2019-3-5

# SDM(MRCP-SERVER)技术文档

# 目录

# 1. 名词解释

这里先将本文档涉及到的名词简称汇总如下。

- ASR

  语音识别技术，也称为自动语音识别(Automatic Speech Recognition)，简称 ASR，其目标是将人类语音中的词汇内容转换为可读的文字。

- TTS

  语音合成技术，也称为自动语音合成(Text To Speech)，简称 TTS，其目标是将文字转成对应的语音声音。

- NLU

  自然语言理解技术（Natural Language Understanding），简称 NLU，有的叫做自然语言处理(Natural Language Processing, NLP)， 这里认为这两者是同一个概念，即：研究如何让计算机读懂人类语言。

- IVR

  交互式语音应答技术(Interactive Voice Response)，简称 IVR，本文将呼叫中心(Call Center)统一概称为 IVR。一般来说，由 IVR 通过 SDM 服务(实现了 MRCP 协议)调用 ASR、TTS、NLU 能力。



- MRCP-SERVER

  语音对话管理服务(Speech Dialogue Managerment)，简称 SDM，也即是本文档所描述的服务，是 MRCP 协议的服务端实现，对外用以和各类呼叫平台(比如华为呼叫中心、avaya、freeswitch)进行对接，对内集成了 ASR、TTS、NLU 等各种能力，是和呼叫平台对接、提供语音核心能力的网关层。

# 2. 概述

## 2.1 背景

在传统的语音应用中，各集成商必须针对不同的 ASR/TTS 厂商提供的 API 接口进行专门的集成开发，不同 ASR/TTS 引擎的接口各不相同，从而导致了集成过程的复杂性和局限性。

针对这一局限性，由 Cisco、Nuance 等多家公司联合开发了一个新的网络协议，该协议由 IETF 作为 Internet 草案发布(draft-shanmugham-mrcp-07)。该协议为那些需要进行语音处理的客户端提供了一种通过网络来控制媒体处理资源(如 ASR、TTS 引擎等)的机制。

利用 MRCP 协议提供的标准接口，语音集成开发商们不必再针对特定的 ASR/TTS 进行开发，而只需关注统一的、标准 MRCP 接口。利用这个特性，甚至可以在同一个应用系统中集成不同厂商的 ASR/TTS 引擎。这为各种语音应用开发提供了更加灵活的选择，有效降低了业务开发周期和企业成本。采用 MRCP 协议后，独立软件商和应用开发商仅需面向 MRCP 接口撰写程序，而无需考虑不同语音厂商的语音引擎产品之间的差异，可以真正做到一次开发，多种环境下应用；任何支持 MRCP 标准的语音引擎都可以被无缝集成和调用。

## 2.2 MRCP 协议

MRCP 协议定义了控制媒体处理资源所必需的请求(Request)、应答(Response)和事件(Event)等消息。MRCP 协议为每一种资源定义了相应状态机，为每一个请求和服务器事件定义了所需的状态转换。MRCP 关注的焦点在于控制那些进行媒体流处理的资源(如 ASR、TTS)，以及如何与这些资源之间进行通讯。

MRCP 协议自身不能独立工作，MRCP 并不定义会话连接，不关心服务器与客户端是如何连接的，也就是说 MRCP 的消息通常是承载于其它协议之上，如 RTSP，SIP 等。比如通过 RTSP(Real Time Streaming Protocol, RFC2326)协议在客户端与服务器端之间建立会话连接，通过 RTP(Real Time Transport Protocol，RFC3550)协议连接传送给语音流数据给语音服务器端。简单来说，MRCP 是一种通讯协议，用于语音服务器向客户端提供各种语音服务(如语音识别和语音合成)。

基于 MRCP 的语音应用系统往往是采用 C/S 架构，由客户端发出媒体流处理请求，服务器端则利用媒体处理资源(ASR/TTS)来处理或生成语音流，并将相应的处理结果返回给客户端。

随着 MRCP 协议的不断推广与应用，各语音技术厂商在实践与部署过程中碰到了各种各样的问题。为此，IESG(The Internet Engineering Steering Group)于 2002 年特许成立了 Speechsc 工作组，专门负责起草更加完善高效的支持分布式语音资源处理的开放协议。在 Speechsc 工作组的努力下，改进后的 MRCPv2(draft-ietf-speechsc-mrcpv2-09)很快应运而生。与 MRCPv1 不同的是 MRCPv2 消息不再依赖 RTSP 协议，而是作为独立的消息进行传输，但

是它仍依赖于会话管理协议 SIP(Session Initiation Protocol, RFC3621)协议，用于客户端与服务器端之间建立控制会话。

目前有两个版本的 MRCP 协议：MRCPv1(RFC4463,2006)及 MRCPv2(RFC6787,2012)。我们建议，且主要支持 MRCPv2 版本，主要参考的是 RFC6787 协议。

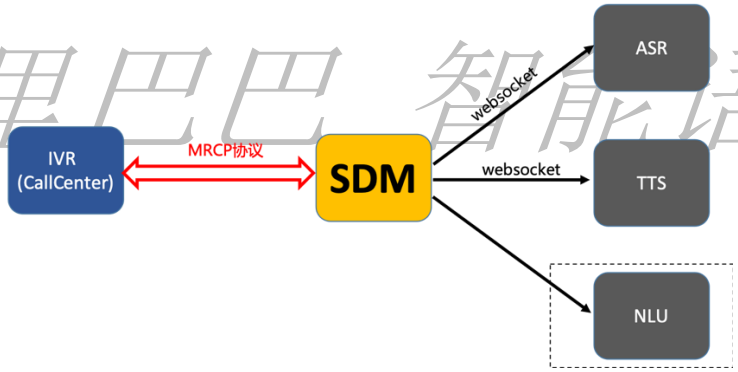
## 2.3 MRCP-SERVER

为顺应语音应用的快速发展，阿里巴巴智能语音团队开发了相应的 MRCP 服务器产品：MRCP-SERVER(简称 SDM)。该产品基于最新的国际通用开放标准 MRCPv2 而开发，为阿里巴巴智能语音服务产品家族提供了新的、标准化的服务接入接口。

该产品的推出，对智能语音应用的应用开发商、集成商、最终客户有着重要意义，为客户带了了更多的灵活性和可选择性，减少了客户接入语音服务的复杂度，降低了客户的开发成本和业务成本。通过 SDM，允许客户通过 MRCP 协议来调用阿里巴巴最新的语音识别(ASR)和语音合成(TTS)能力，而无需关注 ASR 和 TTS 具体细节，仅仅通过标准接口即可完整使用整个智能语音服务。为企业在多种实际应用场景下，赋予产品“能听、会说、懂你”式的智能人机交互体验，比如智能语音导航（热线电话）、机器人外呼（产品推销）、金牌话术（实时质检）。

下面给出一个呼叫中心智能语音导航可能的架构：






从图中可以看出：

1. IVR 只需要通过 MRCP 协议和 SDM 进行交互（具体还依赖 SIP、RTP）协议等，集成、对接更简单；
2. IVR 无需关注 TTS、ASR、NLU 的具体细节，甚至无需关注其各自的调用方式，核心服务透明化；

# 3. SDM 核心功能列表

本产品的功能特性汇总如下。

| | 功能点 | 说明 |
|---|---|---|
| 基础功能 | 支持标准 MRCPv2 协议 | |
| | 支持 G711 编码格式，支持 alaw/ulaw/pcm | |
| | 支持语音实时传输实时转写 | |
| | 支持语音打断 | |
| | 支持基于 SIP 协议的集群负载部署 | |
| 语音合成 | 支持发音人设置 | |
| | 支持合成音量大小调整 | 目前支持的范围是[0, 100] |
| | 支持合成语速快慢调整 | 目前支持的范围是[-500, 500]，建议为 0 |
| | 支持合成语调高低调整 | 目前支持的范围是[-500, 500]，建议为 0 |
| | 支持纯文本合成、ssml 格式文本 | 具体 ssml 规范可以参考阿里巴巴语音合成产品对应的 ssml 定义 |
| 语音识别 | 支持 VAD 语音断句间隔大小设置 | 目前支持的范围是[200, 2000] |
| | 支持阿里巴巴 ASR 定制语言模型设置 | |
| | 支持阿里巴巴 ASR(泛)热词模型设置 | |
| | 支持阿里巴巴 ASR 类热词模型设置 | |
| | 支持对 ASR 文本的规整、顺滑、加标点功能 | |
| | 支持长语音连续转写识别 | 扩展兼容实现 |

# 4. 语音合成(TTS)接口

## 4.1 MRCP 特性

### 4.1.1 MRCP 方法

| 方法 | 说明 | 是否支持 |
|---|---|---|
| SET-PARAMS | 设置 TTS 相关参数 | 支持 |
| GET-PARAMS | 获取参数对应的值 | 支持 |
| **SPEAK** | **合成请求，收到此请求即开始调用 TTS 引擎进行合成** | **支持** |
| **STOP** | **停止语音播放，即 SDM 不再返回语音流给 IVR** | **支持** |
| **BARGE-IN-OCCURRED** | **发生 barge-in 事件，SDM 收到此事件后需立刻停止语音传输并结束本次语音合成请求** | **支持** |
| PAUSE | 暂停语音传输 | 支持 |
| RESUME | 重启，重新进行语音传输 | 支持 |
| CONTROL | | 不支持 |
| DEFINE-LEXICON | | 不支持 |



### 4.1.2 MRCP 事件

| 事件 | 说明 | 是否支持 |
|---|---|---|
| SPEECH-MARKER | | 不支持 |
| SPEAK-COMPLETE | TTS 数据传输完成 | 支持 |

### 4.1.3 MRCP 消息头

| 参数 | 说明 | 是否支持 |
|---|---|---|
| Fetch Timeout | 服务器从网络上获取资源的超时时间 | 无需支持，已淘汰 |
| Content-Type | 待合成文本的格式；<br>text/plain 表示纯文本；<br>application/ssml+xml 表示为 xml 格式 | 支持 |

| | | |
|---|---|---|
| Kill-On-Barge-In | 设置在合成时是否支持打断，为 true 表示可打断，默认为 true | 支持 |
| Speaker-Profile | | 不支持 |
| Completion-Cause | Server 完成合成后返回的状态码 | 支持 |
| Completion-Reason | 状态码对应的原因 | 支持 |
| Voice-Parameter | 1.voice-gender 设置性别<br>2.voice-age 设置年龄；<br>3.voice-variant 设置音库；<br>4.voice-name 设置发音人名字； | 支持，**重点支持 voice-name 的使用** |
| Prosody-Parameters | 1.Prosody-Volume:设置音量，取值[0,100]，默认 50；<br>2.Prosody-Rate:设置 语速，取值[-500, 500]，默认 0； | 支持 |
| Speech-Marker | | 不支持 |
| Speech-Language | 设置合成的语言种类 | 无需支持，默认即可进行中英文合成 |

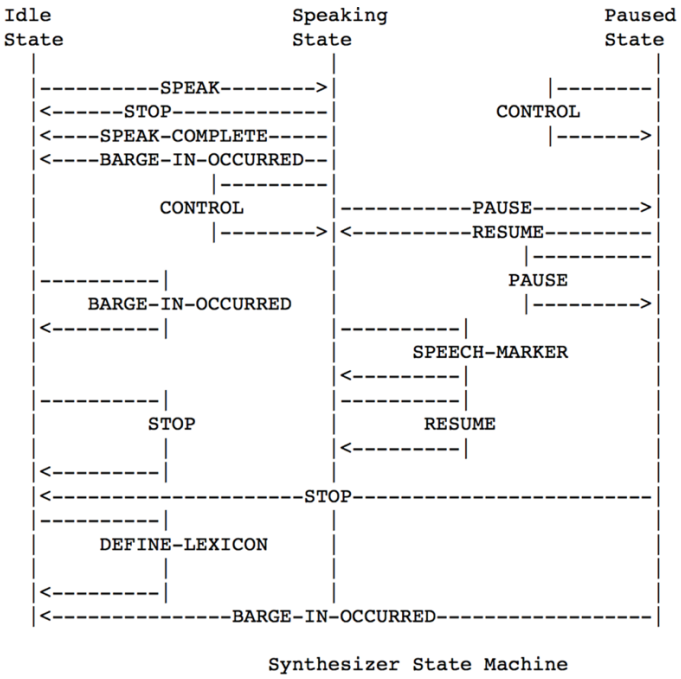


## 4.2 MRCP 返回码

| 返回码 | 返回码描述 | 说明 |
|---|---|---|
| **000** | **normal** | **SPEAK 请求正确完成** |
| **001** | **barge-in** | **SPEAK 请求由于 barge-in 被终止** |
| 002 | parse-failure | SPEAK 请求由于解析语音标识文本错误被终止 |
| 003 | uri-failure | SPEAK 请求由于 URI 错误被终止 |
| **004** | **error** | **合成出错，往往是调用 TTS 服务出错导致** |
| 005 | language-unsupported | 不支持该语言 |

注：表格中加粗的状态码是常见的、也是需要我们特别关注的几种情况。

## 4.3 MRCP 状态机




Synthesizer State Machine

## 4.4 MRCP 请求

一般 speak 请求如下所示：

```
MRCP/2.0 323 SPEAK 1
Channel-Identifier:ef116f480a1e4c5a@speechsynth
Content-Type: text/plain
Voice-Age: 28
Content-Length: 35

Hello，welcometo 阿里巴巴
```

当 server 端接收到 client 的 speak 请求时，会对此消息进行解析：

1. 如果解析失败，或者解析出的合成文本为空，会直接返回 client 一个带有错误码的完成事件：

```
MRCP/2.0118 1 407 COMPLETE
Channel-Identifier:60c15c7201e74f12@speechsynth
Completion-Cause:002 parse-failure
```

2. 如果解析成功，但是 TTS 引擎合成失败，也会返回带有错误码的完成事件：

```
MRCP/2.0110 1 407 COMPLETE
Channel-Identifier:b9fead3f7aec434c@speechsynth
Completion-Cause:004 error
```

注意：这里返回的 004 error 是标准 MRCP 协议错误码，实际错误原因还需要从 SDM 的日志中进行排查；

3. 解析成功且合成成功时，mrcpserver 会返回如下消息表示处理中：

```
MRCP/2.083 1 200 IN-PROGRESS
```

```
Channel-Identifier:b105fad3fe844960@speechsynth
```

4. 当合成语音全部传输传输到 client 端时，返回如下消息表示请求已完成：

```
MRCP/2.0122 SPEAK-COMPLETE 1 COMPLETE
Channel-Identifier:ef116f480a1e4c5a@speechsynth
Completion-Cause:000 normal
```

在上面 1,2 两个场景返回的状态码 407 表示该请求处理失败，具体错误码比如 002、004 的意义可以参考上面单独列举的错误码说明表格；如果处理成功，返回的状态码时 200。

# 5. 语音识别(ASR)接口

## 5.1 MRCP 特性

### 5.1.1 MRCP 方法

| 方法 | 说明 | 是否支持 |
|---|---|---|
| SET-PARAMS | 设置参数 | 支持 |
| GET-PARAMS | 获取参数对应的值 | 支持 |
| DEFINE-GRAMMAR | 合成方法 | 支持 |
| **RECOGNIZE** | **ASR 请求，收到此请求后 SDM 即开始处理语音流并调用 ASR 引擎** | **支持** |
| INTERPRET | NLU 请求 | 不支持 |
| GET-RESULT | 主动获取结果 | 不支持 |
| **START-INPUT-TIMERS** | **Client 向 SDM 通知开始做 noinput 计时，重要** | **支持** |
| **STOP** | **停止继续识别，SDM 不再进行语音处理** | **支持** |



### 5.1.2 MRCP 事件

| 事件 | 说明 | 是否支持 |
|---|---|---|
| START-OF-INPUT | 已检测到语音，IVR 接收到此信号后可以实现语音打断功能 | 支持 |
| RECOGNITION-COMPLETE | 完成 asr 识别请求，此时会返回 ASR 识别结果，并结束本次请求 | 支持 |
| INTERMEDIATE-RESULT | 完成某一句话的识别，此时会返回 ASR 识别结果，但会继续接受 IVR 的语音流和识别。<br>注意：该事件非标准 MRCP 协议定义，扩展实现以支持长语音连续识别(比如坐席质检) | 支持，非标准，扩展实现 |

### 5.1.3 MRCP 消息头

| 参数 | 说明 | 是否支持 |
|---|---|---|
| Confidence-Threshold | 置信度 | 无需支持，目前 asr 准确率较高，无需该参数控制 |
| Sensitivity-Level | 灵敏度 | 无需支持 |

| | | |
|---|---|---|
| N-Best-List-Length | N 个识别结果 | 不支持，目前 asr 准确率较高，只返回给 IVR 最优的一个结果 |
| **No-Input-Timeout** | **无话超时时间，重要** | 支持 |
| Recognition-Timeout | 识别超时时间 | 无需支持，历史原因参数 |
| Waveform-URI | 服务器保存下来的音频名称 | 支持 |
| **Completion-Cause** | **返回的状态码，重要** | 支持 |
| **Completion-Reason** | **状态码对应的原因，重要** | 支持 |
| **Vendor-Specific -Parameters** | **自定义参数，重要，很多业务参数均需通过此字段传递** | 支持 |
| **Start-Input-Timers** | **通知 server 启动无话超时计时，重要** | 支持 |
| **Speech-Complete-Timeout** | **说话完成时间，也即 vad 断句时间，重要** | 支持 |
| Speech-Incomplete-Timeout | | 不支持 |
| Save-Waveform | 是否保存语音到本地磁盘 | 支持 |
| Speech-Language | 语言种类设置 | 不支持 |
| Recognition-Mode | 设置识别模式，”mormal”或” | 支持“normal”、“voice”、“continuous” |



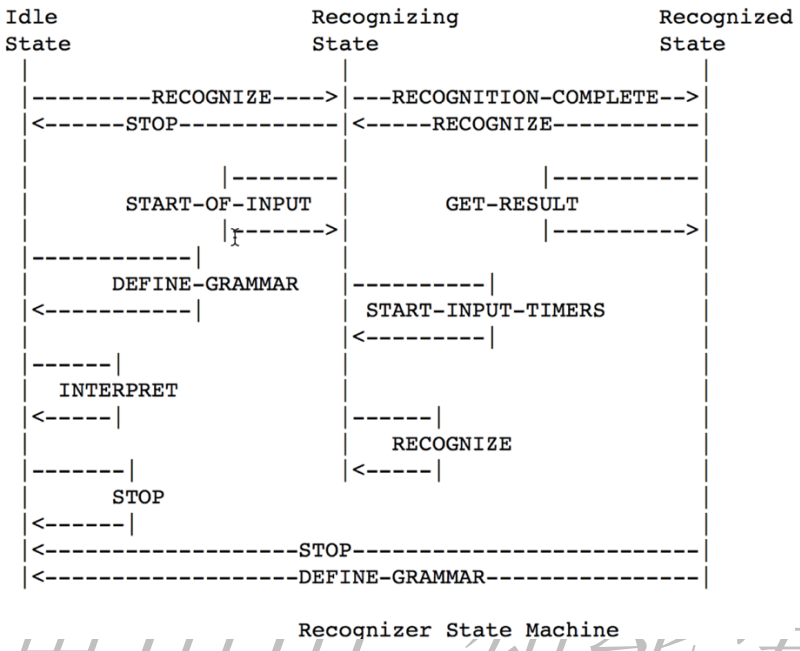
## 5.2 MRCP 返回码

| 返回码 | 返回码描述 | 说明 |
|---|---|---|
| **000** | **success** | **ASR 请求正确完成** |
| **001** | **no-match** | **请求完成，但没有真正的识别结果，多是因为噪音导致** |
| **002** | **no-intput-timeout** | **无话超时，在规定时间内没有检测到有效语音** |
| 003 | hotword-maxtime | hotword 模式下，在规定时间内没有有效结果 |
| 004 | grammar-load-failure | grammar 加载失败 |
| 005 | grammar-compilation-failure | grammar 编译失败 |
| **006** | **recognizer-error** | **识别失败，往往是调用 ASR 出错导致** |
| 007 | speech-too-early | |
| 008 | success-maxtime | 语音太长 |
| 009 | uri-failure | 无法访问该 uri |
| 010 | language-unsupported | 不支持该语言 |

| | | |
|---|---|---|
| 011 | cancelled | |

注：表格中加粗的状态码是常见的、也是需要我们特别关注的几种情况。

## 5.3 MRCP 状态机




Recognizer State Machine



## 5.4 MRCP 请求

一个 ASR 识别请求必须通过一个 RECOGNIZE 请求来触发，一个可能的完整 RECOGNIZE 请求如下所示：

```
MRCP/2.0 327 RECOGNIZE 2
Channel-Identifier: 31b743cfd7a44395@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Start-Input-Timers: true
No-Input-Timeout: 5000
Speech-Complete-Timeout: 4000
Vendor-Specific-Parameters: name111=param_value;name222=1111
Content-Length: 33

session:request1@form-level.store
```

相应的，每收到一个 RECOGNIZE 请求，则必然有且仅有一个 RECOGNITION-COMPLETE 事件返回给 IVR，或者返回多个 INTERMEDIATE-RESULT 事件(扩展实现，非标准协议)。SDM 返回给 IVR 的一个可能的识别结果格式如下所示：

```
MRCP/2.0550 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier:fbaebb6198694caf@speechrecog
Completion-Cause:000 success
```

```
Content-Type:application/x-nlsml
Content-Length:277

<?xmlversion="1.0" encoding="utf-8"?>
<result>
<interpretationconfidence="1">
<instance>
<result>帮我查一下明天的天气</result>
</instance>
<input mode="speech">帮我查一下明天的天气</input>
<asr>帮我查一下明天的天气|420|3485|ea76de0069e14380a3a8ad7a02ab72c7|utter-8kHz- fbaebb6198694caf-1.pcm</asr>
</interpretation>
</result>
```

在 RECOGNITION-COMPLETE 事件响应中，携带了 ASR 的识别结果内容(消息体中)，比如上述示例中的 xml 格式，，尤其需要关注<instance>和<input>标签中的内容，该响应中会包含识别结果，也可能返回有理解结果（**注意：如果需要返回理解结果，则需要在完成 ASR 请求之后由 SDM 调用 NLU 服务，该调用需要单独开发**)；以上述示例来说，识别结果是“帮我查一下明天的天气”。

如果触发了其他流程或者发生了错误，这里分为几种情况(错误情况下一般不返回消息体，重点关注 Completion-Cause 项)：

1. **无话超时**，即接通电话之后用户没有说话，一直静音，超过预定的时间后(比如 5s，可以调整)，SDM 会主动断开并通知给 IVR 应用，此时返回：

```
MRCP/2.0 226 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier:578a7d86c2b349a3@speechrecog
Completion-Cause:002 no-input-timeout
```

2. **识别失败**，即语音数据处理成功，但是由于某种原因导致识别失败(很少发生)，此时返回：

```
MRCP/2.0 226 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier:578a7d86c2b349a3@speechrecog
Completion-Cause:006 recognize-error
```

3. **识别成功，但是没有识别结果**，即结果为空(往往是由于噪音太大，或者用户说了话，但是基本没法听清楚说的什么内容，很少发生)，此时返回：

```
MRCP/2.0 207 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier:90b1915795304810@speechrecog
Completion-Cause:001 no-match
```

4. **识别成功，但是进行语义理解时失败**，即只有识别结果，没有理解结果，此时返回：

```
MRCP/2.0 574 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier:0f8b31a7ac6f4f2c@speechrecog
Completion-Cause:012 semantics-failure
Content-Type:application/x-nlsml
```

```
Content-Length:302

<?xmlversion="1.0" encoding="utf-8"?>
<result>
<interpretationgrammar="request1@form-level.store" confidence="1">
<instance>
<error-code>40011</error-code>
<error-message>tokenhas been expired</error-message>
</instance>
<input mode="speech">我要回家了，买不到火车票</input>
</interpretation>
</result>
```

注意这种情况下，是有识别结果返回的，只是没有正确的理解结果，相应位置上是理解请求的错误信息说明(错误信息依赖依实际场景实际项目，上述只为示例)。

## 5.5 Vendor-Specific-Parameters 参数

在标准协议规范中，除了标准参数之外，也允许用户设置自定义参数，比如传递手机号、工号等特定信息，实现上可以通过 Vendor-Specific-Parameters 消息头进行设置，该消息头在 SET-PARAMS、RECOGNIZE 中生效。



### 5.5.1 ASR 识别参数

为提升 ASR 识别效果，给客户提供更丰富的 ASR 控制能力，我们允许通过 MRCP 协议中的 Vendor-Specific-Parameters 参数进行传递 ASR 相关控制参数。

| 参数 | 意义 |
| --- | --- |
| SPEECH_CUSTOM_ID | 阿里巴巴 ASR 服务独有的定制模型优化手段，客户通过该字段可以调用相应的定制模型 id，依次提升 ASR 识别效果 |
| SPEECH_VOCAB_ID | 阿里巴巴 ASR 服务独有的(泛)热词优化手段，客户通过该字段可以调用相应的热词 id，以此提升 ASR 识别效果 |
| SPEECH_CLASS_VOCAB_ID | 阿里巴巴 ASR 服务独有的类热词优化手段，客户通过该字段可以调用相应的热词 id，以此提升 ASR 识别效果 |

示例用法：

```
MRCP/2.0 368 RECOGNIZE 2
Channel-Identifier: c6bfc0d4819c4977@speechrecog
Start-Input-Timers: true
Confidence-Threshold: 0.87
```

```
Vendor-Specific-
Parameters:SPEECH_CUSTOM_ID=b56d6ef94263429cab4354e824251a8a;SPEECH_VOCAB_ID=b56d6ef942634
29cab4354e824251a8a
Content-Length: 33
```

注意这个请求中 Vendor-Specific-Parameters 字段对应的设置。

### 5.5.2 业务自定义参数

在实际场景中，可能需要 IVR 传递用户电话号码等具体业务信息到 SDM，以便进行后续更多处理，此时也可以通过该消息头进行传递，具体字段仅需 IVR 和 SDM 双方协商即可，具体内容不限。示例用法：

```
Channel-Identifier:c6bfc0d4819c4977@speechrecog
No-Input-Timeout:5000
Recognition-Timeout:10000
Start-Input-Timers:true
Confidence-Threshold:0.87
Vendor-Specific-
Parameters:SPEECH_CUSTOM_ID=b56d6ef94263429cab4354e824251a8a;phone=13856789856;agent=100890;nam
e=xfb
Content-Length:33
```

注意这个请求中 Vendor-Specific-Parameters 字段对应的设置，既设置了 ASR 识别所需要的定制模型，也传递了 phone、agent 等字段。

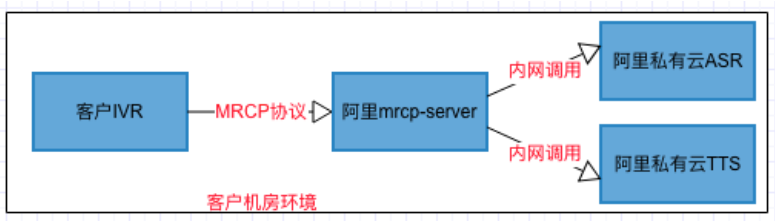
**需要说明的是，不同的 IVR 平台有不同的配置方式，具体需要参考相应 IVR 平台的文档或者接口介绍，设置后最终的请求格式均需和上文介绍的一致才能生效。**

# 6. 部署

机器配置： 4C 8G。操作系统要求：linux，建议 Centos 7 及以上。

## 6.1 全私有云部署

全私有云部署指的是所有服务（MRCP-SERVER、GATEWAY、ASR、TTS）都是私有化部署到本地。此时 MRCP-SERVER 做为阿里巴巴智能语音专有云的一个功能模块提供给用户使用，**阿里巴巴智能语音专有云服务的安装目录里提供了包括本应用在内的完整专有云部署方式。**本节仅做简单说明。




全私有云部署时的系统架构如上图示意，其中在部署启动的时候需要设置相应的后端地址(即语音应用 gateway 模块的地址)。在 service/conf/nls.conf 中有如下的默认设置：

```
# sdm config， 如果无 nls-cloud-sdm ，则无需配置
sdm_token=default
sdm_tts_url=ws://127.0.0.1:8101/ws/v1
sdm_tts_appkey=default
sdm_asr_url=ws://127.0.0.1:8101/ws/v1
sdm_asr_appkey=default
```

如果 MRCP-SERVER 和后端 gateway 应用在同一台机器上（比如单机测试或者 1:1 集群部署时），则保持上面设置不变即可，如果不在同一台机器或者 gateway 的监听端口发生变化，则需要设置上面 sdm_tts_url 和 sdm_asr_url 的 ip:port 为正确的 gateway 地址。

而如果在非 1:1 集群部署时，比如 N 个 MRCP-SERVER 实例，M 个 gateway 实例，则可以使用 vipserver 的方式进行设置：

```
app_tts_url=vip://default.gateway.vipserver/ws/v1
app_asr_url=vip://default.gateway.vipserver/ws/v1
```

注意此时的设置形式和上面不太一样:

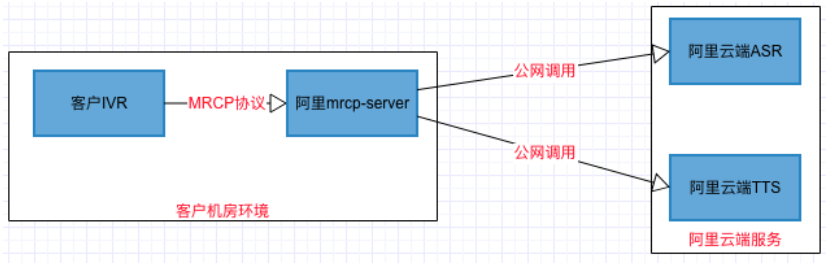
a. ws → vip;
b. ip:port → default.gateway.vipserver

其中 default.gateway.vipserver 即是 gateway 集群的 vip 域名。
需要说明的是通过 vipserver 方式访问的时候 需要在 apes 中配置 vipserver 已经对应的 gateway 集群，具体可以参考专有云部署文档。

至于 appkey 和 token 则一直保持不变。

## 6.2 私有云+公共云混合部署

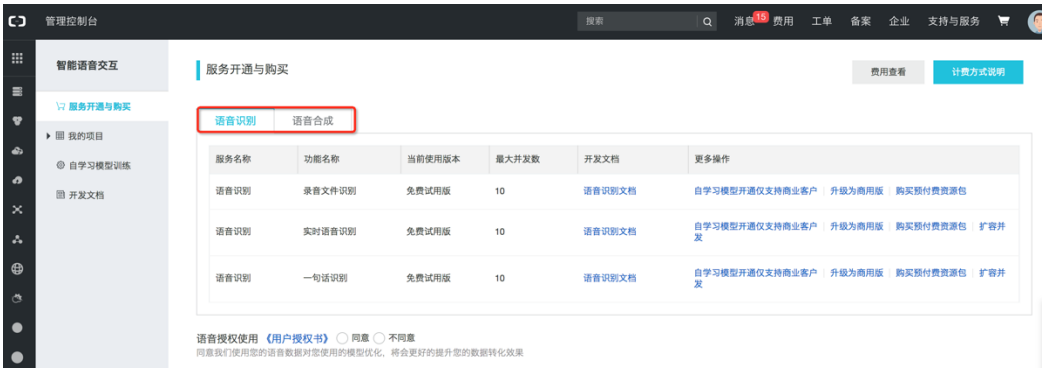
“私有云+公共云混合部署”指的是仅在客户环境部署本应用 MRCP-SERVER，而直接调用公共云上的 ASR 和 TTS 服务，系统架构如下示意：




该方案要求客户在阿里云网站开通智能语音服务。

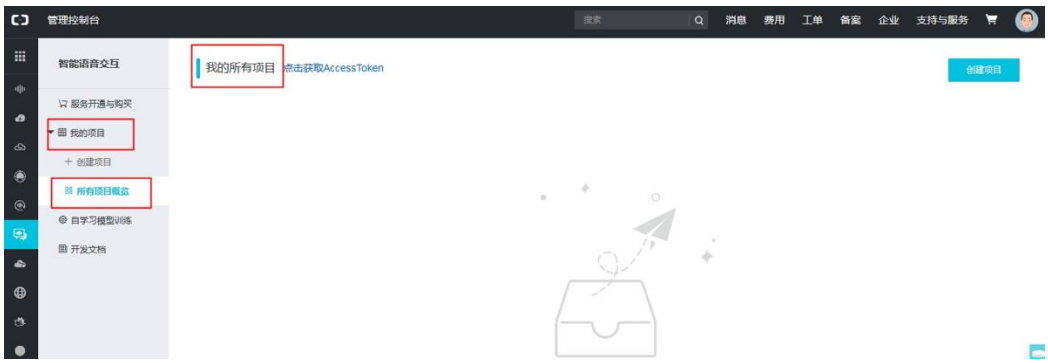
### 6.2.1 开通阿里云智能语音 ASR、TTS 服务

1. 注册阿里云账号；
2. 到“智能语音交互服务”页面，点击『立即开通』，开通 ASR/TTS 相关服务 https://data.aliyun.com/product/nls?spm=a2c4g.11186623.2.14.284627ac1yEDpa；
3. 在跳转后的页面，点击『立即购买』，购买语音服务。默认开通试用权限，目前公共云用户可免费使用不超过 10 路并发；

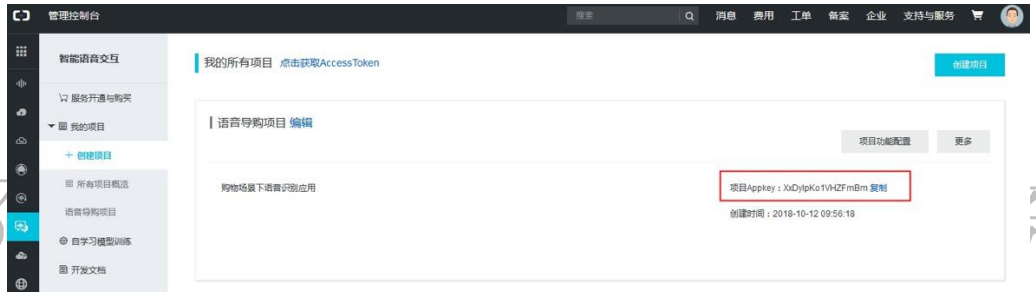



4. 在阿里云管控台 Access Key 管理页面创建并获取您的 AccessKey ID 和 AccessKey Secret。您可以使用它们调用智能语音服务 **这里的 Access Key ID、Secret 在配置 SDM 的时候会用到，请妥善保存。**

5. 登录到控制台，默认首页为”我的项目” -> “所有项目概览”，因此时还未创建项目，所以在右侧”我的所有项目”显示当前暂无项目，创建语音服务项目；




6. 点击界面上”创建项目”按钮，会弹出创建项目模态框，按照提示创建所需要的项目；然后可以在页面上看到新建的项目，**其中的“项目 Appkey”是 SDM 调用 ASR 服务需要配置的；**




7. 注意：一定要再次确认“项目功能配置”下该 appkey 对应的模型类型(一般是 8k 或 16k)，而该场景(使用本应用一般是电话场景)下务必使用 8k 模型。

## 6.2.2 配置 SDM(MRCP-SERVER)

SDM 主要使用的配置文件有三个

- ✓ conf/nlstoken.json
- ✓ conf/service-asr.json
- ✓ conf/service-tts.json

如果 SDM 是以二进制安装包的形式提供的话，则需要手动配置这三个文件：

1. nlstoken.json：

```
{
    "AccessKeyId": "第四步中获取的 id",
    "AccessKeySecret": "第四步中获取的 secret",
}
```

2. service-asr.json(主要配置 url、和 appkey 项)：

```
{
    "url": "wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1",
    "appkey": "创建 asr 项目的时候 获取到的 appkey",
```

```
}
```

3. service-tts.json(主要配置 url、和 appkey 项):

```
{
    "url": "wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1",
    "appkey": "创建 tts 项目的时候 获取到的 appkey",
}
```

如果 SDM 是以 docker 镜像的形式提供的话，则需要:

1. 拉取镜像

   ```
   sudo docker pull registry.cn-shanghai.aliyuncs.com/nls-cloud/sdm
   ```

2. 初始化环境

   ```
   sudo docker run -d --privileged --net=host --name nls-cloud-sdm    -v
   `pwd`/logs:/home/admin/logs    -v `pwd`/data:/home/admin/disk registry.cn-
   shanghai.aliyuncs.com/nls-cloud/sdm:latest standalone
   ```

   该操作第一次执行时只是用来初始化环境，比如将容器中中 SDM 的目录挂载到宿主机上(比如容器的 logs 目录挂载到宿主机当前目录的 logs 下，容器中的/home/admin/disk 挂载到宿主机当前目录的 data 下)，由于相关参数配置没有修改，所以容器最终不会运行起来，需要进行以下配置操作;

3. 配置参数

   此时配置文件在宿主机 data/nls-clou-sdm/conf 下，同上面介绍的方式一样，修改其中的三个配置文件 nlstoken.json、service-asr.json、service-tts.json，修改方式和字段均与上节的介绍一致。

4. 重新启动（在宿主机上执行）

   ```
   sudo docker start nls-cloud-sdm    # 重新启动
   sudo docker ps                     # 查看是否启动成功
   ps -ef | grep alimrcp-server       # 查看进程是否拉起
   ```

5. 查看日志输出，检查是否有 ERROR 信息

   ```
   cat logs/nls-cloud-sdm/alimrcp-server.log
   ```

6. 测试

   SDM 自带了测试工具，可以验证以上部署是否成功，需要进入到容器中进行测试。

```
sudo docker exec -it nls-cloud-sdm /bin/bash    # 进入到容器
cd /home/admin/nls-cloud-sdm/bin
export LD_LIBRARY_PATH=../lib/:$LD_LIBRARY_PATH
./alimrcp-client -a       #执行测试工具，测试 ASR 链路，观察日志是否有 ERROR 信息，是否有识别结果
./alimrcp-client -t       #执行测试工具，测试 TTS 链路，观察日志是否有 ERROR 信息
```

# 7. 集成对接

## 7.1 快速入门

### 7.1.1 什么是 MRCP 协议，用来做什么

呼叫中心供应商有很多，语音厂商也有很多，在从人工坐席转向智能语音问答的过程中，可以说 IVR 和语音厂商是 N：N 的关系，有各自的接口、API、SDK，很难进行对接，客户受限于现有的 IVR 实施商而不能自主选择语音厂商，语音厂商受限于无法快速和所有 IVR 厂商进行对接而不能输出给最终客户；

在此基础上，多家 IVR 厂商和语音厂商共用拟定了一个标准协议来解决这个问题，这个协议经过演化，最终形成一个标准 RFC6787 协议(MRCPv2，点此查看完整协议规范)， 但是注意还有很多客户还在使用很老版本的 IVR 设备，未必支持该协议。

MRCP：Media Resource Control Protocol，媒体资源控制协议。该标准即是为了解决双方协议有差异的问题，双方(IVR 厂商和语音厂商)可以根据该协议进行各自开发，无需相互依赖。

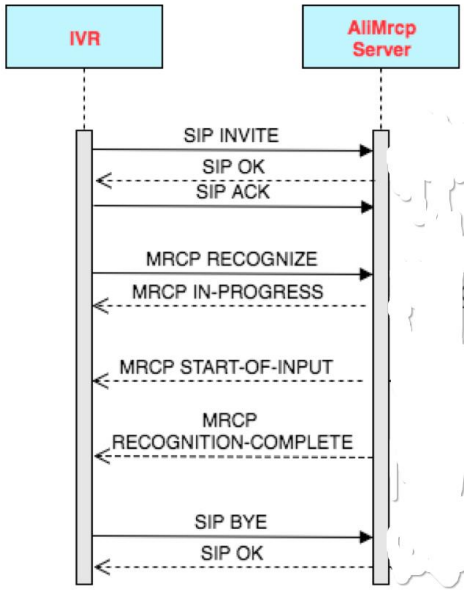
语音团队目前也已支持了 MRCP 协议，对应的服务即是 MRCP-SERVER，也简称为 SDM（speech dialogue managerment）。该服务实现了 MRCP 协议的服务端，同时对应的 IVR 也就是 MRCP 客户端，由 IVR 主动调用 MRCP-SERVER 来使用 ASR、TTS，甚至 NLU 能力。

### 7.1.2 和 IVR(呼叫中心)对接时需要准备什么

对接时，我们作为被调用方，会提供 MRCP-SERVER 的 ip 和端口，除 ip：port 之外，无需其他信息，比如 ASR、TTS 相关信息无需提供给 IVR； 和 IVR 或者呼叫中心对接时，首先需要参考标准 MRCPv2 协议: RFC6787，对接前假设双方均已阅读过该协议，本文档不会对该协议过多描述；

1. 除非特定 IVR 厂商有特殊要求，否则和 IVR 的对接均使用 MRCPv2 协议，需双方熟悉 MRCP 协议；
2. 作为被调用方，需要 IVR 侧配置 MRCP-SERVER 的 ip 和 port，除此之外不需要再额外提供其他信息；MRCP-SERVER 侧不配置任何 IVR 应用信息；
3. 有的 IVR 设备可能要求提供语法文件，这是较早时候的做法，目前我们的实现里并不需要，如果强制要求，可以随意提供一个文件，内容随意(有的 IVR 可能要求 XML 格式，也不重要，随便写点什么就行)；

4. 下面是一个简单的协议交互图，可以作为参考：




## 7.1.3 SDM 使用的端口及协议都有哪些

默认情况下：

- 7010：SIP 协议端口，使用 TCP 和 UDP 协议(大多是 UDP)；
- 1544、1554：MRCP 协议端口，TCP 协议；
- 10000-20000 之间所有端口：RTP 协议端口，用来传输语音流，UDP 协议；(端口范围可以设置缩小)；
- 5070：如果使用我们的 SDM 负载方案，默认走此端口，TCP 和 UDP 协议；



对 IVR 来说，单机部署时只需提供 mrcpserver 的 ip 和 7010 端口即可，负载部署时提供 mrcpserver 负载服务器 ip 和 5070 端口；另外，以上端口均需和 IVR 保持网络互通。

如果需要修改默认端口，均需在 conf/alimrcp-server.xml 中进行设置：

```
<sip-port>7010</sip-port>
<mrcp-port>1544</mrcp-port>
<rtsp-port>1554</rtsp-port>
<rtp-port-min>10000</rtp-port-min>
<rtp-port-max>20000</rtp-port-max>
```

## 7.1.4 支持哪些语音格式

alaw(8k, 8bit)、ulaw(8k, 8bit)、pcm(8k, 16bit)。

### 7.1.5 返回的 ASR 结果格式是什么样的

一个可能的格式如下所示：

```
MRCP/2.0 654 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: c950e0cbd45a4917@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-c950e0cbd45a4917-2.pcm
Content-Type: application/nlsml+xml
Content-Length: 418

<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="session:request1@form-level.store" confidence="1">
<instance>
<result>明天北京的天气怎么样？ </result>
</instance>
<input mode="speech">明天北京的天气怎么样？ </input>
<asr>明天北京的天气怎么样？ |420|3495|a46ececdff384fa0b5b6edc54d940f0b|utter-8kHz-c950e0cbd45a4917-2.pcm</asr>
</interpretation>
</result>
```

说明：



- 消息头中的字段均为标准协议中定义的 header，具体可参考标准协议的介绍；
- 消息体的格式为 xml，具体内容在不同场景和需求下可能有一定差别(主要是 instance 的 tag 字段，其他 tag 下的格式基本保持一致)；
- 消息体中的有一个自定义的 tag: <asr>， 这是我们扩充的一个字段，该 tag 下的内容格式是：asr 识别结果|相对于 recognize 请求的用户有效语音起点时间(单位毫秒)|相对于 recognize 请求的用户有效语音尾点时间(单位毫秒)|后端 ASR 服务的 task_id(可以据此去查询 ASR 链路的请求日志|本次识别请求保存下来的对应录音文件名称(var/目录下)。

### 7.1.6 为什么有的 IVR 需要语法文件

虽然协议是标准的，但各家实现机制不同，语法文件目前并不必须，我们也没有对外的语法文件，如果 IVR 侧必须使用，可以随意写一个示例即可，比如 grammar.xml：

```
<?xml version="1.0"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US" version="1.0" mode="voice" root="digit">
  <rule id="digit">
    <one-of>
```

```
        <item>one</item>
        <item>two</item>
      </one-of>
    </rule>
</grammar>
```

### 7.1.7 如何调整无话超时参数

参数名：No-Input-Timeout，具体可参考标准协议：https://tools.ietf.org/html/rfc6787#section-9.4.6，该参数可能出现在 Set-Param、RECOGNIZE 等请求中。示例：

```
MRCP/2.0 382 RECOGNIZE 2
Channel-Identifier: 13d67277775c4be4@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
No-Input-Timeout: 5000
Start-Input-Timers: false
Confidence-Threshold: 0.87
Content-Length: 33
```

以上表示无话超时时间是 5000 ms，也就是如果开启 Start-Input-Timers 计时，而用户 5 秒内没有说话，则触发无话超时提醒，此时会返回给 IVR 一个 No-Input-Timeout 事件，且不再进行后续语音的识别处理。



### 7.1.8 no-match 和 no-input-timeout 是什么意思

SDM 返回给 IVR 的 ASR 请求状态时，一般常见的有(更多状态可以查看 RFC6787)：

| 返回码 | 错误描述 | 说明 |
| --- | --- | --- |
| 000 | success | ASR 请求正确完成 |
| 001 | no-match | 请求完成，但没有真正的识别结果，多是因为噪音导致 |
| 002 | no-intput-timeout | 无话超时，在规定时间内没有检测到有效语音 |

其中：

1. no-mtach 表示检测到了用户有效语音，但 ASR 无法识别出有效文字，这种情况一般是用户声音过低、语音全是噪音、用户背景噪音太大等情况导致的，这个可以通过试听 SDM 保存下来的录音文件来分析；该状态是由 vad 判断、ASR 识别决定的，目前无参数可调；

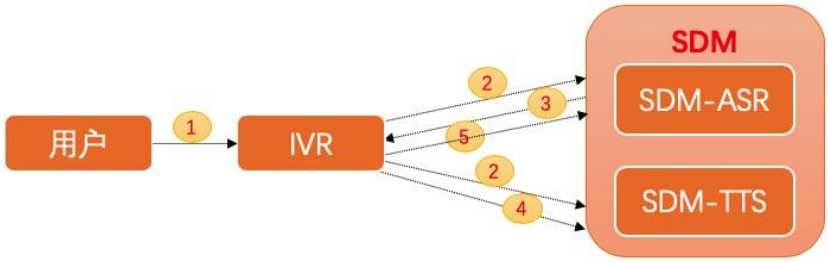
2. no-input-timeout 表示用户长时间不说话，而返回给 ivr 的事件；该状态受 IVR 发送的 No-Input-Timeout 参数和 SDM 本地 no-input-timeout 参数控制，以 IVR 发送的为主，本地默认值 5000ms，即 5 秒钟；可随时、动态调整；

3. 这两种情况均需要 IVR 或者 NLP(如果有的话)参与处理，比如给用户一个兜底提示等等；

### 7.1.9 如何实现语音打断

语音打断(barge-in)能使用户可以在自助语音服务的提示语播放过程中随时说出自己的需求，而无需等待播放结束，系统能够自动进行判断，立即停止提示语的播放，对用户的语音指示做出响应。该功能使人机交互更加高效、快捷、自然，有助于增强客户体验。

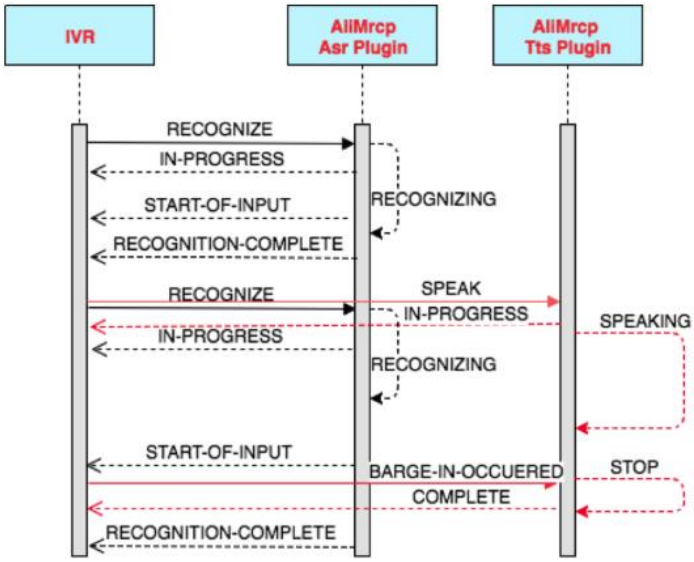
语音打断功能需要 IVR 和 MRCP-SERVER 相互配合完成，IVR 在这里发挥关键作用。我侧 MRCP-SERVER 和 ASR 已支持该功能，理论上大多数 IVR 自身也应支持该功能。一个常见的流程如下所示：




流程描述如下：

1. 用户拨打电话进入语音自助问答系统；
2. IVR 进行通过 SDM TTS 播放的同时，向 SDM 发起 ASR 请求；
3. 依赖 VAD 检测，SDM 判定用户语音为实际有效语音，并返回给 IVR 特定事件(START-OF-INPUT)；
4. IVR 接收到事件之后，立刻停止向用户播放 TTS，发送请求到 SDM (STOP 请求)；
5. SDM-ASR 模块继续将 IVR 发送过来的用户语音发送到 ASR 引擎进行识别，等待完整识别结果返回；

相应的协议流程如下所示：




流程描述如下：

1. IVR 发起 RECOGNIZE 请求 ASR 识别；
2. ASR 识别结果后返回结果给 IVR；
3. IVR 根据 ASR 的返回结果进一步处理，发起 SPEAK 请求，调用 TTS 播放；
4. 在第 3 步的同时，IVR 再次发起 RECOGNIZE 请求 ASR 识别；
5. ASR 在检测到用户有效语音时返回 IVR 一个 START-OF-INPUT 事件；
6. IVR 收到 START-OF-INPUT 事件后，立刻向 MRCP-SERVER 发送 BARGE-IN-OCCUERED 消息或者 STOP 消息；
7. MRCP-SERVER 收到消息后立刻结束语音播放(即不再传输剩余语音给 IVR)；
8. IVR 继续采集用户语音进行语音检测和识别；

## 7.2 和 ASR 服务的联动

在通过 MRCP 协议调用 ASR 能力时，MRCP-SERVER 集成、支持了 ASR 自身的各种功能接口。

### 7.2.1 ASR 相关参数配置

MRCP-SERVER 支持的 ASR 相关参数大多均在 `conf/service-asr.json` 文件中：

```
{
    "url": "所调用的 asr 服务地址",
    "appkey": "所使用的 appkey",
    "format": "pcm",
    "sample_rate": 8000,
    "enable_intermediate_result": 1,
    "enable_punctuation_prediction": 1,
```

```
    "enable_inverse_text_normalization": 1,
    "enable_semantic_sentence_detection": 0,
    "enable_ignore_sentence_timeout": 1,
    "desc": {
        "notice": "此处仅仅是描述信息，无需修改"
    }
}
```

相关参数均在配置文件中的 desc 字段中进行介绍，desc 对应的 json 内容无需修改，仅为描述使用。

### 7.2.2 VAD 断句间隔时长(静默时长)

VAD 静默时长通俗点讲就是当用户说完一句话后，静默多久判断用户已经说完话了。通过 mrcpserver 调用 ASR 时，默认配置是 800ms，也即：当用户说话之后静默 800ms 之内没有说话，则表示用户说完一句话了，此时会将完整识别结果返回给 ivr。该参数有两种调整方式：

1. conf/alimrcp-server.xml， 搜索：speech-complete-timeout，默认设置的是 800ms，可以调整，比如 500，该修改是全局生效，需重启 mrcpserver 服务；；
2. ivr 发起请求时设置 Speech-Complete-Timeout 参数，动态调整，如果设置该值，会替换掉本地的默认配置，该方式是仅在本次 RECOGNIZE 请求中生效；

```
MRCP/2.0 364 RECOGNIZE 2
Channel-Identifier: ab1f55efbd8c4be4@speechrecog
Speech-Complete-Timeout: 500
Content-Length: 33
```

需要说明的是：

1. 该值越小，会更快断句(判断用户一句话是否结束)，相应的返回也会更快，但带来的问题就是有的用户说话本身比较慢、停顿多，会导致提前判定该用户说完话了；
2. 该值越大，则可以避免断句太快的问题，但带来的问题就是返回时间会稍有增加。比较建议的值是 400-1200ms 之间，默认是 800ms，服务端支持的范围是[200, 2000]。
3. 如果用户 IVR 传递该参数时设置的范围超出[200, 2000]，则 SDM 不会使用客户传递的参数，而改为使用本地默认值(800)。

IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.2.3 ASR (泛)热词

如果需要使用 ASR 的定制(泛)热词功能，这里和通过 SDK 调用方式是一致的，即需要传递 ASR 定制热词的 id 信息，有两种方式：

1. 在配置文件 conf/service-asr.json 中有新增加参数："vocabulary_id":"热词 id"，模型 id 是已经训练/生效的热词 id，修改后注意该 json 配置文件格式的合法性；该修改是全局生效，需重启 mrcpserver 服务；
2. 通过 ivr 在每次调用服务时动态传递，比如在发送 RECOGNIZE 请求时：

```
MRCP/2.0 364 RECOGNIZE 2
Channel-Identifier: ab1f55efbd8c4be4@speechrecog
Vendor-Specific-Parameters: SPEECH_VOCAB_ID=定制语言模型 id
Content-Length: 33
```

注意里的 SPEECH_VOCAB_ID 字段，这是在 MRCP 协议扩展字段 Vendor-Specific-Parameters 中增加自定义字段实现的。该方式是仅在本次 Recognize 请求中生效。
IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。



### 7.2.4 ASR 类热词

同上面介绍的 ASR 定制(泛)热词功能，唯一不同的在于是配置文件中不是"vocabulary_id":"热词 id"，而是"class_vocabulary_id":"类热词 id"，而如果是 ivr 传递的话，不使用 SPEECH_VOCAB_ID，而是 SPEECH_CLASS_VOCAB_ID 字段。
IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.2.5 ASR 定制语言模型

如果需要使用 ASR 的定制语言模型功能，这里和通过 SDK 调用方式是一致的，即需要传递 ASR 定制语言模型的 id 信息，有两种方式：

1. 在配置文件 conf/service-asr.json 中有新增加参数："customization_id":"设置定制语言模型 id"，模型 id 是已经训练/生效的定制语言模型 id，修改后注意该 json 配置文件格式的合法性；该修改是全局生效，需重启 SDM 服务；
2. 通过 ivr 在每次调用服务时动态传递，比如在发送 RECOGNIZE 请求时：

```
MRCP/2.0 364 RECOGNIZE 2
Channel-Identifier: ab1f55efbd8c4be4@speechrecog
```

```
Vendor-Specific-Parameters: SPEECH_CUSTOM_ID=定制语言模型 id
Content-Length: 33
```

注意里的 SPEECH_CUSTOM_ID 字段，这是在 MRCP 协议扩展字段 Vendor-Specific-Parameters 中增加自定义字段实现的。该方式是仅在本地 Recognize 请求中生效。
IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.2.6 是否需要对 ASR 文本加标点

在配置文件 conf/service-asr.json 中有参数：enable_punctuation_prediction，默认为 1，即会将 ASR 识别结果加上标点。
如不需要，将该值置为 0，需重启 mrcpserver 服务。

### 7.2.7 是否需要对 ASR 文本进行规整/顺滑

在配置文件 conf/service-asr.json 中有参数：enable_inverse_text_normalization，默认为 1，即会将 ASR 识别结果进行规整/顺滑。比如将“一二三四五”规整为“12456”，将“百分之二十”规整为“20%”。 如不需要，将该值置为 0，需重启 mrcpserver 服务。



## 7.3 和 TTS 服务的联动

mrcpserver 支持的 TTS 相关参数大多均在 conf/service-tts.json 文件中：

```
{
    "url": "所调用的 asr 服务地址",
    "appkey": "所使用的 appkey",
    "format": "pcm",
    "sample_rate": 8000,
    "voice": "xiaoyun",
    "volume": 50,
    "speech_rate": 0,
    "pitch_rate": 0,
    "method": 0,
    "desc": {
        "notice": "此处仅仅是描述信息，无需修改"
        "format": "编码格式，支持 pcm, wav, mp3，保持 pcm 不变",
        "sample_rate": "采样率, 8000 或 16000，保持 8000 不变",
        "volume": "音量，范围是 0~100，可选参数，默认 50",
        "voice": "发音人，支持 xiaoyun(女), xiaogang(男)等",
        "speech_rate": "语速，范围是-500~500，可选参数，默认是 0",
        "pitch_rate": "语调，范围是-500~500，可选参数，默认是 0",
```

```
        "method": "合成方法，可选参数，默认是 0. 参数含义 0:不带录音的参数合成; 1:带录音的拼接合成;
2:不带录音的拼接合成; 3:带录音的参数合成",
        "notice": "此处仅仅是描述信息，无需修改"
    }
}
```

相关参数均在配置文件中的 desc 字段中进行介绍，desc 对应的 json 内容无需修改，仅为描述使用。

### 7.3.1 TTS 发音人设置

如果 TTS 引擎使用了新的发音人资源，则有两种方式可以调整：

1. 在配置文件 conf/service-tts.json 中修改参数：`"voice":"新的发音人名称"`，新的发音人名称是 TTS 引擎支持的发音人；该修改是全局生效，需重启 mrcpserver 服务；
2. 通过 IVR 在每次调用服务时动态传递，比如在发送 SPEAK 请求时(只在当前请求下生效)：

```
MRCP/2.0 285 SPEAK 1
Channel-Identifier: 641cc1f9928f4871@speechsynth
Voice-Name: siqi
Content-Length: 13

你好你好
```

注意里的 Voice-Name 字段，对应的 siqi 是 TTS 支持的一个发音人名称。该方式是仅在本次 Speak 请求中生效，如果 IVR 传递了参数，则不使用本地默认配置参数。

如何确认有哪些支持的发音人？这里有几个办法：

1. 如果调用的是公共云 TTS 服务，则需要在公共云官方文档上确认：语音合成；
2. 如果调用的是专有云 TTS 服务，则可以：
    1. 在 TTS 所在机器上执行：`curl 127.0.0.1:7701/voices`，列举当前 TTS 服务所支持的发音人列表；
    2. 通过 docker 命令进入到 TTS 服务的容器里，ls `/home/admin/nls-tts/data/default/voices/`，列举当前 TTS 服务所支持的发音人列表；

IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.3.2 TTS 音量调整

TTS 音量取值范围可以查询 TTS 服务所支持的音量范围，目前支持的范围是[0~100]，默认 50，值越大音量越高。

1. 在配置文件 conf/service-tts.json 中修改参数：  "volume":"新的音量大小"，，该修改是全局生效，需重启 MRCP-SERVER 服务；
2. 通过 IVR 在每次调用服务时动态传递 Prosody-Volume 参数，取值范围保持一致，比如在发送 SPEAK 请求时(只在当前请求下生效)：

```
MRCP/2.0 465 SPEAK 1
Channel-Identifier: a1379fd0af394070@speechsynth
Prosody-Volume: 80.0

欢迎使用智能语音交互产品
```

IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.3.3 TTS 语速调整

TTS 语速取值范围可以查询 TTS 服务所支持的语速范围，目前支持的范围是[-500, 500]，默认 0，值越大语速越快。

1. 在配置文件 conf/service-tts.json 中修改参数：  "speech_rate":"新的语速大小"，取值范围可以查询 TTS 服务所支持的语速范围，该修改是全局生效，需重启 MRCP-SERVER 服务；
2. 通过 IVR 在每次调用服务时动态传递 Prosody-Rate 参数，取值范围保持一致，比如在发送 SPEAK 请求时(只在当前请求下生效)：

```
MRCP/2.0 465 SPEAK 1
Channel-Identifier: a1379fd0af394070@speechsynth
Prosody-Rate: 50.0

欢迎使用智能语音交互产品
```

IVR 传递的参数优先级高于本地配置，若 IVR 传递了该参数，以 IVR 传递的参数作为本次调用时实际使用的参数。

### 7.3.4 TTS 语调调整

TTS 语调取值范围可以查询 TTS 服务所支持的语调范围，目前支持的范围是[-500, 500]，默认 0，值越大语调越高。

1. 在配置文件 conf/service-tts.json 中修改参数：  "pitch_rate":"新的语调大小"，取值范围可以查询 TTS 服务所支持的语调范围，该修改是全局生效，需重启 MRCP-SERVER 服务；

注意：目前不支持通过 IVR 动态调整该参数。

# 8. 常见示例流程

说明：

1. 下文的 C 表示 Mrcp Client，也即 IVR；S 表示 Mrcp Server，也即本文档中介绍的 SDM 服务；
2. C->S 表示 Client 发送请求到 Server；
3. S->C 表示 Server 返回响应给 Client；

## 8.1 TTS

### 8.1.1 IVR 发送纯文本

C->S:

```
MRCP/2.0 156 SPEAK 1
Channel-Identifier: 4d21421a7cde11e7@speechsynth
Content-Type: text/plain
Voice-Age: 28
Content-Length: 21
```

S->C:

```
MRCP/2.0 83 1 200 IN-PROGRESS
Channel-Identifier: 4d21421a7cde11e7@speechsynth
```

S->C:

```
MRCP/2.0 122 SPEAK-COMPLETE 1COMPLETE
Channel-Identifier:4d21421a7cde11e7@speechsynth
Completion-Cause: 000 normal
```

### 8.1.2 IVR 发送带 SSML 标签的文本

C->S:

```
MRCP/2.0 156SPEAK 1
Channel-Identifier:4d21421a7cde11e7@speechsynth
Content-Type:text/plain
Voice-Age: 28
Content-Length:21

<speak> 我可以按您的要求停顿<break time="2s"/>我可以按您的要求分词，南京<w>市长</w>江大桥。我可以按您的要求发音，<phoneme alphabet="py" ph="dian3 dang4 hang2">典当行</phoneme>。</speak>
```

S->C:

和上一节的返回规范一致。

## 8.2 ASR

### 8.2.1 先发送 Define-Grammar 再发送 Recognize

C->S:

```
MRCP/2.0 448 DEFINE-GRAMMAR 1
Channel-Identifier:f3d611ee7cde11e7@speechrecog
Content-Type:application/srgs+xml
Content-Id:request1@form-level.store
Content-Length: 269

<?xmlversion="1.0"?>
<grammarxmlns="http://www.w3.org/2001/06/grammar"xml:lang="en-US" version="1.0"
mode="voice"root="digit">
<rule id="digit">
      <one-of>
         <item>one</item>
         <item>two</item>
         <item>three</item>
      </one-of>
</rule>
</grammar>
```

S->C:

```
MRCP/2.0 112 1 200 COMPLETE
Channel-Identifier:f3d611ee7cde11e7@speechrecog
Completion-Cause: 000 success
```

C->S(收到 RECOGNIZE 请求时才会开始处理语音和调用 ASR 识别):

```
MRCP/2.0 290 RECOGNIZE 2
Channel-Identifier:f3d611ee7cde11e7@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
No-Input-Timeout: 5000
Recognition-Timeout: 10000
Start-Input-Timers: true
Confidence-Threshold: 0.87
Content-Length: 33

session:request1@form-level.store
```

S->C:

```
MRCP/2.0 83 2 200 IN-PROGRESS
Channel-Identifier:f3d611ee7cde11e7@speechrecog
```

S->C(表示检测到了用户语音):

```
MRCP/2.0 94 START-OF-INPUT 2IN-PROGRESS
Channel-Identifier:f3d611ee7cde11e7@speechrecog
```

S->C:

```
MRCP/2.0 654 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: b993a5d433e44dd3@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-b993a5d433e44dd3-1.pcm
Content-Type: application/nlsml+xml
Content-Length: 418

<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="session:request1@form-level.store" confidence="1">
<instance>
<result>明天北京的天气怎么样？ </result>
</instance>
<input mode="speech">明天北京的天气怎么样？ </input>
<asr>明天北京的天气怎么样？ |420|3485|84e2b71d2a2c443281a61279bd336c8e|utter-8kHz-b993a5d433e44dd3-
1.pcm</asr>
</interpretation>
</result>
```



### 8.2.2 直接发送 Recognize

不发送 Define-Grammar 请求，直接发送 Recognize 请求。

C->S:

```
MRCP/2.0 279 RECOGNIZE 1
Channel-Identifier: f0d5a2f67ce411e7@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Recognition-Timeout: 5000
Start-Input-Timers: true
No-Input-Timeout: 5000
Content-Length: 50

builtin:grammar/pizza_yesno?language=en-US;y=1;n=2
```

后面的交互和上一节 S->C 的返回规范一致。

### 8.2.3 先发送 Recognize 再发送 Start-Input-Timers

C->S(注意这里 Start-Input-Timers 为 false):

```
MRCP/2.0 279 RECOGNIZE 1
Channel-Identifier:00a6cbfa326240c2@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Recognition-Timeout: 5000
Start-Input-Timers: false
No-Input-Timeout: 5000
Content-Length: 50

builtin:grammar/pizza_yesno?language=en-US;y=1;n=2
```

S->C:

```
MRCP/2.0 83 1 200 IN-PROGRESS
Channel-Identifier: 00a6cbfa326240c2@speechrecog
```

C->S(注意 IVR 单独发送了 START-INPUT-TIMERS 请求):

```
MRCP/2.0 86 START-INPUT-TIMERS 2
Channel-Identifier:00a6cbfa326240c2@speechrecog
```

S->C:

```
MRCP/2.0 80 2 200 COMPLETE
Channel-Identifier:00a6cbfa326240c2@speechrecog
```

S->C:

```
MRCP/2.0 94 START-OF-INPUT 1IN-PROGRESS
Channel-Identifier: 00a6cbfa326240c2@speechrecog
```

S->C:

```
MRCP/2.0 654 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: 00a6cbfa326240c2@speechrecog
消息体内容和上一节的规范保存一致，略过不写
```

说明：

6.2.2 和 6.2.3 节的区别在于 6.2.2 节的流程中在 Recognize 消息里直接设置了 Start-Input-Timers 为 true，此时要求 ASR 立刻对接收到的语音进行检测、识别和无话超时计时；而 6.2.3 节流程中时发送完 Recognize 消息之后，客户端又发送了 Start-Input-Timers 消息，在此消息里才设置 Start-Input-Timers 为 true，该场景下 ASR 务必在 Start-Input-Timers 为 true 时再进行无话超时计时(据此实现语音打断功能，比如在 TTS 播放完成之前，不发送 Start-Input-Timers 请求，此时 SDM 不开启无话超时计时，也即不会提前返回 No-Input-Timeout)。

### 8.2.4 连续语音识别请求

**注：连续语音识别并不是标准协议支持的功能，需要对协议进行扩展实现。**

C->S(注意这里 xml 的格式，如果传递 model 参数，必须设置为 continuous):

```
MRCP/2.0 454 DEFINE-GRAMMAR 1
Channel-Identifier: eec0eee653e947ac@speechrecog
Content-Type: application/srgs+xml
```

```
Content-Id: request1@form-level.store
Content-Length: 275

<?xml version="1.0"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US" version="1.0" mode="continuous"
root="digit">
   <rule id="digit">
      <one-of>
         <item>one</item>
         <item>two</item>
         <item>three</item>
      </one-of>
   </rule>
</grammar>
```

S->C:

```
MRCP/2.0 112 1 200 COMPLETE
Channel-Identifier: eec0eee653e947ac@speechrecog
Completion-Cause: 000 success
```

C->S:

```
MRCP/2.0 327 RECOGNIZE 2
Channel-Identifier: eec0eee653e947ac@speechrecog
Content-Type: text/uri-list
Cancel-If-Queue: false
Start-Input-Timers: true
No-Input-Timeout: 5000
Speech-Complete-Timeout: 4000
Vendor-Specific-Parameters: name111=param_value;name222=1111
Content-Length: 33

session:request1@form-level.store
```

S->C:

```
MRCP/2.0 83 2 200 IN-PROGRESS
Channel-Identifier: eec0eee653e947ac@speechrecog
```

S->C:

```
MRCP/2.0 94 START-OF-INPUT 2 IN-PROGRESS
Channel-Identifier: eec0eee653e947ac@speechrecog
```

S->C(注意事件头：INTERMEDIATE-RESULT):

```
MRCP/2.0 617 INTERMEDIATE-RESULT 2 IN-PROGRESS
Channel-Identifier: eec0eee653e947ac@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-eec0eee653e947ac-1.pcm
Content-Type: application/nlsml+xml
Content-Length: 379
```

```
<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="session:request1@form-level.store" confidence="1">
<instance>
<result>聪明的 et，您好</result>
</instance>
<input mode="speech">聪明的 et，您好</input>
<asr>聪明的 et，您好|390|2675|117150cfe84042899d07f7dba34d841c|utter-8kHz-eec0eee653e947ac-
1.pcm</asr>
</interpretation>
</result>
```

S->C(注意事件头：INTERMEDIATE-RESULT):

```
MRCP/2.0 811 INTERMEDIATE-RESULT 2 IN-PROGRESS
Channel-Identifier: eec0eee653e947ac@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-eec0eee653e947ac-1.pcm
Content-Type: application/nlsml+xml
Content-Length: 573

<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="session:request1@form-level.store" confidence="1">
<instance>
<result>能帮我推荐一个好的按摩店吗？ </result>
</instance>
<input mode="speech">能帮我推荐一个好的按摩店吗？ </input>
<asr>能帮我推荐一个好的按摩店吗？ |4240|10265|117150cfe84042899d07f7dba34d841c|utter-8kHz-
eec0eee653e947ac-1.pcm</asr>
</interpretation>
</result>
```

注意在连续语音识别模式中和上文介绍的交互日志有几处差异：

1. 调用时设置连续识别模式(以下三种方式均可以)：
   a) 在发送 Define-Grammar 或者 Recognize 请求的消息体里在 xml 格式里设置 mode="continuous"(上面示例流程里有 Define-Grammar 格式)；
   b) 或者在发送 Recognize 请求时设置消息头：
   Recognize-Mode: continuous
   c) 或者修改 SDM 配置文件 alimrcp-server.xml：
   <param name="recognize-mode-continuous"  value="0"/>  → 改为 1
2. 返回格式
   返回给 IVR 的识别结果事件和前文介绍的基本一致，但有两个关键性的不同：
   a) 不同于标准协议规定的识别结果返回事件 RECOGNITION-COMPLETE，在连续识别时返回的事件为 INTERMEDIATE-RESULT，该事件并不是标准协议中定义的，而是对标准协议的扩展实现，该扩展相对通用，但也需考虑各个呼叫中心是否支持；

b) 在标准协议的定义中，完成一次语音识别时，返回给 IVR 一个 RECOGNITION-COMPLETE 事件，然后结束本次的语音识别，除非 IVR 再次发送新的 RECOGNIZE 请求触发下一次的识别，而在扩展的连续语音识别中，IVR 只需要发送一次 RECOGNIZE 请求，会持续对传递的语音进行不间断的语音识别，并持续的返回识别结果给到 IVR 侧，除非 IVR 主动发送结束请求，否则该链接不会断开识别。

3. 其他说明

在使用 MRCP 的场景中，常见的有语音导航(呼入)、语音外呼(呼出)，随着语音技术和业务需求的发展，一种新的双呼场景也发展了起来，该场景不同于导航/外呼那种用户和机器人的交互，而是用户和坐席两个人的交互，通过对用户、坐席两路语音流的实时识别，并将两侧的识别结果实时推送到坐席端，再加上语义理解技术，可以发挥理解用户意图、检测坐席的情绪稳定、给坐席相应话术提示的优势，也即“话术辅助”。

作为一个新产生的业务场景，标准 MRCP 协议中并没有对这一场景给出相关描述。通过对标准协议的扩展实现，能够比较好的支持该场景需求，但带来的风险是该机制不在协议规范之内，不同厂家的支持粒度也有差异。

# 9. 问题排查

## 9.1 日志说明

本应用总共有三种日志：

- alimrcp-server.log
  交互日志，是最全的日志流，会记录和 IVR、和 ASR 服务、和 TTS 服务的交互日志。
- sdm-asr-request.log
  ASR 请求的访问日志，可以类比为 nginx 服务的 access 日志，一行一条请求记录。
- sdm-tts-request.log
  TTS 请求的访问日志，可以类比为 nginx 服务的 access 日志，一行一条请求记录。

### 9.1.1 alimrcp-server.log

是本应用最全的日志，可以用来分析各类问题，记录了和 IVR、ASR、TTS，以及可能的 NLU 交互的相关日志。

对于和 IVR 的交互来说，每一个新的请求都是以“Remote SDP”(也即 SIP 请求)和“Add Session”开始的。该“Add Session”对应的 id 非常重要，是 SDM 和 IVR 以及各后端服务调用时的中间桥梁，通过该 id 可以找到和 IVR 交互的详细信息，对 ASR/TTS 的调用信息。



### 9.1.2 sdm-asr-request.log

日志规范示例：

```
2019-02-02 16:26:04.629|INFO|172.17.0.2|fa3777b5e22c4a9e|utter-8kHz-fa3777b5e22c4a9e-
1.pcm|c6ce28fbf96241cc8a6b205a90c2cdd8|明天北京的天气怎么样?
|14270|17335|848|||0|SUCCESS|name111=param_value;name222=1111
2019-02-02 18:21:16.186|INFO|172.17.0.2|d2869f893b684cea|utter-8kHz-d2869f893b684cea-
1.pcm|||0|0|0|||0|Meta:APPKEY_NOT_EXIST:Appkey not exist!|name111=param_value;name222=1111
```

每行一条，对应一次 RECOGNIZE 请求，格式如下(有些字段可能为空)：
时间点|INFO|本机 IP|Session Id|本次 RECOGNIZE 的录音文件名|ASR 服务的 task_id|ASR 识别结果|有效语音开始时间点|有效语音结束时间点|Client 端的体感延迟|NLU 结果|NLU 延迟|状态信息|Vendor-Specific-Parameters 值

以第一条日志为例：

✓ Session Id：fa3777b5e22c4a9e

该 id 是 IVR 和 SDM 交互时的唯一 id，每一次新的请求(比如从 sip 请求角度来说)会产生一个新的 id，但一通电话的多次 RECOGNIZE 请求会使用同一个该 id，以表示上下文。该 id 是排查 IVR 和 SDM 交互时非常重要的标识符。

- ✓ **本次 RECOGNIZE 的录音文件名**：utter-8kHz-fa3777b5e22c4a9e-1.pcm
  本次 RECOGNIZE 请求时保存下来的对应文件名(如果开启保存的话)，一通电话里，第一次 RECOGNIZE 请求的文件名为：utter-8kHz-sessiondid-1.pcm，第二次 RECOGNIZE 请求的文件名为：utter-8kHz-sessiondid-1.pcm，以此类推，会保存在本地 var 目录下，也会把文件名返回给 IVR。
- ✓ **ASR 服务的 task_id**：c6ce28fbf96241cc8a6b205a90c2cdd8
  每一次 RECOGNIZE 请求都会调用 ASR 服务，都会产生一个新的 task_id，该 id 是排查 SDM 和 ASR 服务交互时非常重要的标识符，如果需要排查 ASR 链路，需要提供该 id。
- ✓ **ASR 识别结果**：明天北京的天气怎么样?
  每一次 RECOGNIZE 请求对应的识别结果，和上面的 utter-8kHz-fa3777b5e22c4a9e-1.pcm、c6ce28fbf96241cc8a6b205a90c2cdd8 一一对应。
- ✓ **有效语音开始时间点**：14270
  每收到 IVR 的一次 RECOGNIZE 请求，表示开始识别，但用户可能没有说话，那么此时语音都是静音；该时间点即是相对于 RECOGNIZE 来说，用户开始说话的起止点，单位毫秒。
- ✓ **有效语音结束时间点**：17335
  每收到 IVR 的一次 RECOGNIZE 请求，表示开始识别，但用户可能没有说话，那么此时语音都是静音；该时间点即是相对于 RECOGNIZE 来说，用户说话结束时的结尾点，单位毫秒。
- ✓ **Client 端的体感延迟**：848
  相对于用户说话结束后到收到完整 ASR 识别结果的延迟时间，也就是说该时间包括了 VAD 的尾点判停时间，也即：VAD 时间(依赖实际设置的时间)+ASR 耗时(如果关注过 ASR 自身耗时的话，会发现该段时间基本都在 200ms 以内)。
  另外，由于从用户侧来看这个延迟的话，往往会需要人工分析 SDM 录音或者 IVR 录音来确认(下文会介绍)，这里给出的这个时间有一定误差。
- ✓ **NLU 结果**：
  如果调用了 NLU，这里输出 NLU 结果
- ✓ **NLU 延迟**：0
  如果调用了 NLU，这里输出 NLU 调用的延迟。
- ✓ **状态信息**：SUCCESS
  本次请求的状态信息。
- ✓ **Vendor-Specific-Parameters 值**：name111=param_value;name222=1111
  IVR 传递过来的 Vendor-Specific-Parameters 参数值。



### 9.1.3 sdm-tts-request.log

日志规范示例：

```
2019-02-03 11:12:23.021|INFO|172.17.0.2|632dc298277a4264|欢迎使用智能语音交互产品
|f1bcc5d9300a47488c248abbf54c6e0f|361|SUCCESS
2019-02-03 11:16:33.999|INFO|172.17.0.2|a414269cf0ac4fb5|欢迎使用智能语音交互产品
|9eb6f977d2a24dbe98516b77a53afc69|0|TTS:TtsClientError:SetVoice RetCode[2] invalid voice name
```

每行一条，对应一次 SPEAK 请求，格式如下(有些字段可能为空)：
时间点|INFO|本机 IP|Session Id|本次 SPEAK 的合成文本|TTS 服务的 task_id|本次 TTS 调用的首包延迟|状态信息

以第一条日志为例：

- ✓ **Session Id：** 632dc298277a4264
  该 id 是 IVR 和 SDM 交互时的唯一 id，每一次新的请求(比如从 sip 请求角度来说)会产生一个新的 id。该 id 是排查 IVR 和 SDM 交互时非常重要的标识符。
- ✓ **本次 SPEAK 的合成文本：** 欢迎使用智能语音交互产品
  本次请求的合成文本。在客服场景下，不建议该文本太多，否则可能影响到交互效果和用户体验。
- ✓ **TTS 服务的 task_id：** f1bcc5d9300a47488c248abbf54c6e0f
  每一次 SPEAK 请求都会调用 ASR 服务，都会产生一个新的 task_id，该 id 是排查 SDM 和 TTS 服务交互时非常重要的标识符，如果需要排查 TTS 链路，需要提供该 id。
- ✓ **本次 TTS 调用的首包延迟：** 361
  首包延迟即是指第一次收到 TTS 服务返回的语音流时，相对于请求发出时的延迟，也即是本次调用的需要重点关注的耗时，这是由于 TTS 的语音流是流式返回，IVR 播放 TTS 也是边收边播，所以我们只需关注该首包延迟即可。
  另外，TTS 服务的完整合成时间可能会大于该延迟(特别是文本较多的情况)，IVR 播放 TTS 语音的总时间必然远远大于该延迟，因为是和实际合成的语音流时长有关(如果从交互上分析的话，总时间是 IVR 发起 SPEAK 和收到 SPEAK-COMPLETE 的时间差)。
- ✓ **状态信息：** SUCCESS
  本次请求的状态信息。



## 9.2 错误码快查

### 9.2.1 ASR 请求相关错误

一旦请求 ASR 服务出现错误，IVR 平台则会收到 MRCP-SERVER 返回的 006 recognize-error 错误信息，受限于标准协议，并不会将 ASR 服务具体的错误信息返回给 IVR(也即 ASR 厂商特定错误信息)，此时需要从 MRCP-SERVER 的相关日志中分析。具体日志可以从 log 目录下的 alimrcp-server.log 和 sdm-asr-request.log 中查找。

#### 9.2.1.2 APPKEY 不存在或者不合法

```
2019-02-02 18:21:16.186|INFO|172.17.0.2|d2869f893b684cea|utter-8kHz-d2869f893b684cea-1.pcm|||0|0|0|||0|Meta:APPKEY_NOT_EXIST:Appkey not exist!|name111=param_value;name222=1111
```

如上，出现“Meta:APPKEY_NOT_EXIST:Appkey not exist!”的错误信息，需要重新在配置文件中设置 APPKEY，或者检查该 APPKEY 是否与令牌归属同一个阿里云账号。

此时在 alimrcp-server.log 日志中 grep “AsrClient::onOperationFailed”也可以发现相应错误信息，同时上面出现的 d2869f893b684cea 即是本次识别错误的 session id:

```
AsrClient::onOperationFailed(d2869f893b684cea)(40020105)({"header":{"namespace":"Default","name":"TaskFailed","status":40020105,"message_id":"9c45cfb7d18f4c83b540ff676678b753","task_id":"47ce068107404cbcbc5c5a4518e597f1","status_text":"Meta:APPKEY_NOT_EXIST:Appkey not exist!"}})
```

注意这里面的错误信息 status_text 字段和 status 字段，其中 status 字段对应的错误码 40020105 可以参考 SDK 相关文档，或者从阿里云语音产品页面上查看相应描述：阿里云智能语音识别服务状态码。

另外，需要注意上述日志中出现的 task_id 字段 47ce068107404cbcbc5c5a4518e597f1，通过该 id 可以查到整条调用链路(包括 ASR 服务)的日志。往服务端排查时需要依赖 id。

## 9.2.2 TTS 请求相关错误

一旦请求 TTS 服务出现错误，IVR 平台则会收到 MRCP-SERVER 返回的 004 error 错误信息，受限于标准协议，并不会将 TTS 服务具体的错误信息返回给 IVR(也即 TTS 厂商特定错误信息)，此时需要从 MRCP-SERVER 的相关日志中分析。具体日志可以从 log 目录下的 alimrcp-server.log 和 sdm-tts-request.log 中查找。

### 9.2.2.1 APPKEY 不存在或者不合法



同调用 ASR 出现该问题时的排查思路一致，这里略过不提。

### 9.2.2.2 发音人设置不正确

```
2019-02-03 10:16:33.999|INFO|172.17.0.2|e9beeb9503d14703|欢迎使用智能语音交互产品|4e08f8b2989c4a0798a5264adf94479d|0|TTS:TtsClientError:SetVoice RetCode[2] invalid voice name
```

在 sdm-tts-request.log 中发现了上述错误“invalid voice name”，表示是设置了错误的发音人，由于设置发音人的地方有两处（前文有介绍）：本地 service-tts.json 文件中的 voice 字段和 IVR 发送的 SPEAK 请求中 Voice-Name 字段），所以需要从这两处确认发音人是否正确(是否存在以及当前 TTS 服务是否支持该发音人)。

## 9.3 语音识别率低的排查

1. 确认模型版本

   i. 如果调用的是公共云上的 ASR，需要先确认客户阿里云管控台上 appkey 对应的模型采样率（通过 MRCP 调用时 appkey 是通过 conf/service-asr.json 中 AppKey 字段设置的），需要选择 8000 采样率的模型：


ii. 如果调用的是专有云下的 ASR，查看专有云 ASR 基础模型(models/alisr_model/readme.txt)是什么采样率是多少(8k 或者 16k)， 由于通过 MRCP 协议的场景都是电话客服场景，语音采样率都是 8000 的，所以 ASR 模型也需要是 8k 的；

2. 确认语音质量
MRCP-SERVER 会保存录音(var 目录，如果开启保存的话)，命名方式一般是 utter-8kHz-{session id}-{序号}.pcm，比如：

```
MRCP/2.0 552 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: 91062051affb4720@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-91062051affb4720-1.pcm
Content-Type: application/nlsml+xml
Content-Length: 314

<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="session:request1@form-level.store" confidence="1">
<input mode="speech">喂，你好喂，你好喂，你好</input>
<asr>喂，你好喂，你好喂，你好|1670|6325|cdb3e027733a4ca394d88bc8629c2cea</asr>
</interpretation>
</result>
```

以上表示 utter-8kHz-91062051affb4720-1.pcm 文件对应的识别结果为“喂，你好喂，你好喂，你好”，该条语音即是调用方发过来的原始语音，通过复听该条语音，如果质量很差(比如噪音很大，环境很杂等)，则需要从调用方侧来分析语音质量问题；如果是该语音听着很清晰但和识别结果差异很大，则需要 ASR 算法同学来进行分析。

## 9.4 识别延迟高，识别慢的排查

MRCP-SERVER 对 ASR 的调用和客户通过 SDK 调用的机制其实是一样的(如果使用过 SDK 的话)，可以把 MRCP-SERVER 想象成 SDK 程序；除此之外，可以通过 MRCP-SERVER 自身来做一个分析：

```
2018-09-11 19:17:39:121046 [INFO]    Detected Newer Text:56c3caff193f4201a5338b37b4b62a33|是我啊，怎么了？ |2920|4785|completed <d4f95b772279457c>
```

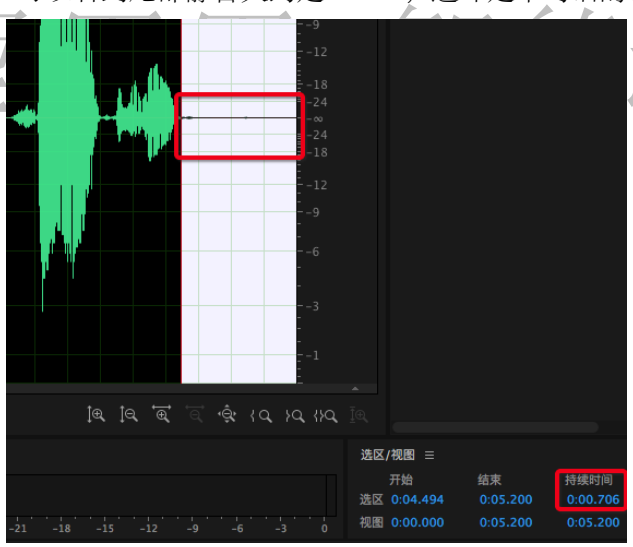
```
2018-09-11 19:17:39:123355 [INFO]    Send MRCPv2 Data 172.17.139.245:1544 <-> 172.17.139.238:29247 [472
bytes]
MRCP/2.0 472 RECOGNITION-COMPLETE 2 COMPLETE
Channel-Identifier: d4f95b772279457c@speechrecog
Completion-Cause: 000 success
Waveform-Uri: utter-8kHz-d4f95b772279457c-1.pcm
Content-Type: application/nlsml+xml
Content-Length: 236

<?xml version="1.0" encoding="utf-8"?><result>
<interpretation grammar="grammar_name" confidence="1">
<instance>是我啊，怎么了？ </instance>
<input mode="speech">是我啊，怎么了？ </input>
</interpretation>
</result>
```

从上面日志(log/alimrcp-server.log) 里可以看出：

1. 该次请求的 request-id 是 56c3caff193f4201a5338b37b4b62a33， 识别结果是：“是我啊，怎么了？”， 录音文件是 utter-8kHz-34c17a1bc4724786-5.pcm;
2. 此时可以拿 56c3caff193f4201a5338b37b4b62a33 按上面 sdk 调用的方式进行向后排查；同时可以通过 utter-8kHz-d4f95b772279457c-1.pcm 的尾部静音来确定最大延迟是多少;
3. 通过分析语音，可以看到尾部静音大约是 700ms，这即是本句话的识别延迟；






需要说明的是，该延迟和上面介绍的 VAD 静默时长有一定关系，假设 VAD 的静默时长是 X ms，再加上 ASR 识别耗时和网络链路延迟(假设 M ms)，则可以预估的尾部静音时长大约是： (X +M) ms。

## 9.5 语音打断慢，延迟大的排查

在语音导航或者智能外呼场景下，往往会把语音打断作为核心功能点，也就是在机器人播放的时候，如果用户说话，可以打断机器的播放，在实际场景里，测试人员主观上会感觉

打断的慢，也就是用户说了好久的话，机器人才停止播放，这里给出一个排查方式（类似上面关于 ASR 识别延迟高的排查）。

在介绍排查方式之前，先简单描述下打断的实现机制：MRCP-SERVER 在检测用户语音时，如果判断用户说话了，会返回一个事件给 ivr（START-OF-INPUT 事件），IVR 收到此事件时，需立刻停止机器人播放（不论是 TTS 播放还是固定录音播放）。

假设此次 IVR 和 MRCP-SERVER 之间的 ASR Session Id 是 897a3c242b714185，需要确认以下几个信息：

1. 收到 ivr 发送的 RECOGNIZE 时间点：

```
2018-10-30 11:32:36:416640 [INFO]    Receive MRCPv2 Data 172.16.0.112:1544 <-> 172.16.0.110:57974 [283 bytes]
MRCP/2.0 283 RECOGNIZE 1
Channel-Identifier: 897a3c242b714185@speechrecog
Content-Type: text/uri-list
Start-Input-Timers: false
No-Input-Timeout: 5000
Speech-Complete-Timeout: 500
Content-Length: 23

builtin:grammar/boolean
2018-10-30 11:32:36:416705 [INFO]    Process RECOGNIZE Request <897a3c242b714185@speechrecog> [1]
```

从图上可以根据 RECOGNIZE 或者“Process RECOGNIZE Request”字段确定：收到的 RECOGNIZE 请求时间点是 “2018-10-30 11:32:36:416”；

2. MRCP-SERVER 返回 START-OF-INPUT 的时间点：

```
2018-10-30 11:32:41:823593 [INFO]    onSentenceBegin(897a3c242b714185)(20000000)(…)
2018-10-30 11:32:41:831268 [INFO]    Process START-OF-INPUT Event <897a3c242b714185@speechrecog> [1]
2018-10-30 11:32:41:831344 [INFO]    Send MRCPv2 Data 172.16.0.112:1544 <-> 172.16.0.110:57974 [94 bytes]
MRCP/2.0 94 START-OF-INPUT 1 IN-PROGRESS
Channel-Identifier: 897a3c242b714185@speechrecog
```

从图上可以根据 “onSentenceBegin”、 “Process START-OF-INPUT Event”、“START-OF-INPUT”字段，确认返回给 IVR 的时间点是：“2018-10-30 11:32:41:831”；

3. 对应的语音 pcm 文件(在 MRCP-SERVER 的 var 目录)，在返回的 RECOGNITION-COMPLETE 事件有如下内容：

```
2018-10-30 11:32:44:071433 [INFO]    Send MRCPv2 Data 172.16.0.112:1544 <-> 172.16.0.110:57974 [590 bytes]
MRCP/2.0 590 RECOGNITION-COMPLETE 1 COMPLETE
Channel-Identifier: 897a3c242b714185@speechrecog
Completion-Cause: 000 success
```

```
Waveform-Uri: utter-8kHz-897a3c242b714185-1.pcm
Content-Type: application/nlsml+xml
```

从上面的信息可以看到此次识别对应的录音文件是 utter-8kHz-897a3c242b714185-1.pcm(位于 var/目录，前提是开启了语音保存机制)：

确定了以上三点之后，可以得出以下结论：

1. 从收到 RECOGNIZE 到返回 START-OF-INPUT，时间相差 5.4 秒：
2. 从语言波形图上得到用户实际说话开始时间大约是在第 5 秒钟；
3. 从 1，2 可以得出，从用户开始说话，到 MRCP-SERVER 返回了 START-OF-INPUT 事件(表示用户已经说话)，延迟了： 5.4 - 5.0 = 0.4 秒；

另外，这里没有把 IVR 收到 START-OF-INPUT 事件之后触发对机器人的停止播放时间(如果是 TTS，即发起 STOP 请求)计算进去，但实际上是一样的分析思路。

## 9.6 其他常见错误



### 9.6.1 DNS 解析错误

```
AsrClient::onOperationFailed··· (10000003)(DNS: resolved timeout., start finised.)
```

或者

```
TtsClient::onOperationFailed···(10000003)(DNS: resolved timeout., start finised.)
```

注意这里的 10000003 和 DNS: resolved timeout.出现这种错误，一般是本机连接服务端引擎出现问题，需要检查：

1. 确认配置的 url 地址是否正确：

```
grep url conf/service-asr.json
grep url conf/service-tts.json
```

2. 确认本机 DNS 配置：
   查看/etc/resolv.conf，建议只保留 nameserver 114.114.114.114