、宏；

按照下列格式进行简单数据的描述

功能描述：描述该数据的用途

数据定义：定义该数据

### 结构1

定义和说明该数据结构(包括联合);

按照下列格式进行数据结构的描述

数据结构描述：描述该数据结构的用途

数据结构定义：用实际的编程语言定义该数据结构

数据项描述：

|  |  |  |
| --- | --- | --- |
| 数据类型 | 数据项定义 | 数据项描述 |
|  |  |  |
|  |  |  |
|  |  |  |

### 结构2

#### 日志回调：

using LogFunction = void (\*)(int logTarget, int level, const std::string& tag, const std::string& msg);

// tag session id

// level 日志等级

结果回调：

using ChannelFunction = char\* (\*)(int callid, int arg1, int arg2, const std::string& arg3, const std::string& arg4);

// arg4 识别结果

// arg3 session id

// arg2 event channel type (支持 1,2)

// arg1 事件类型

// callid event channel type (支持 3,4,5)

// value should comply with com.huawei.hiai.asr.core.channelcall.ChannelCallConstants.ChannelId

enum ChannelType {

CHANNEL\_LOG = 1, // 日志打印

CHANNEL\_EVENT = 2, // 事件上报

CHANNEL\_SESSION\_EVENT = 3, // session级别事件上报

CHANNEL\_GET\_JAVA\_RESULT = 4, // 从java获取执行结果

CHANNEL\_ASYNC = 5, // 异步调用，通过异步返回结果

CHANNEL\_CLOSE = 5000, // internal use only, to detach thread

} ;

// eventId should comply with com.huawei.hiai.asr.core.channelcall.ChannelCallConstants.EventId

typedef enum {

INIT\_COMPLETE = 1,

SPEECH\_STARTED = 2,

RMS\_CHANGED = 3,

PARTIAL\_RESULT\_UPDATED = 4,

SPEECH\_STOPPED = 5,

LEXICON\_UPDATED = 6,

FINAL\_RESULT = 7,

RECOGNIZE\_COMPLETE = 8,

ERROR = 9,

FRONT\_END\_EXPIRED = 10,

REPORT\_VOLUME\_ENERGY = 11,

STOP\_INTERRUPT = 12,

ARTIAL\_RESULT\_MIDDLE = 13,

STREAM\_START = 14,

MUSIC\_START = 15,

MUSIC\_STOP = 16,

ERROR\_OUT = 500,

ERROR\_EMPTY\_WAV = ERROR\_OUT + 1,

} EventType;

## 函数描述

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_initAsr(JNIEnv\* env, jobject jobj, jstring jpath, jstring jparams)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| jpath | string | I | 配置文件路径 |
| jparams | string | I | 引擎参数(json串) |
| 【para 具体格式】  model\_json：配置文件 (复用原有格式) | | | |
| 【接口描述】  创建asr 解码器，并设置日志和结果回调接口 | | | |
| 【接口约束】  全局创建一次 | | | |
|  | | | |

### int32\_t InitASR(const string& path, const std::string& para)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| path | string | I | 资源文件路径 |
| para | string | I | 引擎参数(json串) |
| 【para 具体格式】  model\_json：配置文件 (复用原有格式)  *deviceNameList：*  *decoder\_type：解码器类型* | | | |
| 【接口描述】  创建asr 解码器 | | | |
| 【接口约束】  全局创建一次 | | | |

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_releaseAsr(JNIEnv\* env, jobject jobj)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| 【接口描述】  删除asr 解码器 | | | |
| 【接口约束】 | | | |
|  | | | |

### int32\_t RealseASR()

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| 【接口描述】  删除asr 解码器，释放所有ASR实例 | | | |
| 【接口约束】  全局创建一次 | | | |

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_startRecognition(JNIEnv\* env, jobject jobj, jstring taskId, jstring words, jstring jparams)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| taskId | string | I | 会话id |
| jparams | string | I | 会话参数(json串) |
| *words* | *string* | *I* | *联系人（暂时保留）* |
| 【para 具体格式】  context：热词相关（暂时保留）  audio\_type：音频类型（暂时保留，可以删掉只支持pcm）  language\_type:语种 *（asrcore和服务统一修改）*  soundDistance:远近场，默认far\_Field  domain:领域  pproc：后处理功能 | | | |
| 【接口描述】  创建一次识别任务 | | | |
| 【接口约束】  在识别之前进行创建 | | | |
|  | | | |

### int32\_t InitSession(const string& sessionId, const std::string& sessPara, const std::string& userWords)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| sessionId | string | I | 会话ID |
| sessPara | string | I | 会话参数(json串) |
| userWords | string | I | 联系人 |
| 【para 具体格式】  context：  *audio\_type：音频类型（可以删掉只支持pcm）*  *moa：最长音频*  *eng\_*language*\_type*:语种  soundDistance:远近场，默认far\_Field  *scenario\_type:*  *Interrupt：*  *sos:*  *eos:* | | | |
| 【接口描述】  创建一次识别任务 | | | |
| 【接口约束】  在识别之前进行创建 | | | |

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_importWords(JNIEnv\* env, jobject jobj, jstring type, jstring words, jstring version, jstring deviceName)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| type | string | I | 手机/大屏 |
| words | string | I | 热词内容 json格式 |
| version | string | I | *版本号 1.0* |
| deviceName | string | I |  |
| 【para 具体格式】 | | | |
| 【接口描述】  导入热词 | | | |
| 【接口约束】 | | | |
|  | | | |

### int32\_t ImportWordSession(const std::string& wordsChar, const std::string& type, uint32\_t weight, const std::string& deviceName)

异步热词（识别中）是否需要？

会话前热词

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| wordsChar | string | I | 热词 |
| type | string | I | 类型 |
| weight | string | I |  |
| deviceName |  |  | 设备 |
| 【para 具体格式】 | | | |
| 【接口描述】  导入热词（最终调sRecResourceAdd） | | | |
| 【接口约束】 | | | |

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_sendAudio(JNIEnv\* env, jobject jobj, jstring taskId, jbyteArray frame, jint streamLen)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| taskId | string | I | 会话id |
| frame | jbyteArray | I | 语音流 |
| streamLen | jint | I | 语音长度（字节） |
| 【para 具体格式】 | | | |
| 【接口描述】  云服务调用该接口将语音段送给ASR进行识别 | | | |
| 【接口约束】 | | | |
|  | | | |

### int32\_t SendAudio(const string& sessionId, char\* audioStream, uint32\_t length)

### 是否多次调用 允许多次调用

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| sessionId | string | I | 会话ID |
| audioStream | char\* | I | 语音流 |
| length | uint32\_t | I | 语音长度 |
| 【para 具体格式】 | | | |
| 【接口描述】  调用该接口将语音段送给ASR进行识别 | | | |
| 【接口约束】 | | | |

### JNIEXPORT jint JNICALL Java\_com\_huawei\_hiai\_asr\_core\_AsrJni\_finishRecognition(JNIEnv\* env, jobject jobj, jstring taskId)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| taskId | string | I | 会话id |
| 【para 具体格式】 | | | |
| 【接口描述】  云服务调用该接口结束本次识别 | | | |
| 【接口约束】 | | | |
|  | | | |

调用时间点确认

### int32\_t FinishSession(const string& sessionId)

audio write写音频end状态

引擎开始处理识别，直到结束，引擎抛完结果后，引擎才能调用stop接口

|  |  |  |  |
| --- | --- | --- | --- |
| 参数列表 | 类型 | I/O | 字段描述 |
| sessionId | string | I | 会话ID |
| 【para 具体格式】 | | | |
| 【接口描述】  结束本次会话的识别 | | | |
| 【接口约束】 | | | |

# 模块2详细设计

## 

# 非功能设计

## DFX设计

### 可靠性设计

### 可扩展性(模块化)隔离设计

### 可配置性设计

### 可维护性设计

### 可生产性设计

### 安全性设计

REPLACE ME:参照终端安全红线和安全TMG安全编码规范，本软件可能涉及的安全威胁，并按照规范给出相应的设计，包括但不限于：

禁止非法监听设计：

个人数据保护设计：

敏感数据保护设计：参见上面“个人数据保护设计”

应用软件安全设计：NA

Web应用安全设计：NA

软件完整性保护设计：NA

设备防攻击与防破解设计：NA

禁止未公开接口设计：NA

操作系统加固与防病毒设计：NA

### 兼容性设计

### 可测试性设计

为了满足可测试要求而进行的设计。包括相关日志，特定的测试账号的处理等。

## 性能设计

REPLACE ME:

确保各产品域/责任田，满足自身涉及到的开机内存，动态内存，CPU，广播，静态性能，组件动态性能，组件静态性能，竞品比拼-一生快，竞品比拼-Benchmark跑分，竞品比拼-核心场景等关键性能指标基线；

### 开机内存基线遵从性说明

### 动态内存基线遵从性说明

### CPU基线遵从性说明

### 广播基线遵从性说明

### 静态性能基线遵从性说明

### 组件动态性能基线遵从性说明

### 组件静态性能基线遵从性说明

### 竞品比拼-一生快基线遵从性说明

### 竞品比拼-核心场景基线遵从性说明

## 功耗设计

### 应用场景功耗基线遵从性说明

### 应用服务后台功耗基线遵从性说明

## 飞马架构设计

### 可移植设计（解耦、升级）

REPLACE ME:

（适用于模块级设计文档）描述清楚本模块对外部（华为扩展接口和其它应用）的依赖，确保移植时这些条件的具备；

对各种可能芯片平台和分辨率下的移植方案和步骤，在Google升级中移植方法做阐述，保证移植活动明确、高效。

需要明确特性（或子特性）是否具备二进制独立发布的能力，给出与所有相依赖的模块/组件间的清晰、完备的接口定义。

### ROM占用设计

REPLACE ME:

对ROM占用做预估和分解，各种分辨率下保证ROM占用最低。

## 全球化设计

REPLACE ME:

考虑全球化场景下的设计，包括对多种语言支持、特殊语言排版和截断、全球地区禁忌喜好约束、全球化通用需求的设计等。

## 可扩展性设计

REPLACE ME:

考虑长期潜在需求进行预留，比如：硬件升级带来的变化、当前未决定但后续可能采纳的需求点等等。保证系统容易扩展。

## UI一致性设计自检

REPLACE ME:

原则1：自检整个设计中使用的UI相关控件、布局和逻辑符合UI一致性规范和UI STYLE上的要求，符合UI的模块设计。

*原则2：所有UI布局必须增加横屏、竖屏及相互切换布局设计（*[*案例*](#_UI设计不足案例)*）*

*原则3：所有UI布局必须增加从右至左镜像布局设计（*[*案例*](#_UI设计不足案例)*）*

## 设计IPR自检

REPLACE ME:

对方案中可能使用到的业界开源、方案是否有专利问题进行自检。

*若涉及三方软件自研替代，需分析是否与原技术合作存在合规风险，特别是与涉及美国(包括但不限于美国公司、跨国公司在美国的分支机构、美国公民及美国绿卡持有者个人或其在任何国家开办的公司)的技术合作。*

# 性能指标

云侧：接口指标，数据容量(缓存，数据库性能)

端侧：启动速度，页面打开速度，帧率等

# 部署

对硬件，组网等方面的要求。

根据性能与全球化要求，对缓存服务器配置和数量、数据库的数量、组网方式等提出具体要求。